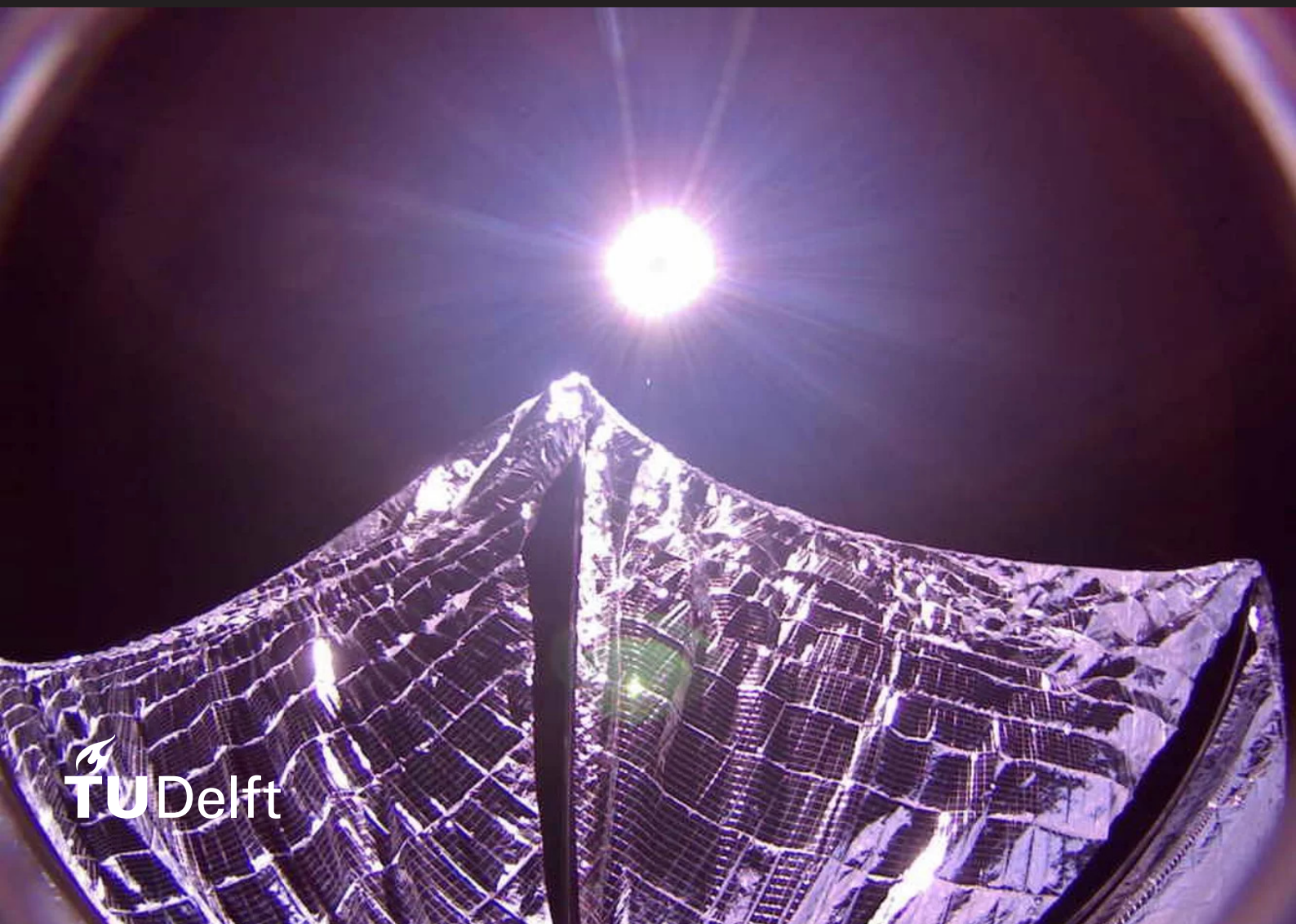
Optimizing a Solar Sailing Polar Mission to the Sun

Development and Application of a New Ant Colony Optimizer

Giacomo Acciarini



Optimizing a Solar Sailing Polar Mission to the Sun

Development and Application of a New Ant Colony Optimizer

by

Giacomo Acciarini

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday November 1, 2019 at 1:30 PM.

Student number:	4754883	
Project duration:	January 4, 2019 – November 1, 2019	
Thesis committee:	Prof. Dr. P. Visser,	TU Delft, chair
	Dr. ir. E. Mooij,	TU Delft, supervisor
	Dr. F. Oliviero,	TU Delft
	Dr. D. Izzo,	European Space Agency

This thesis is confidential and cannot be made public until December 31, 2020.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

Cover image credit: `https://www.theguardian.com/science/2015/jun/11/spacewatch-lightsail-deploys-solar-sail`, date of access: August 2019.

TU Delft

Preface

After months of hard work, I am proud to present to you my thesis to obtain the Master of Science degree at the faculty of Aerospace Engineering, at the Delft University of Technology. It has been an amazing experience that allowed me to come in contact with many researchers from all over the world who share my same love and interest in space exploration. I have always felt privileged and honored to be able to study such a marvelous and challenging subject. First of all, I would thus like to thank my parents, Andrea and Manuela, and my sister, Chiara, without your help and support I would have never been able to reach this point. Your confidence in me has never wavered and I will always be grateful for that. My gratitude is also directed to my daily supervisor, Erwin, without you, this would have never been possible. Thank you for your support and precious suggestions. Moreover, I would like to thank Dario Izzo, from the European Space Agency's Advanced Concepts Team, for his guidance in the development of the optimization software. His innovative and creative ideas, as well as his commitment and long term experience in the field, have been a source of inspiration for me. Also, thank you for offering me the SOCIS summer job: it has been a crucial element for better understanding and diving into the software development world. Finally, special thanks go to all my friends from Perugia (with special mention to Riccio, Mao, Nikki and Greg), and to my girlfriend, Ursula. Crossing entire Europe by car for coming to my thesis presentation is the umpteenth demonstration of our deep and strong bond.

Giacomo Acciarini
Delft, November 1, 2019

Abstract

The Sun is the main contributor to birth and thriving of life on Earth. Yet, a little is known about many physical phenomena that happen on it and that influence the behavior and well being of all the planets in the Solar System. Moreover, as we rely more in technology, our dependency on the Sun increases, and we need to be prepared to limit the damages of disruptive events in advance. Sunspots and other features of the Sun are strongly related to these phenomena, and their study and observation might increase the predictability of these events. However, to monitor and properly investigate these phenomena we need a mission that can offer us a unique and advantaged point of view of the Sun. The aforesaid scientific aspects can only be addressed with a very inclined solar mission, which is near enough to the Sun. Furthermore, since the current technology does not make possible to reach polar orbits in the proximity of the Sun, a solar-sail is pivotal to reach this objective. In particular, the optimal orbit for achieving the objectives would be a circular orbit with 90°inclination with respect to the heliographic equator. These aspects are the cornerstone of our research and have led to the formulation of the following research question:

Can the time and cost of a solar-sail mission to the Sun be optimized by using a global optimization technique?

In literature, there have already been some studies for optimizing a solar sailing polar mission towards the Sun. However, this has never been done using an ant colony optimizer and considering multiple objectives (i.e., cost and duration). In this research, we would like to implement a new global ant colony single and multi-objective optimizer with the aim to reduce the duration and cost of such a mission. Although ant colony optimization has recently demonstrated to be very powerful in solving trajectory optimization problems in space missions, this algorithm is not available in an open-source fashion and it has never been applied to such a mission. The main purpose of this research is thus to generate new optimal trajectories for a solar sailing polar mission around the Sun by implementing a novel ant colony optimizer.

First of all, to achieve this, a simulation model to represent the solar-sail orbits around the Earth and the Sun is set up. In this simulator, the solar radiation pressure force, the atmospheric forces and other environmental aspects are included. Also, a guidance model for the sail is set up, so that the attitude of the sail can be controlled during its journey to the Sun. This entire framework is then formulated as both a single and multi-objective problem. Each of these problems is optimized with the novel ant colony optimizer developed in this research. The results are then also compared and traded-off with other state-of-art global optimizers. In particular, for single-objective, six different optimizers have been benchmarked with our ant colony technique: artificial bee colony (ABC), standard differential evolution (DE), a standard evolution variant (DE1220), self-adaptive differential evolution (SADE), particle swarm optimization (PSO) and simple genetic algorithm (SGA). Whereas for the multi-objective case, three different optimizers have been tested against the multi-objective ant colony extension: multi-objective evolutionary algorithm with decomposition (MOEA/D), nondominated sorting genetic algorithm (NSGA-II) and nondominated sorting particle swarm optimization (NSPSO). The found results are very promising: for the single-objective problem, the ant colony optimizer has managed to find the best overall solution. On the other hand, for the multi-objective case, NSGA-II seems to provide the best solutions, although the multi-objective ant colony optimizer displays a set of solutions that is competitive with those of NSGA-II, especially for lower function evaluations, and that outperforms both NSPSO and MOEA/D.

Once the mission was optimized, the best found mission profile was studied. It was found that the best overall solution happens for the multi-objective case: indeed, in this case, we managed to halve the mass of the single-objective mission, while still keeping a similar time

of flight. Interestingly, we also discovered that a flyby to the Moon is crucial when the time of flight has to be strongly reduced. We hence recommend considering such a gravity assist in future studies.

List of Acronyms

ABC	Artifical Bee Colony
ACO	Ant Colony Optimization
ACOmi	single-objective mixed integer Ant Colony Optimizer
ASA	Adaptive Simulated Annealing
CD	Crowding Distance
CEC2006	Congress on Evolutionary Computation 2006
CME	Corona Mass Ejections
CPU	Central Processing Unit
CSTRS	selft-adaptive constraints handling meta-algorithm
DE	Differential Evolution
DE1220	Differential Evolution variant
DLR	german aerospace center
DTLZ	Deb, Thiele, Laumanns and Zitzler
EA	Evolutionary Algorithm
EACO	Extended Ant Colony Optimization
EP	External Population
ESA	European Space Agency
GA	Genetic Algorithm
GEO	GEOcentric
GR	Golomb Ruler
GTO	Geostationary Transfer Orbit
GTOP	Global Trajectory Optimization Problems
H	Heliocentric
IHS	Improved Harmony Search
IMF	Interplanetary Magnetic Field
JAXA	Japanese Aerospace Exploration Agency
JD	Julian Day
LEO	Low Earth Orbit
M2P2	Mini magnetospheric Plasma propulsion
MaxMin	Maximum Minimum diversity strategy
MC	Monte Carlo
MEE	Modified Equinoctial Elements
MHACO	Multi-objective Hypervolume-based Ant Colony Optimizer
MIDACO	Mixed Integer Distributed Ant Colony Optimization
MINLP	Mixed Integer Non Linear Programming
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOP	Multi-Objective Problems
MPSO	Modified Particle Swarm Optimization
MRP	Modified Rodrigues Parameters
NASA	National Aeronautics and Space Administration
NC	Niche Count
NEP	Nuclear Electric Propulsion
NSGA	Nondominated Sorting Genetic Algoritm
NSPSO	Nondominated Sorting Particle Swarm Optimizer
OKEANOS	Outsized Kite-craft for Exploration and AstroNautics in the Outer Solar system
PaGMO	Parallel Global Multi-objective Optimizer

PyGMO	Python parallel Global Multi-objective Optimizer
PF	Pareto Front
pop	population
PSO	Particle Swarm Optimization
REP	Radioisotope Electric Propulsion
SA	Simulated Annealing
SADE	Self-Adaptive Differential Evolution
SGA	Simple Genetic Algorithm
SO	Single-Objective
SBX	Simulated Binary crossover
SEP	Solar Electric Propulsion
SMRP	Shadow Modified Rodrigues Parameters
SPICE	Spacecraft Planet Instrument C-matrix Events
tof	time of flight
Tudat	TU Delft astrodynamics toolbox
USM	Unified State Model
V&V	Verification and Validation
WFG	Walking Fish Group
ZDT	Zitzler Deb and Thiele

List of Symbols

A	sail area [m^2]
a	semi-major axis [m]
$\hat{\mathbf{a}}$	Euler axis [-]
a_0	characteristic acceleration [m/s^2]
a_g	solar gravitational acceleration [m/s^2]
$\mathbf{a}_{pert}$	perturbing acceleration vector [m/s^2]
B	non-Lambertian coefficient [-]
C	mission cost [$]
C	reflectivity [-]
$\mathbf{C}$	velocity component vector normal to the radial vector laying on the orbital plane [m/s]
cdf	cumulative distribution function [-]
C_r	reflectivity coefficient of the front sail [-]
CR	Crossover probability [-]
c	speed of light [m/s]
D	distances between ants [-]
D	weight generation method [-]
D	drag [N]
DI	distribution index [-]
d_n	individuals' distance [-]
E	relativistic mass of a particle [kg m^2/s^2]
e	eccentricity [-]
$evalstop$	evaluation stopping criterion [-]
$\mathbf{F}_{\overline{\alpha}}$	absorption force vector [N]
$\mathbf{F}_{\tau}$	transmission force vector [N]
$\mathbf{F}_{\rho}$	reflection force vector [N]
$\mathbf{F}_{tot}$	total force vector [N]
$F_i(d_j)$	local crowding distance [-]
$\mathbf{F}_s$	specularly reflected force vector [N]
$\mathbf{F}_d$	diffusively reflected force vector [N]
F	front fitness [-]
F	weight coefficient [-]
F	parameter for the differential evolution operator [-]
$\mathbf{f}(\mathbf{x})$	objective functions [-]
f	magnitude of the force resulting from the momentum transfer [N]
$focus$	focus parameter [-]
FIT	fitness function [-]
G	universal gravitational constant [$m^3\ kg^{-1}\ s^{-2}$]
$\mathbf{G}$	generation set [-]
$\mathbf{g}(\mathbf{x})$	inequality constraints [-]
$\#gen$	number of generations [-]
$\mathbf{h}(\mathbf{x})$	equality constraints [-]
$\mathbf{h}$	specific angular momentum vector [m^2/s]
h_p	perigee height [km]
h_a	apogee height [km]
i_{rank}	non-domination rank [-]
i	inclination [deg]

I_{max}	maximum radiant intensity [W/m^2]
$I(k).m$	value of the mth objective function of the kth individual in I [-]
ker	solution archive size [-]
L	lift [N]
L_b	length of deployable boom [m]
l	path length [-]
L	true longitude [-]
L	maximum number of copies reinserted in the population [-]
L_s	luminosity of the Sun [W]
M	mean anomaly [rad]
m	mutation probability [-]
m	number of equality constraints [-]
m_{emit}	mass of the emitter [kg]
m_{coat}	mass of the coating [kg]
m_{sub}	mass of the substrate [kg]
m_{bus}	mass of the bus [kg]
m_p	payload mass [kg]
m_s	sail mass [kg]
m_{sys}	system mass [kg]
m_b	support mass [kg]
m_m	mechanisms mass [kg]
m_{bond}	bonding mass [kg]
m_{tot}	total mass [kg]
m_E	mass of the Earth [kg]
m_S	mass of the Sun [kg]
M_S	mass of the Sun [kg]
m_0	rest mass of the particle [kg]
$\hat{\mathbf{m}}$	non-ideal sail force unit vector [-]
N	number of weight factors [-]
N	number of objective functions [-]
N	neighbourhood parameter [-]
N	size of weight's neighbourhood [-]
n	mean motion [rad/s]
n_a	number of artificial ants [-]
n_{con}	number of continuous probability density function [-]
$NGenMark$	standard deviations convergence speed parameter [-]
n_{int}	number of discrete probability density function [-]
n_i	number of solutions which dominate solution i [-]
NP	population size [-]
P	continuous probability density function [-]
P	probability of considering the neighbourhood [-]
p	momentum of the particle [kg m/s]
p	semi-latus rectum [m]
$p(\mathbf{X})$	penalty function [-]
P_{max}	maximum solar radiation pressure [W/m^{3}s]
P_{ir}	infrared radiation pressure [Pa]
P_{al}	albedo radiation pressure [Pa]
$\mathbf{p}$	environmental parameters [-]
p	number of equality and inequality constraints [-]
p_m	mutation strategy [-]
p_s	selection strategy [-]
p_c	crossover strategy [-]
$popSize$	population size [-]

Q	discrete probability density function [-]
q	dynamic pressure [kg $m^{-1}s^{-2}$]
q	convergence speed parameter [-]
$\mathbf{R}$	velocity component vector shifted 90° ahead w.r.t. the eccentricity vector [m/s]
r	Sun-sailcraft distance [au]
r_0	Sun-Earth distance [au]
$r_{a,f}$	final apocenter distance [m]
$r_{p,f}$	final pericenter distance [m]
$r_{s.i.,E}$	sphere of influence of the Earth [m]
r_{E-S}	Sun-Earth distance [km]
$\mathbf{r}_E$	position vector of the Earth [km]
$res(\mathbf{X})$	residual function [-]
s	coefficient of specular reflection [-]
s	independent variable [-]
s	coefficient of specular reflection [-]
$\mathbf{S}$	solution archive [-]
S_0	solar constant [Wm^{-2}]
S_i	set of solutions which i dominates [-]
ST	Tournament size [-]
T	temperature [K]
t	thickness [m]
T_{eq}	equilibrium temperature [K]
T_i	initial heliocentric ephemeris time [s]
T_{lim}	temperature limit [K]
$T_{mission}$	mission duration [years]
$\mathbf{T}$	tableau factors [-]
T	number of weight vectors in the neighborhood of each weight vector [-]
$threshold$	threshold parameter [-]
$\mathbf{u}$	control parameters [-]
$\mathbf{u}$	trial vector [-]
$U(\mathbf{r})$	potential energy [J/kg]
u	argument of latitude [rad]
$V_{a,f}$	final apocenter velocity [m/s]
V_c	circular velocity [m/s]
$V_{c,a}$	circular apocenter velocity [m/s]
$V_{c,f}$	circular pericenter velocity [m/s]
$V_{c,f}$	final pericenter velocity [m/s]
V_{MAX}	maximum allowed particle's velocity [-]
v_r	radial velocity magnitude [m/s]
v_v	velocity magnitude perpendicular to the radial laying in the orbital plane [m/s]
$\mathbf{v}\|_G$	velocity of the sailcraft in the geocentric inertial frame [m/s]
$\mathbf{v}\|_H$	velocity of the sailcraft in the heliocentric inertial frame [m/s]
W	radiative flux [W/m^2]
W	weights associated to the penalty functions [-]
W_E	solar flux at Earth's position [W/m^2]
$\mathbf{x}$	optimization variables [-]
(x, y, z)	position vector [m]
(x_i, y_i, z_i)	initial heliocentric position vector [km]
$(\dot{x}, \dot{y}, \dot{z})$	velocity vector [m]
$(\dot{x}_i, \dot{y}_i, \dot{z}_i)$	initial heliocentric velocity vector [km/s]
$\overline{\alpha}$	absorption probability [-]
α	cone angle [rad]
α_{lim}	sail cone angle limit [rad]
α	sail cone angle [rad]
β	lightness number [-]

β	clock angle [rad]
δ	clock angle [rad]
ΔT_{final}	final time constraint [s]
ΔV	final velocity constraint [m/s]
ϵ	obliquity [rad]
ϵ_f	front emissivity [-]
ϵ_b	back emissivity [-]
ζ	attack angle between the sail and the velocity vector [rad]
η	sail efficiency factor [-]
η_c	distribution index for crossover [-]
η_m	non-distribution index [-]
η_m	distribution index for polynomial mutation [-]
θ	non-ideal cone angle [rad]
$\boldsymbol{\lambda}$	weight factors [-]
μ	means of the probability density functions [-]
μ_S	gravitational parameter of the Sun [km^3/s^2]
μ_E	gravitational parameter of the Earth [km^3/s^2]
ν	true anomaly [rad]
ρ	density [kg/m^3]
ρ	evaporation parameter [-]
ρ	reflection probability [-]
ρ	air density [$kg\ m^{-3}$]
ρ_s	specularly reflected coefficient [-]
ρ_d	diffusively reflected coefficient [-]
Σ	solution space [-]
σ	sail loading [kg/m^2]
σ	standard deviations of the probability density functions [-]
$\boldsymbol{\sigma}$	MRP vector [-]
$\boldsymbol{\sigma}^S$	SMRP vector [-]
σ	Stefan-Boltzmann constant [$Wm^{-2}K^{-4}$]
η_1	best force magnitude [-]
η_2	neighbourhood force magnitude [-]
τ	pheromone value [-]
τ	transmission probability [-]
τ	time of pericenter passage [s]
Φ	Euler angle [rad]
ϕ	offset angle [rad]
ϕ	ecliptic latitude [rad]
ψ	ecliptic longitude [rad]
ψ	angle between sailcraft velocity vector and Sun line [rad]
Ω	decision variable space [-]
Ω	oracle parameter [-]
Ω	right ascension of the ascending node [rad]
ω	particles' inertia weight [-]
ω	argument of perigee [rad]
ω	weight factors [-]

Contents

Chapter 1

Introduction

Many missions have been exploring the Sun either through heliocentric trajectories or through the Lagrangian point (L1) between the Sun and the Earth (which allows a privileged continuous observation point). These missions have contributed to unveil the mysteries of our star and to start to understand phenomena such as solar wind and coronal mass ejections (CME), which interact with Earth's atmosphere and magnetic field. However, there are many questions that are still unanswered and need to be addressed to enhance the predictability of potentially disruptive phenomena that can originate from the Sun. Many small scale dynamics events have been observed and have demonstrated to be strongly interconnected with the evolution and structure of the solar magnetic field. Furthermore, there is a general consensus that the study and understanding of these phenomena (such as bi-directional jets, blinkers, network flares, bright points, etc.) might enhance our understanding of the acceleration and heating of the solar plasma (Doyle and Madjarska, 2004). Indeed, the solar coronal heating and the energy/mass flow of solar wind are still unsolved but fundamental problems for solar physicists. Some of the more relevant scientific questions still unanswered about the Sun and its environment are (Mueller and Gilbert, 2013):

1. How and where the solar wind plasma and magnetic field originate in the corona?

2. How does the Sun create and maintain the heliosphere?

3. What is the mechanism that drives the Solar dynamo? And how does it affect the interaction between the Sun and the heliosphere?

4. What is the mechanism that drives the coronal mass ejections?

It is important to notice that these phenomena are not only important from a scientific point of view, but also for the welfare and survival of mankind. Indeed, events such as solar flares and coronal mass ejections are equivalent to hurricanes of the space weather and their appearance might cause several blackouts in our navigation systems, electrical systems and communication systems. A solar polar mission to the Sun (inclined 90°with respect to the equator) would help scientists to map these phenomena accurately and possibly answer most of the aforesaid questions (Goldstein et al., 1998). In particular, such a mission would help to address the following scientific objectives:

1. Analyze the heating and acceleration process that affects the solar wind in the corona.

2. Determine magnetic structures and connection patterns in the polar region to model and predict more accurately the occurrence of solar cycles.

3. Discover sources, longitudinal structure, rotational curvature and time variability of corona features.

4. Follow the evolution of the Sun throughout various rotations.

5. Determine the source of the solar wind and its acceleration mechanism.

6. Analyze the fluctuations of sunspots to understand their locations and formations.

Even though there have been missions to the Sun, most of the past missions have been orbiting the Sun within the ecliptic, and even the ones that analyzed the Sun from an orbital plane inclined with respect to the ecliptic (e.g. Ulysses mission [1]), they did not do it neither close enough to study these phenomena thoroughly, nor in circular orbits. The reason is that it is not technologically possible to bend the trajectory at 90°inclination in close proximity to the Sun with the chemical or electric propulsion system, due to the huge amount of propellant required (Macdonald et al., 2006). Hence, the objective of this research is to implement an ant colony optimization algorithm to minimize the cost and duration of a mission to reach a solar polar orbit, using the solar-sail technology. The solar-sail technology is a low-thrust technology, which would allow performing the mission without any propellant consumption. This technology is very powerful when the spacecraft is near the Sun (due to the squared dependency between the thrust provided and the distance from the Sun), hence, it results to be a very viable choice for the mission of interest. Furthermore, several studies have been performed to investigate the feasibility of solar sailing for a solar polar mission and they all seem to confirm that such technology represents a valid option to reach solar polar orbits in a reasonable time (Coverstone and Prussing, 2003), (Goldstein et al., 1998). Nonetheless, the debate is still open about which kind of solar sailing trajectory can optimize the cost and duration of such a mission. Other studies have been performed to optimize a trajectory with the objective to reach a final orbit of 0.40 au radius and 90°inclination around the Sun, starting from a GTO: this final orbit is appealing because it permits to fulfill the scientific objective while also allowing an orbit in a 4:1 resonance with Earth, which would ease the communication with ground and the downlink of data (Candy, 2002), (Garot, 2006), (Spaans, 2009). In this report, we will investigate a solar polar mission using solar sailing technology, and we will focus on the use of an ant colony optimizer for optimizing the mission objectives.

In particular, the main research question of this thesis study can be formulated as follows:

Can the time and cost of a solar sailing polar mission to the Sun be optimized by using a global optimization technique?

Moreover, several subquestions related to the main research question can be derived. These are:

1. How well can an ant colony optimization algorithm perform compared to other already implemented optimization strategies in the framework of a solar sailing polar mission?

a. Is the ant colony optimizer able to find trajectories that do not violate the equality and inequality constraints?

b. How does the ant colony optimizer behave in terms of best found solutions when compared to other optimization algorithms?

2. Is it possible to improve the current solutions for a solar sailing mission by optimizing the trajectory with multiple objectives functions?

a. How does the physical trajectory vary when the cost and duration of the mission are optimized separately?

b. Can we develop a new ant colony optimization algorithm for multiple objectives that is competitive with multi-objective state-of-art algorithms?

3. What is the most suitable way for modeling the trajectory of a sailcraft in a solar sailing mission?

a. What is the influence of the perturbing accelerations in the optimization technique?

b. Can the problem be represented in a simpler and more effective way, without compromising the quality of the solutions?

[1] http://sci.esa.int/ulysses/47369-fact-sheet/, August 2019

From these research question and subquestions, we can derive the main research objective of this study:

The objective of this research is the implementation of an extended ant colony optimizer to be applied on a solar sailing polar mission around the Sun for minimizing the duration and cost of the mission.

This objective can be further divided into multiple sub-objectives:

1. Set-up the state variables, propagator model, integrator model and environment to represent the problem.

2. Develop a guidance scheme, which can model the solar-sail orientation with respect to the Sun during its journey.

3. Develop an extended ant colony optimization algorithm with oracle penalty method for single-objective, continuous/integer variables and constrained/unconstrained problems.

4. Develop a multi-objective extension of the ant colony optimization algorithm.

5. Validate the trajectory simulation using the previous mission concept and validate the optimization algorithm using the several available optimization problems in literature. In particular, all the different extensions of the algorithm will be tested in this phase (integer and continuous variables, etc.). The purpose is to have a single and multi-objective ant colony optimizer competitive with state-of-art global optimization methods.

6. Benchmark the results found with ant colony on the solar sailing polar mission with other optimization algorithms (in particular differential evolution and evolutionary algorithms) for both single and multi-objective.

A brief overview of each chapter of this thesis report can be presented as follows:

Chapter 2 introduces the mission. All previous similar missions (either flown or studied) are investigated and both a reference mission and vehicle are established. Finally, the mission and system requirements are listed.

Chapter 3 discusses the dynamics of the solar sailing mission. Therefore, different state variables, reference systems and equations of motions are defined. Moreover, the physical principle behind the generation of the solar-sail force is also discussed for both the ideal and non-ideal cases. Finally, the environment of such a mission is investigated, by discussing the mathematical model of each perturbation (e.g. atmospheric drag, third body perturbations, etc.).

Chapter 4 investigates the interplanetary journey of the sail, in all its phases. In particular, the journey is divided into a heliocentric and a geocentric phase. In each of these phases, a guidance strategy based on some physical conditions is deduced: this guidance will constitute the baseline for the formulation of the global optimization problem. In fact, this chapter serves to justify the logic behind the strategy used for optimizing the trajectory.

Chapter 5 serves to introduce the global optimizers used in this study. Artificial bee colony, simple genetic algorithm, differential evolution (both adaptive and non-adaptive) and particle swarm optimization are discussed and investigated. Furthermore, several multi-objective optimizers are also treated: these include evolutionary strategies, genetic algorithms, and particle swarm optimizers. Moreover, the ant colony optimizer is introduced by first discussing the single-objective extension and by then defining the multi-objective counterpart. Finally, the optimization problem is formulated.

Chapter 6 firstly introduces the software architecture implemented in this research. Then, the external software used for both the trajectory simulation and the optimization phase is discussed. Furthermore, the propagator and integrator selection strategies are investigated. In particular, the former is carried out by trading-off different propagator models for our mission scenario, whereas the latter is done based on results found in the literature. Finally, the chapter concludes with a review of verification and validation methods.

Chapter 7 presents all the results of the optimization process. In particular, a first tuning phase for the newly implemented algorithms is performed, then, both the single and multi-objective optimization process are discussed. In that phase, the best candidate solutions are selected and investigated. Besides, several Monte Carlo simulations are run for improving the best found solutions. Finally, the optimal trajectory is treated in a separate section and the key features of its orbits are investigated.

Chapter 8 presents the conclusions of our thesis study. Moreover, the recommendations for future studies are discussed.

Chapter 2

Mission Characteristics and Heritage

In this chapter, in Section 2.1, we will first introduce the solar sailing polar mission by talking about previous missions and state-of-art technology. In Section 2.2, we will then follow-up with a discussion about the mission parameters definition. Finally, in Section 2.3, we will discuss the reference vehicle, the reference mission, previous global optimization strategies applied for a solar sailing polar mission and the mission and system requirements for the mission of our interest.

2.1. Past Missions and State-of-Art Technology

The first idea that the light can exert pressure, which can then be transformed into a force to steer objects, was proposed by Maxwell in 1860. Later, Konstantin Tsiolkovsky, one of the pioneers of rocket science, discussed the concept of solar sailing and Fridrickh Tsander wrote in 1924: "For flight in interplanetary space I am working on the idea of flying, using tremendous mirrors of very thin sheets, capable of achieving favorable results."

However, the only proposal for a solar sailing mission only arrived in 1976, when NASA planned to reach the Halley comet using this technique. The sail would have been a huge square with a width of nearly 1 km, and the spacecraft would have spiraled inside the Solar System near Mercury and the Sun to acquire the proper acceleration finally rendezvous with the comet (Wie, 2007). Nevertheless, in the end, the solar electric propulsion (SEP) was preferred to solar sailing (which was discarded during phase B of the project), due to launch window and schedule problems of sailing technology (which would have not allowed to reach the comet within the right time to observe the effects of its passage near the Sun) [1].

In 2005, Planetary Society and Cosmos Studios, planned a solar sailing mission called *Cosmos 1*, whose purpose was to demonstrate this technology for the first time in history. However, the mission failed due to rocket failure [2].

Finally, in 2010, the Japanese Aerospace Exploration Agency (JAXA) launched IKAROS: the first experimental spacecraft that successfully demonstrated solar sailing technology in a journey to Venus and beyond [3].

In the same year, NASA put the first sailcraft, NANO SAIL-D2, in low Earth orbit (LEO). This mission mainly had two technical objectives:

1. to successfully stow and deploy the sail
2. to demonstrate deorbit functionality [4]

After a few years, in 2015, the Planetary Society launched Light-Sail 1, which completed a shakedown cruise around Earth. Whereas in June 2019, Light-Sail 2 was launched into

[1] http://www.planetary.org/blogs/jason-davis/2017/20170504-halleys-comet-sail-documents.html, date of access: August 2019.

[2] http://www.planetary.org/explore/projects/lightsail-solar-sailing/story-of-lightsail-part-2.html, date of access: August 2019.

[3] https://directory.eoportal.org/web/eoportal/satellite-missions/i/ikaros, date of access: August 2019.

[4] https://directory.eoportal.org/web/eoportal/satellite-missions/n/nanosail-d2, date of access: August 2019.

a much higher LEO than Light-Sail 1 (i.e., over 720 km high): this will perform the first controlled solar-sail flight around Earth [5].

Besides these missions, there are also many proposals that highlight the interest of the space community for this new kind of technology, which would potentially allow huge velocity changes with reasonable launch and development costs. Some examples are:

1. Near-Earth Asteroid Scout mission: a reconnaissance mission proposed by NASA to flyby and return data from an asteroid [6].
2. OKEANOS (Outsized Kite-craft for Exploration and Astronautics in the Outer Solar System): a mission proposed by JAXA to reach Jupiter's Trojan asteroids and collect samples (Okada et al., 2018).
3. Breakthrough Starshot: a research and engineering project by the Breakthrough Initiatives to develop a fleet of light sailcrafts to reach Alpha Centauri in 20 years[7].

All these missions and all the concept studies and research performed in the past 30 years on solar sailing have permitted to develop advanced technologies, which can withstand very harsh environments. For instance, hot thermal analysis performed by ESA for an interstellar heliopause probe has demonstrated that the aluminized temperature-resistant material CP-1 used as protective layer against the massive heating coming from the Sun, can withstand temperatures up to 240° C, which is around 513 K: only 20 degrees below the glass transition temperature of the material. This temperature, assuming a squared solar-sail configuration of 160×160 m, corresponds to a minimum distance from the Sun of nearly 0.22 AU, which is considered the current limit for solar-sail technology. Of course, with such temperatures, the payload and bus of the spacecraft should be opportunely shadowed and protected to avoid damages. Furthermore, whether the sail film can really withstand these temperatures as well as the harsh environment of the Sun (radiation and particles emitted from the Sun can seriously compromise and degrade the sail film) has still to be demonstrated with devoted qualification tests (Lyngvi et al., 2007), (Dachwald et al., 2006a).

For the specific case of a solar polar mission, we thus assume that we cannot reach the Sun at distances lower than nearly 0.22 au (in principle this distance could change depending on the surface area of the sail, however, realistically the area will hardly be so large that a sail can approach the Sun at distances lower than this).

Past Solar Sailing Polar Missions

As it was already pointed out in Chapter 1, there are no flown missions that have performed a polar orbit with an inclination of 90° around the Sun. Even though there are other missions that have performed heliocentric trajectories to study the Sun at a near distance (e.g., Helios-B launched by NASA reached a perihelion distance of 0.29 au in a nearly equatorial orbit [8]) we do not have any real data or flown missions of a solar polar mission using solar sailing. For this reason, we decided to base the entire study on available literature and past research papers concerning such a mission.

Two papers have discussed the optimization of a solar sailing polar mission with a final circular orbit of 75° inclination and 0.48 au of radius (Dachwald et al., 2006a), (Sauer, 1999). These orbits were studied starting from a heliocentric trajectory at Earth's distance, considering a first spiraling phase inwards to the Sun, and a successive phase of orbit cranking in which the inclination is varied up to 75°. Two different possibilities have been considered for the cranking of the orbit: one case in which the sail spirals up to the nearest distance that the solar film can withstand (i.e., nearly 0.22 au) and then cranks its orbit and finally spirals outwards to the desired final orbit of 0.48 au. Whereas, in the second case, the sail was directly steered to its final orbit of 0.48 au, where the cranking operation was executed. It was found that the first option resulted to be better in terms of transfer time (Dachwald et al., 2006a).

[5] http://www.planetary.org/explore/projects/lightsail-solar-sailing/, date of access: August 2019.
[6] https://www.nasa.gov/content/nea-scout/, date of access: August 2019.
[7] https://breakthroughinitiatives.org/initiative/3, date of access: August 2019.
[8] https://ntrs.nasa.gov/search.jsp?R=19760019166, date of access: August 2019.

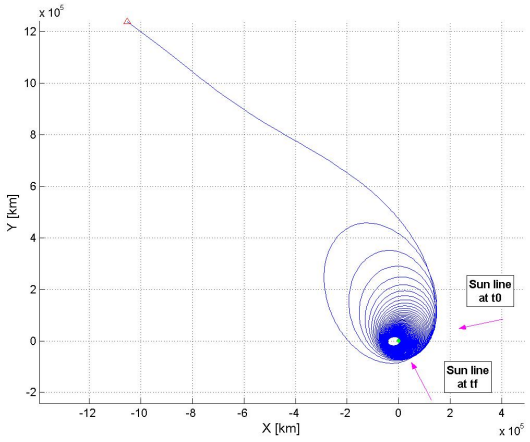
Also, other studies have been performed where a trajectory optimization was performed for a final orbit of 0.48 au and 90° of inclination, starting from a heliocentric trajectory at Earth's distance (Macdonald et al., 2006). Together with this, also technical feasibility studies have been made to corroborate that the payload and the bus required for studying the Sun at high altitudes can withstand the harsh conditions (in terms of temperatures and radiation) (Goldstein et al., 1998).

Besides these mentioned research papers, there have been studies in which the solar-sail mission was optimized starting from a Geostationary Transfer Orbit (GTO) around the Earth (Coverstone and Prussing, 2003), and finally reaching its final orbit of 0.4 au and 90° of inclination around the Sun (Garot, 2006), (Candy, 2002). In these studies, the trajectory of the sailcraft, from the GTO orbit to its final orbit, is divided into two main parts. The first one is a geocentric phase in which the spacecraft starts from a GTO orbit of zero degrees of inclination and spirals outwards until it escapes the Earth's gravitational pull (see Figure 2.1a for a graphical representation of this phase). The second phase is heliocentric, in which the main attractor of the sailcraft is the Sun: in this phase the spacecraft first spiral inwards the Sun, then is subject to a circularization of the orbit, and finally to an orbit cranking (see Figures 2.1b and 2.1c, for a graphical representation of the spiral inwards and cranking phases, respectively). In case that the orbit cranking happens to be at a nearer distance to the Sun with respect to the final orbit, a final outward spiraling is also executed to reach the final orbit. To find the best trajectory, an optimization procedure was performed for minimizing the total cost of the mission (driven by the mass of the sailcraft and the duration of the mission). Different optimization techniques have thus been implemented for this purpose. Also, it was demonstrated that the so-called perfect sail model (i.e., perfect reflectivity and no curvature of the sail) gives a 15% more optimistic value of the total travel time, and should therefore not be used (Candy, 2002). In these studies, many optimization techniques have been investigated for the solar sailing polar mission, with a particular focus on evolutionary algorithms. Also, particle swarm optimization and differential evolutions have been studied by Spaans (2009) for improving the trajectory optimization carried on a few years before by Garot (2006): it emerged that it was possible to improve the optimization, and the best results were found when a particle swarm optimization technique was used. However, all concluded that an increase in the number of individuals in the population might improve the results even further, and none of these studies have performed multi-objective optimization. Also, ant colony optimization has never been tested on this problem. These aspects will hence be studied in this research.

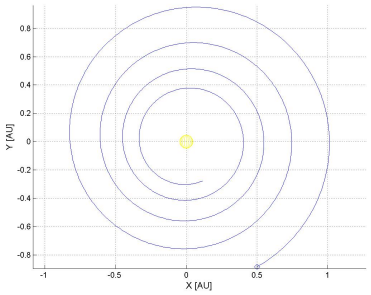
In previous research papers, some authors have also discussed different techniques for controlling the attitude of the spacecraft during the flight (which is a crucial aspect for the generation of the acceleration, and thus for the dynamics of the spacecraft) (Wie, 2004), (Wie, 2005), (Wie et al., 2005). In this thesis study, however, we will focus on neither any control strategy to be implemented for maintaining the attitude angles of the sail during his journey towards the Sun nor its actual feasibility. In fact, we will limit ourselves to discuss an ideal guidance strategy to be employed starting from certain physical considerations during the geocentric and heliocentric phases (e.g. Sun's position w.r.t. the sail, etc.). Further details on these aspects and on their limitations are given in Chapter 4.

2.2. Mission Parameters Definition

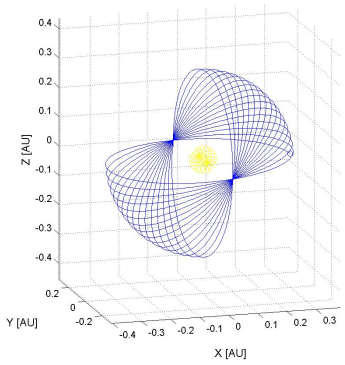
In this section, we will discuss how certain critical parameters are defined in the solar sailing polar mission and how they affect the entire trajectory planning strategy. In particular, in Section 2.2.1 we will tackle the thermal conditions to which the sail is subjected in its journey to the Sun and we will analyze how the temperature constrains the minimum distance that the sailcraft can reach w.r.t. the Sun. Moreover, in Section 2.2.2 we will investigate the initial and final conditions of the sailcraft's orbits. Also, we will talk about the initial and final time conditions to be achieved by the sailcraft.



(a) Spiral outwards of the sailcraft from Earth (Garot, 2006)



(b) Spiral inwards of the sailcraft to the Sun (Garot, 2006)



(c) Cranking of the orbit (Garot, 2006)

Figure 2.1: Escape from the Earth together with spiral inwards and orbit cranking phases around the Sun

2.2.1. Thermal Conditions

Going closer to the Sun, the sailcraft can generate a higher propulsive force thus increasing its capabilities and speed in executing orbital maneuvers. However, even if solar-sail degradation is not considered in this research, it must be noted that there is a temperature limit that the sailcraft can withstand. Above this value, the sailcraft is not able to operate anymore: thus, this temperature limit constrains the minimum solar distance achievable. The equilibrium temperature of the sail film can be computed as (Dachwald et al., 2006a):

$$T = \left[\frac{S_0}{\sigma} \frac{1 - C}{\epsilon_f + \epsilon_b} \left(\frac{r_0}{r} \right)^2 \cos \alpha \right]^{1/4} \tag{2.1}$$

where $\sigma = 5.67 \times 10^{-8}$ $\mathrm{Wm^{-2}K^{-4}}$ is the Sefan-Boltzmann constant, C is the reflectivity, $r_0 = 1$ au is the Sun-Earth distance, r is the Sun-sailcraft distance in au, ϵ_f and ϵ_b is the front and back emissivity, α is the sail cone angle and $S_0 = 1368$ W $\mathrm{m^{-2}}$ is the solar constant.

It is thus clear that having fixed the sail characteristics, the equilibrium temperature not only depends on the sailcraft distance from the Sun but also from its attitude ($T = T(r, \alpha)$). Assuming a temperature limit of the sail of $T_{lim} = 240°\mathrm{C}$, there will be a minimum distance from the Sun that cannot be exceeded by the sailcraft, regardless of the attitude angle values. This distance depends on the characteristics of the sailcraft. As it was already mentioned in Section 2.1, with the current technology for producing and making the sail films, this distance corresponds to 0.22 au. In this thesis study, we will thus assume that the sailcraft cannot go below this distance from the Sun. If this will happen in some mission scenarios,

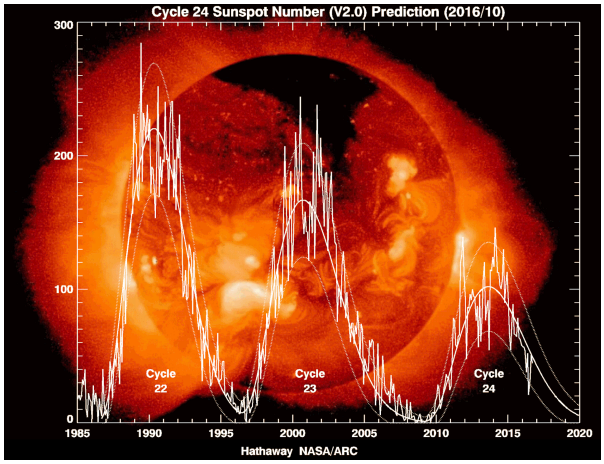


Figure 2.2: Last 3 solar cycles: cycles 22, 23, 24; credit: NASA

the related orbits will be considered unfeasible.

2.2.2. Initial and Final Time Conditions and Orbital Elements

We have already mentioned that we would like to observe the Sun during its period of strong activity: a solar maximum. We know that the Sun is subjected to a periodic activity in which solar maximums are repeated once every 11 years. Since the past solar maximum happened in 2013 (see Figure 2.2) and the next one is scheduled for 2024, the second next one will probably happen around 2035. As we would like to observe the solar cycle in its most active period, we would like to reach the final orbit before 2033 and fly around the Sun for at least 4 years, up to 2037. This means that we have an upper limit constraint for the date of arrival, which has to be a date preceding the 1^{st} of January 2033.

Nevertheless, the launch date is not known, since it depends on the duration of the mission. Therefore, the launch date is inserted as a variable and will be determined as an outcome of the optimization procedure.

Besides the launch and arrival time, also the initial and final conditions need to be discussed. We have already stated that the final orbit needs to be a circular polar orbit with 90° inclination and 0.4 au radius.

As far as concerns the initial conditions, it is established that the spacecraft is injected into a GTO orbit with a perigee height of h_p=185 km, an apogee height of h_a=35885 km and an initial inclination of i=0° w.r.t. the equatorial plane, similarly to Garot (2006). However, the argument of perigee of such orbit is assumed to be an outcome of the optimization: indeed, we would expect that the argument of perigee changes depending on the position of the Sun w.r.t. the Earth, as well as the position of the Moon w.r.t. the Earth. Since the launch date is not fixed and these positions are not known a priori, we cannot fix the argument of perigee, which will thus also be an outcome of the optimization process.

2.3. Mission Heritage and Requirements

In this paragraph, we will first discuss the reference mission and vehicle, respectively, in Sections 2.3.1 and 2.3.2. Then, in Section 2.3.3, we will talk about the optimization procedure. Finally, in Section 2.3.4, we will introduce and discuss some of the system and mission requirements that have to be fulfilled in the solar sailing polar mission of our interest.

2.3.1. Reference Mission

For having a baseline trajectory that can be used as a reference for both checking the performances of the implemented methods and compare the results, we decided to select one of the optimal trajectories found by Spaans (2009). The values of the variables found in that study are shown in Table 2.1: these result to be an outcome of the optimization process. In

Table 2.1: Reference mission variables' parameters, found by optimizing the geocentric and heliocentric phases concurrently (Spaans, 2009).

Symbol	**Values**	*Units*
T_{launch}	2014-11-15	yyyy/mm/dd
ω_0	9.98	[deg]
α_{E1}	44.5	[deg]
α_{E2}	17.6	[deg]
α_{E3}	5.6	[deg]
α_{S1}	-40.25	[deg]
α_{S2}	-17.25	[deg]
α_{S3}	37.14	[deg]
α_{S4}	54.42	[deg]
R_1	0.2970	[au]
R_2	0.2600	[au]

Table 2.2: Reference mission final orbital elements and flight duration values (Spaans, 2009). The time of arrival is in yyyy/mm/dd.

tof_{tot}	T_{arrival}	i_{f}	e_{f}	a_{f}
2064.5 JD	2020-07-10	90°	0.004	0.4 au

Table 2.2, we also show the resulting total time of flight (tof_{tot}), the arrival date ($T_{arrival}$) and the final orbital elements (e_f, i_f, a_f) when such optimal variables were used for simulating the orbit. The exact meaning of the various variables presented in this table will be clarified and explained in Chapters 3, 4 and 5.

This orbit will be used as a reference mission for our study: this will also help us to verify our entire simulation model once it will be constructed. The verification and validation phases of our entire simulator based on this reference mission are discussed in Appendix B.

2.3.2. Reference Vehicle

A reference vehicle is very useful for trading-off the same optimization strategy and dynamical model with other similar. For this reason, we have decided to choose a solar-sail vehicle such that it has the same sail parameters as previous studies. In this way, we can directly compare and discuss our results with those of previous works. Also, these parameters are considered to be realistic, as they are derived from the state-of-art sail technology. In Chapter 3, further details about the role of these coefficients in the generation of the sail force will be investigated. Now, we will only focus on these coefficients and their definition for a reference vehicle.

The objective of any solar-sail design is to furnish a wide area to generate enough thrust for the mission while keeping the structural mass as low as possible. For this reason, several solar-sail configurations have been studied throughout the years. Due to the high thrust-to-mass ratio, the fact that it has been already demonstrated in space (e.g., with the JAXA's mission IKAROS) and that it has been already analyzed for solar polar mission concepts in literature (Candy, 2002), (Garot, 2006), Spaans (2009), we decided to choose the square sail configuration.

In a square sail, the sail area (A) will be composed of four petals. These four petals are attached to four deployable booms, each of length $L_b = 70$ m, for the reference vehicle. Since each boom represents half of the diagonal of the square sail, the sail area can be computed as:

$$A = \frac{L_b^2}{2} \tag{2.2}$$

For the reference vehicle it thus results $A = 9800$ m^2.

The four booms are made of carbon fiber reinforced plastic and will constitute the support

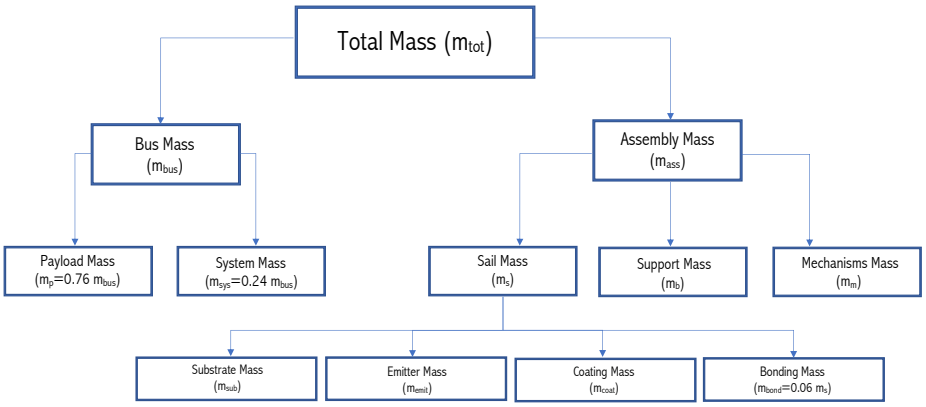


Figure 2.3: Mass components of the sailcraft.

structure, and their mass, together with the sail mass and the deployment and steering mechanisms mass, will be part of the assembly mass. The total mass of the sailcraft is thus composed of the assembly mass and the bus mass. This latter is further divided into the system mass and payload mass. The mass division of the sailcraft is shown in Figure 2.3.

It is assumed, similarly to Garot (2006), that the payload mass will constitute 24% of the bus mass, whereas the deployment and steering mechanisms will constitute 23% of the assembly mass. The sail mass can be computed as a sum of the substrate, coating, emitter and bonding mass. The bonding mass is estimated to be 6% of the sail mass, whereas the mass of the coating (i.e., aluminum), the emitter (i.e., chromium) and the substrate (i.e., kapton) can directly be derived from the thickness, density, and area of these materials.

One important performance metric of the sail is the sail loading. This parameter, together with the sail area, is typically used for retrieving the sail mass. Similarly to Garot (2006), we assume that our sail has a sail loading of:

$$\sigma = \frac{m}{A} = 20.8 \times 10^{-3} \text{ kg/m}^2 \tag{2.3}$$

It thus results that for the reference area of 9800 m^2, the sail results to have a total mass of around 204 kg. Within this total mass, the mass allotted for the payload is 5 kg.

We assume that that the attitude controller of the sailcraft together with its deployment mechanism (called 'Mechanisms Mass' in Figure 2.3) constitutes 23% of the assembly mass. Also, we assume that the sail control is such that the sailcraft is three-axis stabilized and the sail steering is assumed instantaneous.

Furthermore, it is assumed that the sail substrate is made of Kapton, with aluminum front coating and chromium back coating. These three materials allow a low thermal expansion and shrinkage: two fundamental characteristics for our mission, as we are flying the sailcraft relatively near to the Sun. Moreover, the aluminum front coating is fundamental for having an effective photon reflection.

The characteristics of the chosen substrates and back and front coating are shown in Table 2.3. In the table, t stands for the thickness of the layer, ρ for the bulk density, ϵ for the emissivity, C for the reflectivity, s for the coefficient of specular reflection and B for the non-Lambertian coefficient.

2.3.3. Solar Sailing Mission Optimization

Global optimization methods offer the possibility to optimize problems without having any insights on the specific physics behind them. This means that the same global optimization techniques can be applied to a very different range of problems (e.g., chemistry, finance, space applications, etc.).

However, compared to local optimization techniques, they often require longer computation times and less problem understanding. Nonetheless, with the now increasing compu-

Table 2.3: Relevant physical characteristics of the front coating (aluminum), substrate (kapton) and back coating (chromium) (McInnes, 1999).

Material	ρ [g/cm^3]	t [μm]	ϵ	C	B	s
Aluminum	2.70	0.1	0.05	0.88	0.79	0.94
Kapton	1.42	2	-	-	-	-
Chromium	7.14	0.015	0.64	-	0.55	-

tational power, computers are able to handle global optimization for complex problems in a way shorter time than previously.

In particular, several studies carried out by the European Space Agency (ESA), have demonstrated the potentiality of global optimization techniques for space applications: this has even led to the construction of the global trajectory optimization (GTOP) database, which is a collection of box-bounded global optimization problems on several space applications[9].

Several studies have been carried out on global optimization techniques applied to solar sailing polar missions (Dachwald et al., 2006b), (Macdonald et al., 2006), (Candy, 2002), (Garot, 2006), (Spaans, 2009). However, in these studies, a thorough benchmark between different popular global optimization techniques was not carried out. In particular, the capabilities of an ant colony optimizer in such a mission were never tested. Various research has indicated that this type of optimizer is capable of giving very promising results for trajectory optimization in space applications (Schlüter, 2014). In this study, the performance of different global optimization techniques will be tested on solar sailing trajectory optimization problems and a special focus will be devoted to a novel ant colony optimizer for both single and multi-objective optimization. This global optimization technique will be set-up and formulated for general problems and then applied to the solar sailing missions of our interest. In Chapter 5 a more thorough discussion about these methods and their characteristics is presented.

2.3.4. Mission and System Requirements

The solar sailing polar mission shall comply with several mission and system requirements. In particular, the following mission requirements are active:

MR.1: the sailcraft shall be launched in an initial GTO orbit with a perigee height of h_p=185 km, an apogee height of h_a=35885 km and an initial inclination of $i = 0°$ w.r.t. the equatorial plane.

MR.2: the final orbit shall have a 90° inclination w.r.t. the ecliptic plane.

MR.3: the final orbit shall be such that the sailcraft is in a 2:5 resonance with the Earth.

MR.4: the sailcraft shall reach the final orbit before 1 January 2033.

MR.5: the sailcraft shall reach the final orbit without neither impacting nor reaching altitude less than 1500 km w.r.t. all the Solar System planets. The only exceptions are the Moon, for which the sailcraft can reach a minimum altitude of 300 km, and the Earth, for which the minimum altitude is set to 100 km.

MR.6: the trajectory shall comply with the constraints on the cone angle $\alpha \in [-90°, 90°]$.

The system shall comply with the following requirements:

SR.1: the payload shall have a weight of 5 kg.

SR.2: the area of the sailcraft shall be comprised of 6000 m^2 and 10000 m^2.

[9] `https://www.esa.int/gsp/ACT/projects/gtop/gtop.html`, date of access: August 2019.

SR.3: the reflecting area of the sailcraft shall be made of an aluminum front coating, a kapton substrate, and a chromium back coating.

SR.4: the sailcraft shall not exceed the temperature limit of 513 K. Therefore, the sailcraft cannot approach the Sun at distances lower than 0.22 au.

Chapter 3

Flight Dynamics

In this chapter, we will discuss the dynamics involved in a solar sailing polar mission. For doing this, it is first important to discuss how the motion of the sail is described (i.e., which coordinate systems are involved) and what are the reference systems involved in the simulation. Hence, we will first include the reference frames definitions in Section 3.1 and the state variables discussion, in Section 3.2. Singularities, computational speed, and other factors might make one set of state variables advantageous w.r.t. another. It is thus crucial to discuss the different possible sets of state variables and their drawbacks. Finally, in Sections 3.3 and 3.4, an investigation of the equations of motion and the environment properties will be carried out.

3.1. Reference Frames

The purpose of this section is not only to introduce and describe the various reference systems used in both the heliocentric and geocentric phases but also to explain their mathematical relationships and how it is possible to pass from one reference system to another, taking advantage of coordinate transformations. For doing this, we will introduce several angles that are strictly related to the orientation of the solar-sail thrust vector. Indeed, depending on how the sail is inclined w.r.t. the Sun, a different direction and magnitude of the solar pressure force will result. Since the orientation of the solar-sail is often described through its cone and clock angles, these will also be introduced in this section. Typically, due to the high influence of these angles on the solar-sail trajectory, they are often optimized in the search of optimal mission scenarios (Wie, 2008).

Since our simulation model consists of two different phases with two different central bodies (i.e., the Earth first and then the Sun), it is crucial to define both the geocentric and heliocentric pseudo-inertial[1] reference systems, as well as their transformations.

The inertial geocentric reference system can be constructed as follows: the x-axis (unit vector: $\hat{\mathbf{I}}_{GEO}$) points towards the vernal equinox, the z-axis (unit vector: $\hat{\mathbf{K}}_{GEO}$) passes through Earth's pole, and the y-axis is perpendicular to both (unit vector: $\hat{\mathbf{J}}_{GEO}$, where $\hat{\mathbf{J}}_{GEO} = \hat{\mathbf{K}}_{GEO} \times \hat{\mathbf{I}}_{GEO}$). We know that the Earth's inertial reference system ($\hat{\mathbf{I}}_{GEO}$, $\hat{\mathbf{J}}_{GEO}$, $\hat{\mathbf{K}}_{GEO}$), is inclined with an angle ϵ (i.e., the obliquity: ϵ=23.43689°) with respect to the heliocentric inertial reference system. The transformation matrix from the geocentric to the heliocentric frame can thus be written as:

$$\boldsymbol{C}_{H\leftarrow G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\epsilon) & \sin(-\epsilon) \\ 0 & -\sin(-\epsilon) & \cos(-\epsilon) \end{bmatrix} \tag{3.1}$$

The rotation between the two reference frames is performed when the sailcraft exceeds the distance defined by the Earth's sphere of influence. This distance can be computed as (Wakker, 2015):

$$r_{s.i.,E} = r_{E-S}\left(\frac{m_E}{m_S}\right)^{2/5} \tag{3.2}$$

[1] In practice we cannot use the word 'inertial reference frames' for neither the geocentric nor the heliocentric reference frames and we are always forced to work with so-called pseudo-inertial reference frames: however, since the Coriolis and centrifugal forces caused by these bodies on the sailcraft's dynamics are negligible, we will always refer to the pseudo-inertial geocentric and heliocentric reference frames as inertial reference frames.

where $r_{E-S} = 149.6 \times 10^6$ km is the Sun-Earth distance, $m_E = 5.97 \times 10^{24}$ kg is the mass of the Earth, $m_S = 1,988,500 \times 10^{24}$ kg is the mass of the Sun[2] [3]. The resulting Earth's sphere of influence has a value of $r_{s.i.,E} = 0.9245 \times 10^6$ km.

Besides the position vector, it is also fundamental to transform the velocity vector from the geocentric to the heliocentric frame, when the sailcraft passes such a distance. By defining as $\boldsymbol{\omega}$ the angular rotation vector of the Earth w.r.t. the Sun, $\mathbf{r}_E$ the position vector of the Earth w.r.t. the Sun and as $\mathbf{v}|_G$, $\mathbf{v}|_S$ the velocity vector of the sailcraft w.r.t. the geocentric and heliocentric, respectively, inertial frames, we can write:

$$\mathbf{v}|_S = \boldsymbol{C}_{H \leftarrow G}\mathbf{v}|_G + \boldsymbol{\omega}_E \times \boldsymbol{r}_E \tag{3.3}$$

Besides the inertial Earth reference system, it is also useful to introduce the orbit frame ($\hat{\mathbf{e}}_r$, $\hat{\mathbf{e}}_t$, $\hat{\mathbf{e}}_h$), in which an axis is directed radially (i.e., $\hat{\mathbf{e}}_r$), another one in the direction of the normal to the orbital plane around the central body (i.e., $\hat{\mathbf{e}}_h$), and the the third axis is perpendicular to these two (i.e., $\hat{\mathbf{e}}_t$). Hence this reference system can be found as: $\hat{\mathbf{e}}_r = \dfrac{\mathbf{r}}{|\mathbf{r}|}$, $\hat{\mathbf{e}}_h = \dfrac{\mathbf{r} \times \mathbf{v}}{|\mathbf{r} \times \mathbf{v}|}$, and $\hat{\mathbf{e}}_t = \hat{\mathbf{e}}_h \times \hat{\mathbf{e}}_r$. This reference system can be particularly useful because some forces and steering angles for particular orbits are defined relative to the orbit orientation.

Concerning the heliocentric phase, we first of all define as ($\hat{\mathbf{I}}$,$\hat{\mathbf{J}}$,$\hat{\mathbf{K}}$) the right-handed orthonormal vectors of the heliocentric ecliptic rectangular reference system: $\hat{\mathbf{I}}$ is directed towards the vernal equinox; $\hat{\mathbf{K}}$ is perpendicular to the ecliptic; and $\hat{\mathbf{J}}$ can then be found using the cross product: $\hat{\mathbf{J}} = \hat{\mathbf{K}} \times \hat{\mathbf{I}}$. Furthermore, we indicate with ($\hat{\mathbf{r}}$,$\hat{\boldsymbol{\psi}}$,$\hat{\boldsymbol{\phi}}$) the spherical coordinates reference frame (see Figure 3.1 for a graphical interpretation).

We can thus write the position vector of the spacecraft with respect to the Sun (by calling r the Sun-sailcraft distance) as:

$$\mathbf{r} = r\hat{\mathbf{r}} = (r\cos\phi\cos\psi)\hat{\mathbf{I}} + (r\cos\phi\sin\psi)\hat{\mathbf{J}} + (r\sin\phi)\hat{\mathbf{K}} = X\hat{\mathbf{I}} + Y\hat{\mathbf{J}} + Z\hat{\mathbf{K}} \tag{3.4}$$

where $\psi \in [0°,360°]$ and $\phi \in [-90°,90°]$ are the ecliptic longitude and latitude, respectively, of the sailcraft.

We can express the unit vector that indicates the normal direction to the sail plane as:

$$\hat{\mathbf{n}} = (\cos\alpha)\hat{\mathbf{r}} + (\sin\alpha\sin\beta)\hat{\boldsymbol{\psi}} + (\sin\alpha\cos\beta)\hat{\boldsymbol{\phi}} \tag{3.5}$$

where α is the cone angle and β is the clock angle (defined as shown in Figure 3.2).

The unit vector $\hat{\mathbf{n}}$ is one of the three vectors that define the so-called sail frame: ($\hat{\mathbf{n}}$, $\hat{\mathbf{t}}$, $\hat{\mathbf{p}}$): $\hat{\mathbf{n}}$ points normal to the sail surface and away from it, from the back side; $\hat{\mathbf{t}}$ lays in the plane

[2] https://nssdc.gsfc.nasa.gov/planetary/factsheet/ves.html, date of access: August 2019.
[3] https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html, date of access: August 2019.

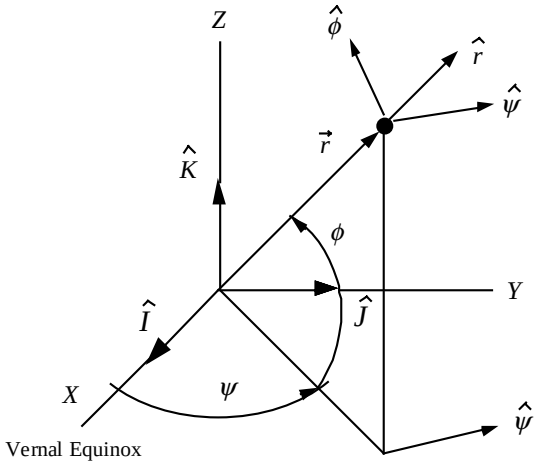


Figure 3.1: Heliocentric ecliptic coordinates (X, Y, Z) and spherical coordinates (r, ψ, ϕ) (Wie, 2008)

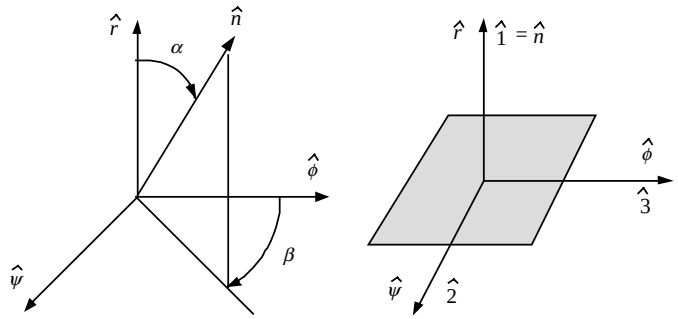


Figure 3.2: Cone (α) and clock (β) angles (Wie, 2008)

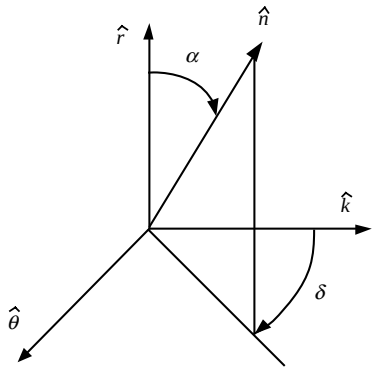


Figure 3.3: Cone (α) and clock (δ) angles (Wie, 2008)

identified by the normal to the surface $\hat{\mathbf{n}}$ and the Sun-sail line, which corresponds to $\hat{\mathbf{r}}$ in the heliocentric frame, and is perpendicular to $\hat{\mathbf{n}}$; whereas $\hat{\mathbf{p}}$ completes the reference system: $\hat{\mathbf{p}} = \hat{\mathbf{n}} \times \hat{\mathbf{t}}$.

We define the two angles α and β as:

$$\cos\alpha = \hat{\mathbf{r}} \cdot \hat{\mathbf{n}} \qquad\qquad \cos\beta = \frac{\hat{\mathbf{r}} \times (\hat{\mathbf{n}} \times \hat{\mathbf{r}})}{|\hat{\mathbf{r}} \times (\hat{\mathbf{n}} \times \hat{\mathbf{r}})|} \cdot \hat{\boldsymbol{\phi}}$$

where $\alpha \in [-90°,90°]$ and $\beta \in [0°,360°]$.

From these angles, it is easy to relate the sail reference system $(\hat{\mathbf{n}}, \hat{\mathbf{t}}, \hat{\mathbf{p}})$, with the spherical coordinate system. Similarly, we can also relate the sail reference system to an osculating reference system (i.e., $(\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}}, \hat{\mathbf{k}})$), which describes the orientation of the spacecraft w.r.t. a reference system that describes the osculating orbital plane of the sailcraft. The osculating orbital elements, which represent the Keplerian elements (i.e., a: semi-major axis, e: eccentricity, i: inclination, Ω: right ascension of ascending node, ω: argument of perigee, ν: true anomaly) of the sailcraft in space at any given time, can then be used for describing the sailcraft state.

This reference system is composed by a radial unit vector $\hat{\mathbf{r}}$ (which coincides to the one of the spherical reference system), a unit vector always perpendicular to the local Keplerian orbit of the sailcraft (i.e., $\hat{\mathbf{k}}$) and a third unit vector perpendicular to the aforementioned two. In such a reference system, we can introduce a new set of cone and clock angles: (α, δ) (defined as shown in Figure 3.3), where it is clear that α is the same as before, but the clock angle δ is now different than the previous one (β).

We can thus relate the sailcraft reference frame with the osculating plane reference system as follows:

$$\begin{pmatrix} \hat{\mathbf{n}} \\ \hat{\mathbf{t}} \\ \hat{\mathbf{p}} \end{pmatrix} = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{bmatrix} \begin{pmatrix} \hat{\mathbf{r}} \\ \hat{\boldsymbol{\theta}} \\ \hat{\mathbf{k}} \end{pmatrix} \tag{3.6}$$

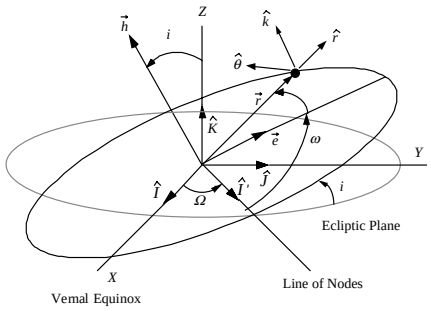


Figure 3.4: Orbital geometry (Wie, 2008)

Where the two angles $\alpha \in [-90°,90°]$ and $\delta \in [0°, 360°]$ are used to describe the orientation of the spacecraft w.r.t. the normal to the sail plane $\hat{\mathbf{n}}$ as follows:

$$\hat{\mathbf{n}} = (\cos\alpha)\hat{\mathbf{r}} + (\sin\alpha\sin\delta)\hat{\boldsymbol{\theta}} + (\sin\alpha\cos\delta)\hat{\mathbf{k}} \tag{3.7}$$

where:

$$\cos\alpha = \hat{\mathbf{r}}\cdot\hat{\mathbf{n}} \qquad\qquad \cos\delta = \frac{\hat{\mathbf{r}}\times(\hat{\mathbf{n}}\times\hat{\mathbf{r}})}{|\hat{\mathbf{r}}\times(\hat{\mathbf{n}}\times\hat{\mathbf{r}})|}\cdot\hat{\mathbf{k}}$$

Finally, for expressing the sailcraft reference frame starting with the heliocentric inertial reference system, we just need to link the osculating reference system with the heliocentric inertial one. This can be done remembering the geometry of the Keplerian elements with respect to the rectangular heliocentric ecliptic system (shown in Figure 3.4) and thus writing:

$$\begin{pmatrix}\hat{\mathbf{r}}\\ \hat{\boldsymbol{\theta}}\\ \hat{\mathbf{k}}\end{pmatrix} = \begin{bmatrix}\cos(\omega+\nu) & \sin(\omega+\nu) & 0\\ -\sin(\omega+\nu) & \cos(\omega+\nu) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & \cos i & \sin i\\ 0 & -\sin i & \cos i\end{bmatrix}\begin{bmatrix}\cos\Omega & \sin\Omega & 0\\ -\sin\Omega & \cos\Omega & 0\\ 0 & 0 & 1\end{bmatrix}\begin{pmatrix}\hat{\mathbf{I}}\\ \hat{\mathbf{J}}\\ \hat{\mathbf{K}}\end{pmatrix} \tag{3.8}$$

Therefore, from the above discussion, we have not only managed to mathematically define and relate all the reference systems involved in the solar-sail problem, but we have also defined and introduced two pivotal angles: the cone angle (α) and the clock angle (δ). These two angles will be crucial for the entire mission, as they will be used for expressing the orientation (i.e., attitude) of the sailcraft in its journey towards the Sun.

3.2. State Variables

For setting up the equations of motion of the sailcraft, it is first fundamental to specify the state variables used to express the vehicle's state. Since the choice of the state variables influences both the performance of the simulation model, thus influencing the total computational effort needed to optimize the mission, and also the accuracy of the results, it is important to understand which set of state variables can be best suitable for our case. In this framework, we investigate three types of state variables often used for space applications: Cartesian components, modified equinoctial elements, and unified state model elements. Moreover, Keplerian elements will also be discussed as they are useful to introduce the modified equinoctial elements. On the one hand, the Kepler elements, Cartesian elements, and modified equinoctial elements represent the easiest to interpret physically and sometimes the most robust formulation. On the other hand, the unified state model has demonstrated to be particularly effective when used for low-thrust propulsion missions (such as the one that we are investigating). Therefore, a thorough discussion and trade-off of these different state variables are necessary and it is presented in Section 6.3.1. We will now limit ourselves to describe their working principles and key features. In particular, we will start by describing the Cartesian elements, for then moving to the modified equinoctial elements (introduced

by an initial necessary description of the Keplerian elements) and finally to the three types of unified state models: with quaternions, with exponential mapping, and with Rodrigues parameters.

Cartesian Elements

Cartesian elements are used to express position and velocity w.r.t. an inertial or rotating orthogonal reference frame. The position (i.e., $\mathbf{r} = (x, y, z)^T$) and velocity (i.e., $\mathbf{v} = (v_x, v_y, v_z)^T$) vectors are thus used to constitute the following state vector:

$$\mathbf{x} = \left(\mathbf{r}^T,\ \mathbf{v}^T\right)^T \tag{3.9}$$

Kepler Elements

Keplerian elements describe the state of the sailcraft through some parameters that define the position of the sailcraft on a time-varying ellipse. These parameters that describe the shape and orientation of such ellipse and the position of the sailcraft in the ellipse are: the eccentricity (e), the semi-major axis (a), the inclination (i), the longitude of the ascending node (Ω), the argument of periapsis (ω) and the true anomaly (ν).

These six parameters can be seen in Figure 3.4 and they constitute the following state vector:

$$\mathbf{x} = (e,\ a,\ i,\ \Omega,\ \omega,\ \nu)^T \tag{3.10}$$

Modified Equinoctial Elements

As we already pointed out, MEE is a modified version of the Kepler elements. This was made to avoid mathematical singularities, which limited the use of the Kepler elements. Their first appearance was in Walker (1986), and the relation between MEE and Kepler elements $(a, e, i, \Omega, \omega, \nu)$ can be expressed as follows:

$$\begin{aligned} p &= a(1 - e^2) & f &= e\cos(\omega + \Omega) \\ g &= e\sin(\omega + \Omega) & h &= \tan\frac{i}{2}\cos\Omega \\ k &= \tan\frac{i}{2}\sin\Omega & L &= \Omega + \omega + \nu \end{aligned} \tag{3.11}$$

Therefore, the state vector can be written as:

$$\mathbf{x} = (p,\ f,\ g,\ h,\ k,\ L)^T \tag{3.12}$$

Unified State Model Elements

In this paragraph, we will first describe the original unified state model using quaternions. Then we will discuss two modifications done by replacing the quaternion with either the Rodrigues parameters or the exponential mapping. Unified state model elements have been extensively used in the past years, especially for orbits with continuous low-thrust propulsion, thanks to their demonstrated good performances in terms of accuracy and CPU time.

We know that reference frames are typically used for expressing position and motion of bodies. Usually, a reference system is chosen as a set of three right-handed mutually perpendicular unit vectors $(\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{a}}_3)$. For converting a set of coordinates from a certain reference frame to another, various possible ways exist. These include, among others, direction cosine matrix, Euler axis ($\hat{\mathbf{a}}$) and angle (Φ), quaternions, modified Rodrigues parameters (MRP) and exponential mapping. In particular, for introducing the USM state variables, we are mostly interested in Euler axis and angle, quaternions, MRP and exponential mapping.

Quaternions of unit magnitude can be used for representing rotations. In particular, they can be seen as a vector of four elements constituted by:

- one vector of three elements, $\boldsymbol{\epsilon}$, which can be expressed as a function of Euler axis and angle as:

$$\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \epsilon_3)^T = \hat{\mathbf{a}}\sin(\Phi/2) \tag{3.13}$$

- one scalar, η, which can also be expressed as a function of Euler angle as:

$$\eta = \cos(\Phi/2) \tag{3.14}$$

A modified Rodrigues parameter (MRP) vector can be expressed w.r.t. Euler axis and angle as follows:

$$\boldsymbol{\sigma} = \hat{\mathbf{a}} \tan(\Phi/4) \tag{3.15}$$

Another type of representation exists besides MRP and quaternions for USM: exponential mapping. This three parameters set is also defined as exponential mapping of quaternions (Grassia, 1998). The exponential map (**a**) is defined as:

$$\mathbf{a} = \Phi\hat{\mathbf{a}} \tag{3.16}$$

These three representations are used for expressing the orientation of the orbit in the unified state model. In particular, four elements are used for the quaternions and three for the Rodrigues and exponential mapping parameters, as we just saw. On the other hand, the shape of the orbit is described by three parameters, in all these representations. For introducing these three parameters, it is first important to introduce the concept of hodograph: a hodograph is a velocity diagram used for representing graphically the velocity of a body. For an unperturbed orbit, the velocity can be described, at any point, as the sum of a velocity vector normal to the radial vector and laying in the orbital plane (**C**) and a velocity vector (**R**), which is shifted 90° ahead w.r.t. the eccentricity vector. The magnitudes of these two velocity vectors (namely, **C** and **R**) can be written as:

$$C = \frac{\mu}{h} \qquad\qquad R = \frac{\mu e}{h} = Ce$$

where e and h are the eccentricity and the specific angular momentum of the orbit, respectively, whereas μ is the gravitational parameter of the central body. We can link these components of the velocity with the polar components (i.e., one component of the velocity along the radial (v_r), and the other perpendicular to it (v_ν) and laying in the orbital plane forming a right hand orthogonal reference system together with the radial direction and the perpendicular to the orbit) as:

$$v_r^2 + (v_\nu - C)^2 = R^2 \tag{3.17}$$

Now, to introduce the three elements used for describing the shape of the orbits in the USM, it is first important to introduce three different frames:

- $\mathcal{F}_g$=($\hat{\mathbf{g}}_1$, $\hat{\mathbf{g}}_2$, $\hat{\mathbf{g}}_3$): an inertial reference frame fixed on the central body, where $\hat{\mathbf{g}}_1$ and $\hat{\mathbf{g}}_2$ lay in the equatorial plane of the central body and the third unit vector is perpendicular to that plane.
- $\mathcal{F}_f$=($\hat{\mathbf{f}}_1$, $\hat{\mathbf{f}}_2$, $\hat{\mathbf{f}}_3$): an intermediate rotating reference frame, where $\hat{\mathbf{f}}_1$ and $\hat{\mathbf{f}}_2$ lay in the orbital plane and the unit vector $\hat{\mathbf{f}}_3$ lays along the angular momentum vector.
- $\mathcal{F}_e$=($\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$, $\hat{\mathbf{e}}_3$): a rotating reference frame, where $\hat{\mathbf{e}}_1$ lays in the radial direction, $\hat{\mathbf{e}}_3$ lays along the angular momentum vector and $\hat{\mathbf{e}}_2$ is perpendicular to those two.

The angle that separates the unit vector $\hat{\mathbf{f}}_1$ to $\hat{\mathbf{e}}_1$ is called λ and is computed as the sum of the perigee anomaly (ω), longitude of the ascending node (Ω) and true anomaly (ν).

We now have all the elements to define the USM state vector. As far as concerns the quaternion representation, the following state vector is derived:

$$\mathbf{x} = \left(C,\ R_{f1},\ R_{f2},\ \boldsymbol{\epsilon}^T,\ \eta\right)^T \tag{3.18}$$

where C is the magnitude of the vector $\mathbf{C}$, R_{f1} and R_{f2} are the two components of the vector $\mathbf{R}$ when written w.r.t. the $\mathcal{F}_f$ frame and $\boldsymbol{\epsilon}$ and η are the components of a quaternion that describes the orientation of $\mathcal{F}_e$ w.r.t. $\mathcal{F}_g$.

On the other hand, using the MRP representation, the following state vector can be derived:

$$\mathbf{x} = \left(C,\ R_{f1},\ R_{f2},\ \boldsymbol{\sigma}^T\right)^T \tag{3.19}$$

Finally, using the exponential mapping representation, we obtain the following state vector:

$$\mathbf{x} = \left(C,\ R_{f1},\ R_{f2},\ \mathbf{a}^T\right)^T \tag{3.20}$$

3.3. Equations of Motion

In this section, we will describe the equations of motion relevant for our solar-sail problem: these constitute the mathematical laws that drive the trajectory of the sailcraft instant by instant. It is important to notice that each set of state variables will cause a different set of equations of motions. As this is just a different mathematical formulation of the same symbolic equations, the various different formulations of these equations of motion are addressed in Appendix A. All these different formulations have the same objectives to describe the dynamics of the sailcraft both in the geocentric and heliocentric phase. In particular, in both cases, the equation of motion can be written, in vectorial form, as:

$$\ddot{\mathbf{r}}|_{G/H} + \mathbf{F}_{grav}\Big|_{G/H} = \left(\frac{\mathbf{F}_{SRP}}{m} + \frac{\mathbf{G}}{m}\right)\Big|_{G/H} \tag{3.21}$$

where $\mathbf{F}_{grav}$ is the gravitational force of the central body, $\mathbf{F}_{SRP}$ is the force generated from the solar radiation pressure, and $\mathbf{G}|_{G/H}$ are the perturbations present in the geocentric or heliocentric phase, depending on where the sailcraft is located. The subscripts G and H thus refer to either the geocentric or heliocentric phase, depending on which is the central gravitational body.

3.4. Environment

This section deals with the environment description, that is the general framework of all the natural interacting forces with the sailcraft, during its journey to the Sun. The sailcraft will be subjected to the gravitational pull of several celestial bodies: some will act as central attractors, others will only result as third body perturbations. Furthermore, during the geocentric phase, the sailcraft will also be subjected to the Earth's atmosphere. Moreover, one of the main forces acting on the sailcraft will be the solar pressure radiation force, which results to be the only thrusting force.

To summarize, the main forces that constitute the environment in such a mission are:

- Main gravitational forces: these include only the celestial bodies that act as a main central gravitational attractor. In this mission, the Earth and the Sun will act as main attractors in the geocentric and heliocentric phases, respectively.

- Solar radiation pressure force: due to the transfer of momentum between the photons emitted by the Sun and the sailcraft's area a force arises. For this mission, this force is used as propelling and thrusting force and thus plays a pivotal role in defining the trajectory.

- Solar eclipse: due to the high influence of the solar radiation pressure on the sailcraft motion, it is important to model solar eclipses. Indeed, the sailcraft might result to be shadowed by the Earth, especially during the geocentric phase (as it revolves around the Earth multiple times). This effect can be particularly relevant in shaping the trajectory of the sail, and it should thus be included in the mission scenario.

- Atmospheric forces: these are only active in the geocentric phase when the sailcraft is near to the Earth's surface and include both the lift and the drag. These forces should be accounted for in our mission scenario, as the sailcraft has a very big area, which is thus subjected to very strong aerodynamic accelerations if not properly controlled.

- Third bodies gravitational perturbations: several celestial bodies might exert on the sailcraft small gravitational perturbations that can add up over time and cause significant changes in the orbit. For this reason, it is important to model and discuss these perturbing forces and their mathematical descriptions.

All the gravitational forces caused by the central bodies are modeled considering central bodies as point masses. In this mission, there will be two central bodies: the Earth in the geocentric phase and the Sun in the heliocentric phase. The gravitational forces generated by these two bodies can be written as:

$$\mathbf{F}_{grav} = \frac{\mu}{r^3}\mathbf{r} \tag{3.22}$$

where $\mathbf{r}$ is the vector that identifies the sailcraft's position w.r.t. the center of mass of the central body, whereas μ is the gravitational parameter of the central body. For the Sun, it holds: μ_s=1.32712440018 × 10^{11} km^3/s^2; whereas for the Earth it holds: μ_e=3.986004418 × 10^5 km^3/s^2.

3.4.1. Solar Radiation Pressure

In this section, we will tackle the solar radiation pressure phenomenon. In particular, in Section 3.4.1.1, we will first discuss the generation of this pressure. Then, in Section 3.4.1.2, we will describe how this pressure translates into a vector force that can be used for steering the sailcraft. The main references for this section are McInnes (1999), Wakker (2015) and Fu et al. (2016).

3.4.1.1 Fundamentals of Solar Sailing

One of the most important aspects of this mission is how the solar-sail force is produced from the solar radiation pressure acting on the sail.

When solar photons hit an object, the momentum is transferred and pressure is felt by the object: called solar radiation pressure. This pressure can be expressed as:

$$P = \frac{W}{c} \tag{3.23}$$

Where W ([W/m^2]) is the radiative flux (i.e., power density), which is the power per meter squared radiated and c is the speed of light (c=299,792,458 m/s).

The effectiveness of solar-sail propulsion is strictly related to the distance of the sail from the Sun: namely, the more the sail is distant the less propulsive force can be generated. This can be mathematically seen from the Sun's radiative power definition:

$$W(r) = \frac{L_s}{4\pi r^2} \tag{3.24}$$

Where r is the distance of the object radiated from the Sun, and $L_s = 3.84 \times 10^{26}$ W is the luminosity of the Sun.

In general, the maximum solar radiation pressure experienced by an object at a distance r from the Sun is determined not only by the distance but also on how the photons are reflected: namely, on the reflective properties and orientation of the object. Assuming a condition in which the photons are all reflected back from the object to the Sun, we can compute the maximum solar radiation pressure, $P_{max}(r)$, as $P_{max}(r) = 2P(r)$. Conversely, for a partially reflective surface, the second term must be multiplied by a radiation pressure coefficient of the body, that is smaller than two.

3.4.1.2 Solar Radiation Pressure Force Vector

In this section, we discuss two different models for treating the solar radiation pressure force vector: the first one assuming an ideal model in which the solar-sail only reflects all the incident radiation and the force vector generated from the reflection of photons is always perpendicular to the sail surface. The second one is a more realistic model in which the reflection is imperfect and the vector is not directed perpendicularly to the surface but is shifted of a certain angle ϕ (called offset angle). In the following description we will base our notation on the reference systems and state variables defined under Sections 3.1 and 3.2.

Developing an appropriate model can be very complex and challenging: in general, when a solar photon hits the sail, we will assume that it will be either transmitted with a probability τ or absorbed with a probability $\overline{\alpha}$ or reflected with a probability ρ, where it holds: $\tau+\rho+\overline{\alpha}=1$. The total radiation pressure force is found by summing the forces generated by these three phenomena. We will consider a sail area A, and we will indicate with $\hat{\mathbf{n}}_s$ the unit vector that individuates the direction of the incident solar radiation, which forms an angle α (i.e., the cone angle defined in Section 3.1) with the unit vector normal to the surface $\hat{\mathbf{n}}$. Assuming that we are operating in a Sun-centered frame, then the angle between the solar incident radiation and the normal to the plane is α.

Therefore, the total force can be written as:

$$\mathbf{F}_{SRP} = \tau\,\mathbf{F}_{\tau} + \overline{\alpha}\,\mathbf{F}_{\overline{\alpha}} + \rho\,\mathbf{F}_{\rho} \tag{3.25}$$

Moreover, after considering that transmitted photons do not generate any force on the surface (i.e., $\mathbf{F}_{\tau}=\mathbf{0}$), we can express each of the other two force contributions as follows:

$$\begin{cases} \overline{\alpha}\,\mathbf{F}_{\overline{\alpha}} & = \overline{\alpha}\,\dfrac{WA\cos\alpha}{c}\left(\hat{\mathbf{n}}_s - \dfrac{\epsilon_b B_b - \epsilon_f B_f}{\epsilon_f+\epsilon_b}\hat{\mathbf{n}}\right) \\ \rho\,\mathbf{F}_{\rho} & = \rho_S\,\dfrac{2WA}{c}\cos^2\alpha + \hat{\mathbf{n}} + \rho_d\,\dfrac{W\cos\alpha}{c}(\hat{\mathbf{n}}_s + B_f\hat{\mathbf{n}}) \end{cases} \tag{3.26}$$

where ϵ is the surface emissivity, σ is the Stefan-Boltzmann constant (σ=5.670367 × 10^{-8} J m^{-2} s^{-1} K^{-4}), B is the Lambertian coefficient and the subscripts f and b indicate front and back side of the surface, respectively. Furthermore, $\hat{\mathbf{n}}$ is the unit vector that points away from the sail surface from the bottom side (where we define the front side as the one that is directly exposed to the Sun's rays, whereas the back side is the opposite one). Besides, the force associated with the reflected radiation is composed of two contributions: a specularly reflected and diffusely reflected radiation component. Hence, ρ_s is the probability associated with the specularly reflected radiation, whereas ρ_d is the one associated with the diffusely reflected radiation.

In case that we consider that the sailcraft is subjected to specular reflection only (which means that ρ_s=1 and hence all the other coefficients are zero) the equations simplify and the force becomes the ideal solar-sail force:

$$\mathbf{F}_{SRP,id} = 2PA\cos^2\alpha\hat{\mathbf{n}} \tag{3.27}$$

Instead, the so called non-ideal model considers all the photon-sail interactions, which we have described above, and it assumes that the sail area A is flat. By putting together all the aforesaid equations, it is possible to derive an equation to express the total force as a result of the interaction between the photons and the sail as:

$$\mathbf{F}_{SRP} = \frac{WA}{c}\left(\overline{\alpha}\cos\alpha\left(\hat{\mathbf{n}}_s - \frac{\epsilon_b B_b - \epsilon_f B_f}{\epsilon_f+\epsilon_b}\hat{\mathbf{n}}\right) + 2\rho_s\cos^2\alpha\hat{\mathbf{n}} + \rho_d\cos\alpha(\hat{\mathbf{n}}_s + B_f\hat{\mathbf{n}})\right) \tag{3.28}$$

We notice that the force generated from the solar radiation pressure is not only perpendicular directed along $\hat{\mathbf{n}}$ (as it is the case for the ideal force model), but it also has a transverse component ($\hat{\mathbf{n}}_t$) in the direction perpendicular to the normal of the surface and laying on the sail surface. Since it holds that $\hat{\mathbf{n}}_t\cdot\hat{\mathbf{n}}_s \geq 0$, $\hat{\mathbf{n}}\cdot\hat{\mathbf{n}}_s = \cos\alpha$ and $\hat{\mathbf{n}}_s\cdot\hat{\mathbf{n}}_t = \cos(\pi/2-\alpha) = \sin\alpha$, we can write the force over the sail area A in terms of tangential and normal components as:

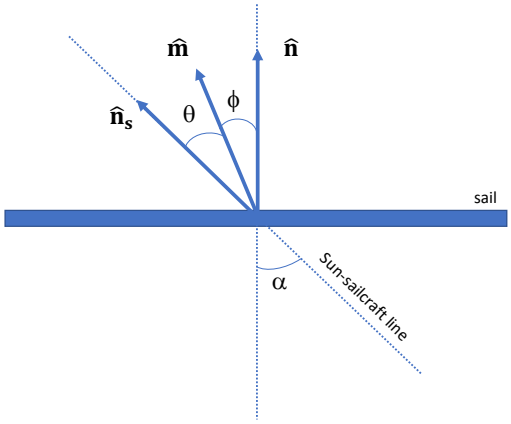


Figure 3.5: Direction of ideal (along $\hat{\mathbf{n}}$) and non-ideal (along $\hat{\mathbf{m}}$) forces (where $\hat{\mathbf{n}}_s$ is the unit vector that relates the Sun to the sailcraft)

$$\mathbf{F}_{SRP} = PA\cos\alpha\left((\overline{\alpha}+\rho_d)(\hat{\mathbf{n}}_s\cdot\hat{\mathbf{n}}_t)\hat{\mathbf{n}}_t + \left((\overline{\alpha}+\rho_d+2\rho_s)\cos\alpha - \overline{\alpha}\frac{\epsilon_b B_b - \epsilon_f B_f}{\epsilon_f+\epsilon_b} + \rho_d B_f\right)\hat{\mathbf{n}}\right) \tag{3.29}$$

Alternatively, the force components along $\hat{\mathbf{n}}$ and $\hat{\mathbf{n}}_t$ can be written as (McInnes, 1999):

$$\mathbf{F}_{SRP,n} = PA\left\{(1+C_r s)\cos^2\alpha + B_f(1-s)C_r\cos\alpha + (1-C_r)\frac{\epsilon_f B_f - \epsilon_b B_b}{\epsilon_f+\epsilon_b}\cos\alpha\right\}\hat{\mathbf{n}} \tag{3.30}$$

$$\mathbf{F}_{SRP,t} = PA(1-C_r s)\cos\alpha\sin\alpha\,\hat{\mathbf{n}}_t \tag{3.31}$$

This form of expressing the force is exactly equivalent as the one presented above, but it makes use of different parameters: C_r is the reflectivity coefficient of the front of the sail, s is the coefficient of specular reflection, and the other parameters are the same as the ones already explained above. In particular, the following relationships hold:

$$\alpha = (1-C_r) \qquad \rho_d = C_r(1-s) \qquad \rho_s = C_r s \tag{3.32}$$

where it thus also holds $(\alpha+\rho_d) = 1-C_r s$, $(\alpha+\rho_d+\rho_s) = 1$ and $(\alpha+\rho_d+2\rho_s) = (1+C_r s)$.

From Equations (3.27) and (3.29), we can observe that while the ideal force is directed in the same direction as the normal to the surface (pointing away from the back surface of the sail), the non-ideal one is not. In fact, the non-ideal force is inclined by the offset angle ϕ w.r.t. the normal direction, where ϕ can be defined as:

$$\tan\phi = \frac{|\mathbf{F}_{SRP,t}|}{|\mathbf{F}_{SRP,n}|} \tag{3.33}$$

Furthermore, defining as θ the angle between the non-ideal force direction and the Sun-sailcraft line, it is possible to relate the cone and offset angles as follows:

$$\phi + \theta = \alpha \tag{3.34}$$

These angles are graphically shown in Figure 3.5. The ideal force will, therefore, be parallel to $\hat{\mathbf{n}}$, which is displaced of an angle α with respect to $\hat{\mathbf{n}}_s$, whereas the non-ideal one will be shifted of an angle θ w.r.t. $\hat{\mathbf{n}}_s$ and of an angle ϕ w.r.t. $\hat{\mathbf{n}}$.

Moreover, we know that both the components of the force are dependent from the cone angle α, and from Equation (3.33) and (3.34) we notice that both ϕ and θ are also dependent from α. This can help us to define what are the maximum allowable θ angles achievable by the non-ideal force. Indeed, we know that the sailcraft is designed to work with a certain front surface, which has always to be directed to the Sun. Therefore, the angle from the normal to

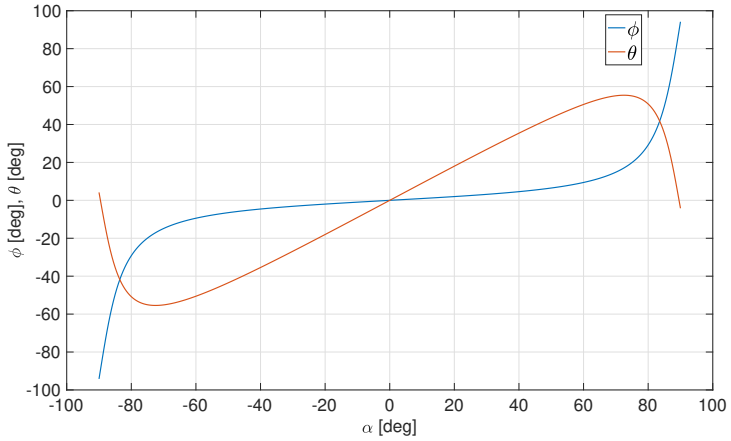


Figure 3.6: ϕ and θ as a function of α

the surface and the Sun-sailcraft line (i.e., α) can never exceed 90° or go below -90°. Thus, using Equations (3.30), (3.31) and (3.33), we can first express ϕ as a function of α as:

$$\phi = \tan^{-1}\left(\frac{(1-C_r s)\cos\alpha\sin\alpha}{(1+C_r s)\cos^2\alpha + B_f(1-s)C_r\cos\alpha + (1-C_r)\dfrac{\epsilon_f B_f - \epsilon_b B_b}{\epsilon_f + \epsilon_b}\cos\alpha}\right) \tag{3.35}$$

Now, considering the boundary values of α, and substituting the values shown in Table 2.3 for the back and front coating of the vehicle, we can plot ϕ as a function of α. Also, by solving Equation (3.34) for θ and substituting the value of ϕ found with Equation (3.35) we can also discover what are the maximum and minimum allowable values for the cone angle θ. Therefore, both θ and ϕ as a function of α are computed for $\alpha \in [-90°, +90°]$ and the resulting curves are shown in Figure 3.6.

From the graph it can be seen that for such a type of sailcraft, the maximum and minimum allowable θ angles are: θ_{max}= +/- 55.433°.

Now that both the ideal and non-ideal forces are introduced and analyzed, we have to establish whether to use one or the other for optimizing the trajectory. Since it has already been demonstrated that for a solar sailing polar mission of our kind, the ideal model gives too optimistic results for the cost function, which are not representative of the real situation (Candy, 2002), (Garot, 2006), we have thus decided to employ a non-ideal solar-sail force model for our thesis study.

Furthermore, it is clear that we cannot point the force in any direction we want, but there are two limiting factors: the fact that we cannot use the front surface of the sail for propelling the sailcraft and the fact that the θ angle is limited within [-55.433°,+55.433°]. Therefore, by plotting the normal to the surface and parallel to the surface components of the solar-sail force, it is possible to observe that the allowable directions of the non ideal solar radiation pressure force can be included in a bubble shape. In Figure 3.7, a graphical representation of such bubble shape for PA=1 is shown.

Also, it is important to notice that the direction of the ideal and non-ideal forces ($\hat{\mathbf{n}}$ and $\hat{\mathbf{m}}$, respectively) in the reference frame associated with the osculating plane around the Sun can be written as:

$$\hat{\mathbf{n}} = \cos\alpha\,\hat{\mathbf{r}} + (\sin\alpha\sin\delta)\,\hat{\boldsymbol{\theta}} + (\sin\alpha\cos\delta)\,\hat{\mathbf{k}} \tag{3.36}$$

$$\hat{\mathbf{m}} = \cos\theta\,\hat{\mathbf{r}} + (\sin\theta\sin\delta)\,\hat{\boldsymbol{\theta}} + (\sin\theta\cos\delta)\,\hat{\mathbf{k}} \tag{3.37}$$

If we want to express these unit vectors using an heliocentric inertial reference system for solving the equations of motion, we first need to rotate them throughout a proper rotation

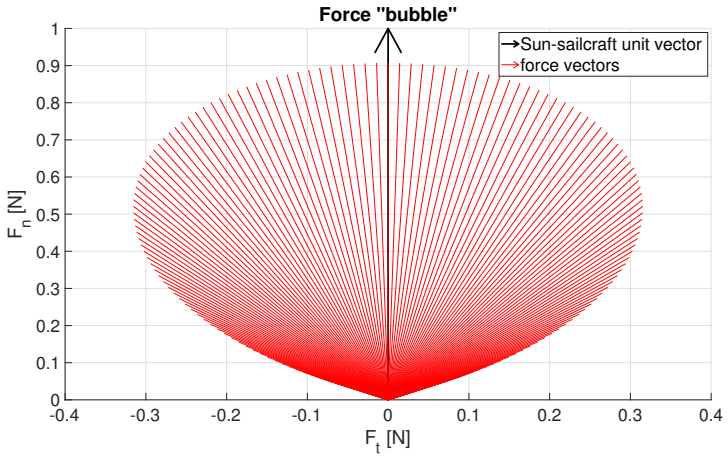


Figure 3.7: Force bubble, where F_n is the component of the force along $\hat{\mathbf{n}}_S$, F_t is the component of the force perpendicular to $\hat{\mathbf{n}}_s$, belonging to the $(\hat{\mathbf{n}}, \hat{\mathbf{n}}_S)$ plane and the black arrow represents the Sun-sail unit vector $\hat{\mathbf{n}}_s$.

matrix. Therefore, we can generally define as $\mathbf{\Phi}$ the transformation matrix and then write the equations of motion written in the heliocentric inertial frame as:

$$\ddot{\mathbf{r}} \mid_H + \mathbf{F}_{grav} \mid_H = F_{SRP}\, \mathbf{\Phi}\hat{\mathbf{m}} \mid_H + \mathbf{G} \mid_H \tag{3.38}$$

Therefore, with all the reference transformations expressed in Section 3.1 we are capable of writing the non ideal solar-sail force in the reference frames of our interests.

3.4.2. Perturbations

In the geocentric phase, the central body will be the Earth, whilst in the heliocentric phase the Sun. However, due to the length of the flight and the various perturbations that will encounter the sailcraft along the way, it is necessary to model these perturbations depending on the position of the spacecraft. A comparison of the perturbing forces as the sailcraft moves away from the Earth was done in Garot (2006) and Candy (2002), where the relevance and effect of the perturbations were studied. There, it was concluded that the main perturbations that will shape the trajectory orbit during the journey to the Sun are: third body perturbations (Moon, Venus, Mercury, Sun, Earth), Earth's atmospheric drag and Earth shadowing. Furthermore, the Sun will act as a third body perturbation when the sail is in the geocentric phase, while the Earth will act as third body perturbation during the heliocentric phase.

In the following sections, we will treat and model the aforementioned perturbations by first investigating the atmospheric drag in Section 3.4.2.1. Then, in Section 3.4.2.2, we will discuss the influence of the shadowing by other celestial bodies on the sailcraft's trajectory during its journey to the Sun. Finally, in Section 3.4.2.3 we will investigate the third body perturbations and their mathematical expressions.

3.4.2.1 Atmospheric Drag

The sailcraft, due to its huge area and relatively small perigee height might be subjected to the atmospheric drag. As explained in Anderson (2010), we can write the resulting force acting in the normal direction of a flat rigid sail as:

$$N = 2\overline{q}A \sin^2 \zeta \tag{3.39}$$

where ζ is the angle of attack between the sail and the velocity vector, $\overline{q} = \frac{1}{2}\rho V^2$ is the dynamic pressure, ρ is the density (modeled as a function of the altitude and position relative to the Earth), V is the velocity, and A is the sail area. Moreover, it must be noted that the angle of attack can be computed as a subtraction between the angle between the velocity vector and the Sun-line (ψ), and the angle between the sail normal and the Sun-line (α). Also, the lift

and the drag components of the aerodynamic force can be derived from Equation (3.39) and written as:

$$\begin{aligned} L &= N\cos(\psi-\alpha) = \rho V^2 A \sin^2(\psi-\alpha)\cos(\psi-\alpha) \\ D &= N\sin(\psi-\alpha) = \rho V^2 A \sin^3(\psi-\alpha) \end{aligned} \tag{3.40}$$

This means that the lift and drag coefficients can be expressed as $C_L = 2\sin^2\zeta\cos\zeta$ and $C_D = 2\sin^3\zeta$, respectively.

For low heights (i.e., high densities) the atmospheric force can be prominent, hence, the sail is positioned edgewise w.r.t. the velocity for low altitudes. In this way, for a perfectly thin sailcraft, no aerodynamic force would be generated. However, this is not the case for a real sailcraft, as there will always be a small thickness that prevents the aerodynamic force to be exactly zero. We model this effect similarly to Garot (2006), by supposing that the angle of attack is assumed to be 3° when the sailcraft flies at low altitudes w.r.t. the Earth. This results in the following aerodynamic coefficients for the vehicle: $C_L = 0.0054706$ and $C_D = 0.0002867$.

3.4.2.2 Solar Eclipse

It has to be noted that the sailcraft cannot generate the solar-sail propelling force when the Sun is eclipsed by the Earth or other celestial bodies. Hence, we must always check whether the sailcraft is being shadowed or not. In the following description, we will only consider the shadowing of the Earth, however, these same definitions hold for any other celestial body. To figure out whether the sailcraft is being shadowed or not, we can first compute the distance $r_\perp$ between the sailcraft and the Earth-Sun line as:

$$r_\perp = |\mathbf{r}|\sin\psi \tag{3.41}$$

where $\mathbf{r}$ is the Earth-sailcraft vector, and ψ is the angle between the Earth-sailcraft vector and the Sun-Earth vector. The angle can be computed as:

$$\psi = \cos^{-1}\left(\frac{\mathbf{s}\cdot\mathbf{r}}{|\mathbf{s}|\cdot|\mathbf{r}|}\right) \tag{3.42}$$

If the distance between the sailcraft and the Sun is bigger than the distance between the Earth and the Sun (i.e., $\mathbf{s}$), and if the perpendicular distance $r_\perp$ is smaller than the Earth's radius, then the sailcraft is considered to be located in the Earth's shadowed zone and no solar radiation pressure force is generated in that phase. This same reasoning also holds for any other celestial body.

3.4.2.3 Third Body Perturbations

From Newton's laws, we know that we can write the accelerations acting on a body i (in our case the sailcraft) w.r.t. an inertial system placed in O, which is the central body (with mass M_0), where the motion of body i is dominated by the central body (i.e., $M_0 \gg m_i$) and influenced by the gravitational attraction of other N-2 bodies (i.e., $3b,j$, for $j = 1,..,$ N-2), as:

$$\frac{d^2\mathbf{r}_i}{dt^2} = -\frac{G(M_0+m_i)}{|\mathbf{r}_i|^3}\mathbf{r}_i + \sum_{3b,j=1;3b,j\neq i}^{N-1} Gm_{3b,j}\left(\frac{\mathbf{r}_{3b,j}-\mathbf{r}_i}{|\mathbf{r}_{3b,j}-\mathbf{r}_i|^3} - \frac{\mathbf{r}_{3b,j}}{|\mathbf{r}_{3b,j}|^3}\right) \tag{3.43}$$

where G is the gravitational constant (G=6.67408 × 10^{-11} m^3 kg^{-1} s^{-2}), $\mathbf{r}$ is the position vector and t the time. All the distances (i.e., $\mathbf{r}_i$ and $\mathbf{r}_{3b,j}$) are computed w.r.t. an inertial reference frame centered in the center of mass of the main attractor (i.e., in the geocentric phase the Earth, in the heliocentric phase the Sun).

The first part of the right hand side of Equation (3.43) describes the two body motion of the sailcraft w.r.t. the main attractor, whereas the second part is the sum of the three body

perturbing accelerations from other planets. Hence, we can generally write the 3^{rd} body acceleration of a planet p on the sailcraft ($\mathbf{a}_p$), as:

$$\mathbf{a}_p = \mu_p\left(\frac{\mathbf{r}_p - \mathbf{r}_{sail}}{|\mathbf{r}_p - \mathbf{r}_{sail}|^3} - \frac{\mathbf{r}_p}{|\mathbf{r}_p|^3}\right) \tag{3.44}$$

where $\mu_p = Gm_p$ is the gravitational constant of the planet, and $\mathbf{r}_p$ is its position w.r.t. the inertial reference system (centered either in the central body).

Equation (3.44) also holds for the acceleration of the Sun when the Earth is the main attractor, and for the acceleration of the Earth, when the Sun is the main attractor.

Both in Candy (2002) and Garot (2006), it has been shown that the main three body perturbations that act on the sailcraft are, for the geocentric phase: the Sun and the Moon, whilst for the heliocentric phase: the Earth, Mercury, and Venus.

In this research, we will thus consider these bodies as third body perturbations during the geocentric and heliocentric phases.

Chapter 4

Guidance

In Section 3.4.1.2, we have investigated the characteristics of the solar-sail force, its direction and how to express it as a function of the clock and cone angles. Several orbital maneuvering techniques exist to control the force in such a way that is desirable for achieving certain objectives (e.g., spiraling inwards as fast as possible, cranking the orbit, reducing the amount of drag, increasing the orbital velocity, etc.). Since we are typically interested in achieving some objectives in specific phases of the flight (e.g. first spiral inwards and then cranking the orbit, etc.) we would like to understand how to orient the solar-sail force (and thus steer the clock and cone angles) to best accomplish these purposes. In general, we distinguish two types of trajectories: locally optimal trajectories and global optimal trajectories. In this section we will deal with local optimal trajectories and local steering laws based on several physical considerations. In McInnes (1999), there is an extensive analysis of the possible local optimal trajectories for a solar-sail during both the geocentric and heliocentric phases: we will derive most of our analysis from that. It is important to notice that the attitude and trajectory of the sailcraft are strictly related since the attitude influences the force generation, which in turn influences the orbit shape. Therefore, we will investigate each of the maneuvers of our interest (e.g. for cranking the orbit, for escaping the Earth, etc.) separately in this chapter. Besides, we will study a possible modelling of the control angles of the sailcraft during both phases.

These optimal control laws are hereby derived and discussed only for the purpose of giving physical insights on our global optimization choices. Indeed, as we will discuss in Chapter 5, in this study we have decided to implement global optimization techniques for designing the trajectory, rather than local optimization techniques. Nevertheless, for doing this, we still need to establish how to model the attitude of the sailcraft while it is in the geocentric and heliocentric phase: for this purpose, we will make use of local optimal trajectories laws.

This chapter will thus be structured as follows: first, in Section 4.1, we discuss the local optimal trajectories during the heliocentric phase. Then, in Section 4.2, we investigate those of the geocentric phase. Finally, in Section 4.3, the flight sections of both phases will be discussed. This division will help us in Chapter 5 to establish the control nodes (in terms of attitude angles) to be optimized during the whole mission. As already stated, we will always assume that the sailcraft is capable of varying its attitude angles instantaneously: we will thus not deal with the design and modelling of an attitude control system.

4.1. Heliocentric Phase

For deriving the various local optimal trajectories to increase/decrease efficiently certain orbital elements in the heliocentric phase, it is first necessary to write the equations of motion of the sailcraft and to solve them with any coordinate system considered suitable (e.g. spherical polar coordinates or else). One effective way to express the solar-sail dynamics is through the Gauss' form of the Lagrange's planetary equations (Wakker, 2015). Lagrange's planetary equations are meant to express the equations of motion as derivative of the various Keplerian orbital elements. We know that, in a generic case in which the sail is subjected to the solar radiation pressure force and to other perturbing forces, these elements change over time. However, at any instant in time, the orbit can be described using fictitious instantaneous Keplerian elliptical orbits, which continuously touch ('osculate') the true perturbed orbit of the spacecraft. For this reason, we call these changing unperturbed orbits 'osculating Keplerian orbits'. In the point of contact, the object's position and velocity are pivotal for modifying the equations of motion into Lagrange's planetary equations. These latter equations only hold for

conservative perturbing forces. However, Gauss, decided to generalize them by introducing also non conservative forces. In McInnes (1999), these equations are further developed in such a way that the rate of change of any arbitrary orbital element (Z) can be written, in a general form, as:

$$\frac{dZ}{dv} = \boldsymbol{\lambda}(Z) \cdot \mathbf{f} \tag{4.1}$$

where $\mathbf{f} = (f_S, f_N, f_W)$ is the vector of the perturbing force(s) and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^T$ is a vector of functions of only the solar-sail orbital elements. The force components are expressed in terms of osculating reference system: meaning that the first component is directed radially (f_S), the second component (f_N) is perpendicular to the radial and laying on the orbital plane and the third component (f_W) that is directed perpendicularly w.r.t. the orbital plane.

As we can see from Equation (4.1), the benefit of having expressed the equations of motion in this form is clear: the equations directly relate the force components to the orbital elements' rate, thus making possible to formulate a steering law to control the force in such a way that the increase/decrease of variation of certain orbital elements is maximized.

Furthermore, we observe that the equations highly depend on the perturbing forces that act on the sailcraft in the various phases of the flight. These perturbing forces were discussed in Section 3.4.2, however, we will hereby consider only the solar radiation pressure force as perturbing force, without introducing the other perturbations, which will be treated as deviations from nominal conditions.

Therefore, we would like to find locally optimal sail-steering laws to maximize instantaneously the rate of change of a certain solar-sail orbital element: in this way we can model the trajectory by orienting the spacecraft in the local optimum found at every instant time. Clearly, the orientation of the spacecraft (i.e., clock and cone angle) has always to fulfill physical constraints. The advantage is that these local optimal trajectories are typically easy to find and usually furnish simple manoeuvre strategies. These will serve as baseline study for understanding how to set up the solar-sail problem in a global optimization framework.

If we assume, that the only perturbing force in the two body system of the Sun and the sailcraft is the solar radiation pressure, then $\mathbf{f}$ coincides with the solar radiation pressure force (whose components are expressed in Equations (3.30) and (3.31), assuming a non-perfect sail, and whose modulus is only a function of the angle θ). This force is exerted along the direction expressed in Equation (3.37). Now, from Equation (4.1) we hint that to maximize the rate of change of a certain orbital element we have to maximize perturbing force along the direction of $\boldsymbol{\lambda}$. Therefore, assuming that the direction of $\boldsymbol{\lambda}$ can be identified through an arbitrary unit vector $\mathbf{q}$, and expressing this unit vector in terms of certain cone and clock angles, we have:

$$\mathbf{q} = \cos\overline{\theta}\,\hat{\mathbf{r}} + (\sin\overline{\theta}\sin\overline{\delta})\,\hat{\boldsymbol{\theta}} + (\sin\overline{\theta}\cos\overline{\delta})\,\hat{\mathbf{k}} \tag{4.2}$$

Then, by using some trigonometric manipulations, we can write the force magnitude along the direction $\mathbf{q}$ as:

$$f_q = \mathbf{f} \cdot \mathbf{q} = f(\alpha)[\cos\theta\cos\overline{\theta} + \sin\theta\sin\overline{\theta}\cos(\delta - \overline{\delta})] \tag{4.3}$$

As seen in Section 3.4.1.2, θ can be expressed as a function of α and ϕ, and ϕ can also be expressed as a function of α. Since we want to find the clock and cone angles (α and δ) that can maximize f_q, it is trivial that the clock angle must be: $\delta = \overline{\delta}$, which means that the sail clock angle is aligned with that of the unit vector $\mathbf{q}$. On the other hand, we can find the optimal α by differentiating the force component along $\mathbf{q}$ w.r.t. the cone angle α:

$$\frac{\partial f_q}{\partial \alpha} = 0 \tag{4.4}$$

In Garot (2006), it has been shown that Equation (4.4) cannot be solved analytically, but numerical methods are required for finding the angle α that maximizes f_q. Once α is found, θ can then also be found.

During the heliocentric phase, we are only interested in maximizing the rate of change of semi-major axis, eccentricity and inclination (as we would like to reduce the distance, crank the orbit and circularize it). The three equations associated with the rate of change of these orbital elements are:

$$\begin{aligned}\frac{da}{d\nu} &= 2\frac{pr^2}{\mu(1-e^2)^2}\left[f_S e\sin\nu + f_N\frac{p}{r}\right]\\ \frac{de}{d\nu} &= \frac{r^2}{\mu}\left[f_S\sin\nu + f_N(1+\frac{r}{p})\cos\nu + f_N\frac{r}{p}e\right]\\ \frac{di}{d\nu} &= f_W\frac{r^3}{\mu p}\cos(\nu+\omega)\end{aligned} \tag{4.5}$$

As we can see, when varying both the eccentricity and the semi-major axis, only in-plane forces are required (i.e., $f_W = 0$). This means that the optimal θ angle can be found as: $\overline{\theta} = atan(\lambda_2/\lambda_1)$, while the optimal clock angle would be: $\overline{\delta} = \frac{\pi}{2}$.

Assuming a solar-sail ideal model, the optimal cone angle would be easily found (knowing that $\overline{\theta} = \overline{\alpha}$ in the perfect case). However, in the non-ideal case, this is not as trivial, and we would need to solve Equation (4.4) for any instant in time (a numerical technique might be necessary for this purpose). Interestingly, both in the ideal and non-ideal case it still holds: $\delta = \overline{\delta} = \pi/2$. In this way, a steering law can be determined, deriving the required solar-sail attitude (for maximizing the rate of change of a certain orbital element) as a function of the osculating orbital elements of the solar-sail. For doing this, we just have to substitute Z with the orbital element to be optimized, in Equation (4.1), and then solve Equation (4.4), after the dot product expressed in Equation (4.3) is computed. In addition to this, we have also to add the operational constraints, which limit the allowable cone angles θ.

For changes of the orbit inclination, we observe from Equation (4.5) that only the out-of-plane component of the solar-sail force is required. This means that the clock angle is equal to: $\delta = \overline{\delta} = 0$ (for a force directed above the instantaneous orbit plane) or $\delta = \overline{\delta} = \pi$ (for a force directed below the instantaneous orbit plane): this suggests that during the cranking phase the sailcraft's clock angle will need to be either 0° (when the force is directed above the orbit plane) or 180° (when it is directed below the orbit plane).

4.2. Geocentric Phase

Now that the heliocentric phase has been modeled and discussed, we can investigate the geocentric phase. Similarly to what has been done for the heliocentric case we would like to divide the geocentric phase in different flight sections that can then be optimized with a global optimization technique.

For the geocentric phase the local steering law are not as trivial as for the heliocentric phase. In principle, we could apply the same reasoning, and try to maximize the increase in the rate of change of the semi-major axis to spiral outwards from the Earth, however, this is not convenient due to operational and physical limitations active in this phase. In particular, two major aspects have to be considered: first of all, the perturbations acting during this phase. For instance, when the sailcraft is below a certain altitude w.r.t. the Earth, the drag acts on it causing a major reduction in the velocity, if the sail is not controlled properly (due to its considerably big area). Therefore, in these cases, it is fundamental to orient the sailcraft edgewise with respect to its velocity vector so that the drag is minimized. Secondly, as we already discussed in Section 3.4.1.2, the solar-sail force cannot be oriented in any direction, but it is constrained to a bubble behind the sailcraft, as shown in Figure 3.7. This also means that the sailcraft will not always be able to orient its force in the direction of the velocity, so that it is accelerated and its escape time is reduced, but it also happens that the only allowable force directions would result to have components opposing to the velocity direction, which would thus slow down the sailcraft causing an higher time of flight for escaping the Earth's gravitational pull. Therefore, in these cases, the sailcraft is oriented edgewise with respect to the Sun-sail line, to avoid a reduction in its velocity.

Therefore, apart from these two aforementioned configurations, we need to establish the orientation of the sailcraft in the remaining parts of the orbit around the Earth to rapidly escape from the Earth's gravitational pull.

In an ideal scenario where only the solar radiation pressure is acting as perturbing force we would like to maximize the energy rate by pointing the solar-sail force along the velocity vector, so that the increase in the semi-major axis is maximized (Coverstone and Prussing, 2003). However, this is not always possible, as the force direction is limited within a bubble and there are many perturbing forces that may play a pivotal role (e.g., the atmospheric drag for low altitudes).

4.3. Flight Sections

In this section, we will describe the flight sections for both the geocentric and heliocentric phases. In particular, the former phase will be investigated in Section 4.3.1, whereas the latter in Section 4.3.2. For both cases, we will use the knowledge derived from Sections 4.1 and 4.2 to establish the division criteria of these flight sections.

4.3.1. Heliocentric Flight Sections

In the heliocentric phase, we distinguish 4 main phases: spiral inwards, circularization, cranking of the orbit, and spiral outwards. Indeed, the sailcraft has to first approach the Sun at near distances, and then, having circularised the elliptical orbit to make it circular, it can start to increase the inclination of the orbit up to the 90° required by the final orbit. Furthermore, it is necessary to increase/decrease the radius of the inclined orbit to the one required by the final orbit (i.e., 0.4 au). These phases will be performed consequently.

Phase 1: Spiral Inwards

In the first phase of the heliocentric trajectory, the sailcraft has to drastically reduce its distance from the Sun (i.e., da/dv minimized). As already explained, this means that the clock angle equals to $\delta = 90°$, whereas the cone angle α_{S1} has to be optimized. Furthermore, for inward spiraling, the cone angle must always be negative (i.e., $\alpha_{S1} \in [-90°, 0°]$).

Phase 2: Circularization

At a distance R_1 (to be optimized), the sailcraft will start the circularization of the orbit, which will bring it to a circular orbit of radius R_2 (to be optimized). Therefore, in this phase our main objective is to maximize the rate of change of the eccentricity (i.e., de/dv). We have already mentioned that the clock angle must be $\delta = 90°$ and the cone angle α_{S2} is the only one to be optimized in this phase. For circularizing the orbit, there are not any physical aspects that constrain the cone angle to specific values, hence the optimization process will be performed in the domain where the cone angle is defined (i.e., $\alpha_{S2} \in [-90°, 90°]$).

Phase 3: Orbit Cranking

Having a circular orbit at a distance R_2 from the Sun, we can start the cranking phase. This phase usually turns out to happen at a low distance (R_2) of the sailcraft with respect to the Sun during its cruise: indeed, this maximizes the force magnitude available for performing the maneuver, allowing to speed up the process. The objective is to maximize the rate of change of the inclination (i.e., di/dv). As it was already mentioned, the clock angle is fixed and corresponds to $\delta = 0°$ for a perturbing force directed above the orbital plane, and $\delta = 180°$ for a perturbing force directed below it. Conversely, the optimal cone angle of the orbit cranking phase (i.e., α_{S3}) will be an outcome of the optimization process. Nonetheless, the optimal cone angle must always be positive (i.e., $\alpha_{S3} \in [0°, 90°]$).

Phase 4: Spiral Outwards

This phase is analogous to the spiral inwards phase, with the difference that da/dv has to be maximized. Therefore, the clock angle results to be the same ($\delta = 90°$). Similarly to the other phases, the cone angle of the spiral outwards phase (i.e., α_{S4}) will be maintained fixed during

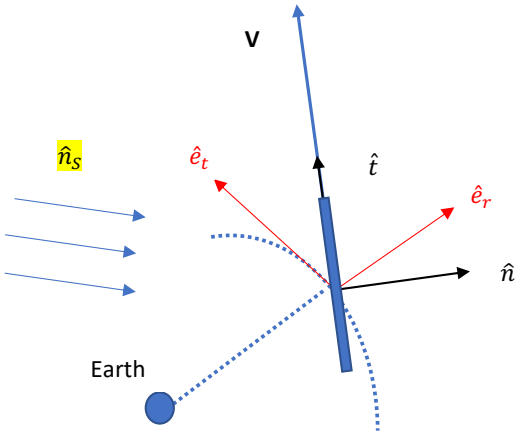


Figure 4.1: Sailcraft orientation during the drag phase.

the mission and its value will be established throughout the global optimization process. Besides, the optimal cone angle must always be positive (i.e., $\alpha_{S3} \in [0°, 90°]$).

4.3.2. Geocentric Flight Sections

As it has been done in previous studies, we can divide the geocentric phase into 4 different sub-phases: drag phase, towards the Sun phase and acceleration phase (Candy, 2002), (Garot, 2006), (Spaans, 2009). In this case, contrary to what happened in the heliocentric phase, the various flight sections happen several times along one single orbit, depending on the position of the sailcraft with respect to the Sun and the Earth. These flight phases will result in several different attitude angles of the sail, which will then be established according to the results obtained in a global optimization framework, if their value cannot be trivially established (as it happens for the drag phase). For all these phases, a clock angle of $\delta = \pi/2$ will be maintained. Indeed, as we have seen in Section 4.1, unless the main objective is to generate out-of-plane forces for changing either the inclination or the right ascension of the ascending node, which is not the case for the geocentric phase, the optimal clock angle is always set to 90°.

Drag Phase

Due to the huge sail area, when the sailcraft is at an altitude of around 1500 km or lower, the drag effect starts to perturb the orbit, and the sailcraft is thus oriented edgewise w.r.t. its velocity to avoid this effect. The orientation of the sailcraft in this phase is shown in Figure 4.1. Since the sailcraft is placed edgewise w.r.t. the velocity vector, its orientation w.r.t. the geocentric frame can be retrieved as:

$$\hat{\mathbf{t}} = \hat{\mathbf{V}} \qquad \hat{\mathbf{p}} = \hat{\mathbf{n}}_s \times \hat{\mathbf{V}} \qquad \hat{\mathbf{n}} = \hat{\mathbf{t}} \times \hat{\mathbf{p}} \tag{4.6}$$

where $\hat{\mathbf{V}}$ is the unit vector that defines the solar-sail velocity direction, whereas $\hat{\mathbf{n}}_s$ is the unit vector that describes the Sun-sail line. As a consequence, the cone angle α can be computed as:

$$\alpha = \cos^{-1}\left(\hat{\mathbf{n}}_s \cdot \hat{\mathbf{n}}\right) \tag{4.7}$$

By using the retrieved cone angle value and by writing the three unit vectors: $(\hat{\mathbf{n}}, \hat{\mathbf{t}}, \hat{\mathbf{p}})$, w.r.t. the geocentric frame (using $\hat{\mathbf{n}}_s$ and $\mathbf{V}$ in the geocentric frame), we can express the components of the force shown in Equations (3.30) and (3.31). It is thus clear that, in this phase, the cone angle α can be precisely determined and does not need to be optimized.

Towards the Sun Phase

When the unit vector that points from the Sun to the sail ($\hat{\mathbf{n}}_s$) and the velocity unit vector ($\hat{\mathbf{V}}$) form an angle bigger than 90°, then the sailcraft is inside the towards the Sun phase.

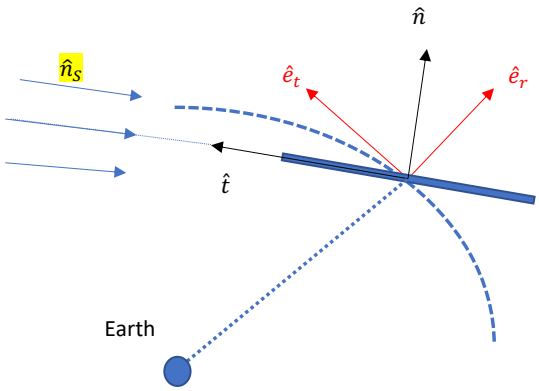


Figure 4.2: Sailcraft orientation during the towards the Sun phase.

Mathematically, this can be verified by checking if $\chi \geq \pi/2$, where:

$$\chi = \cos^{-1}\left(\hat{\mathbf{n}}_s \cdot \hat{\mathbf{V}}\right) \tag{4.8}$$

where $\chi \in [0, \pi]$.

If this condition is verified, then the spacecraft is oriented edgewise with respect to the Sun-sail line (the tangential body axis is aligned with the Sun-sail unit vector). In this way, no components of the force are generated in the opposite direction w.r.t. the velocity vector, thus avoiding to slow down the sail. The orientation of the sailcraft in this phase is shown in Figure 4.2.

Therefore, it results that the attitude of the sail, similarly to what happened in the drag phase, can precisely be determined and that the generated solar radiation pressure force is zero (i.e., $\mathbf{F}_{SRP} = 0$), as the sail always points edgewise w.r.t. the Sun-sailcraft line (i.e., $\hat{\mathbf{t}} = -\hat{\mathbf{n}}_S$). In practice, this means that the cone angle is set to ninety degrees (i.e., $\alpha = 90°$).

Acceleration Phase

The sail enters this phase whenever is not in the drag phase and $\chi \in [0, \pi/2)$. If these two conditions apply, then we aim to steer the sailcraft to escape the Earth as fast as possible. Since the solar radiation pressure is not the only perturbation but there are other perturbations active in this phase (e.g., third body perturbation by the Moon), directing the sailcraft's solar radiation pressure force in the velocity direction does not necessary imply that the resulting trajectory will be optimal. Therefore, the attitude of the sailcraft in this phase will be split into different subsections, which will then be fed to a global optimizer. For doing this, we have used the same strategy proposed in Garot (2006) and Spaans (2009): according to both these studies, dividing this phase into six different subsections provides a good control strategy for the cone angle. Furthermore, it was also found that when these six angles are chosen such that they constitute a symmetric set, then they result to be equal in pairs, and the set of angles can thus be reduced to only three.

As a result, this phase will be split in six phases in which the sail will change three different cone angles (i.e., α_{E1}, α_{E2} and α_{E3}), each of these angles will be active in a different flight section of the orbit. In particular, the following conditions define these sections:

1. if $\chi \in [60°, 90°)$ then $\alpha = \alpha_{E1}$
2. if $\chi \in [30°, 60°)$ then $\alpha = \alpha_{E2}$
3. if $\chi \in [0, 30°)$ then $\alpha = \alpha_{E3}$

These three cone angles will be determined as outcomes of the global optimization procedure, which will be thoroughly discussed in Chapter 5. Also, their values must range between zero

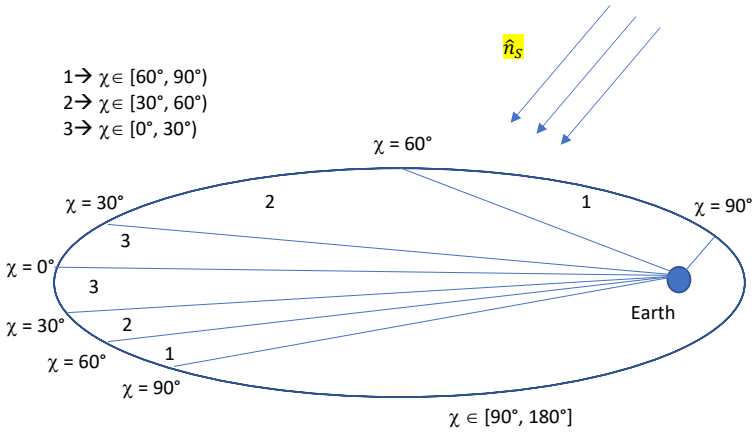


Figure 4.3: The acceleration phase is divided into three phases, according to the χ angle values.

and ninety degrees (i.e., $\alpha_{Ei} \in [0°, 90°)$ for $i = 1, 2, 3$). If this is not verified, it means that the sailcraft is not in the acceleration phase as it cannot generate any component in the velocity direction. A graphical representation of the acceleration phase is shown in Figure 4.3.

Chapter 5

Optimization

In previous chapters, we have discussed all the models used for simulating the orbit of a solar-sail, during its journey to the Sun. The purpose of this research is to compare the performances of several different global optimization techniques in optimizing such orbit. In this chapter, we will thus discuss the various optimization approaches and techniques employed. Moreover, the constraints formulation and handling will be investigated. In particular, in Section 5.1, the theoretical framework of global optimization techniques as well as several popular methods will be discussed. Whereas, in Section 5.2 some performance metrics for evaluating multi-objective algorithms will be presented. Then in Section 5.3, we will separately discuss the newly implemented ant colony optimizer (for both SO and MO problems). Finally, the optimization problem of our interest and the implemented approaches will be discussed in Section 5.4.

5.1. Global Optimization

Global optimization techniques are usually critical in terms of computation time, however, thanks to the fast developments in computer power in recent years, the interest in global optimization methods have been constantly increasing. Generally, two different kinds of global optimization algorithms are distinguished: deterministic and stochastic. Deterministic optimization provides theoretical proof that the found solution is indeed the global best one. Hence, this term usually refers to a complete and rigorous optimization method which converges to an optimum in a finite amount of time (e.g. linear programming). Deterministic methods do not include any randomness, and they guarantee a finite amount of work for reaching the solution. However, they also require physical insights into the problem which can allow them to formulate them rigorously from the mathematical point of view.

On the other hand, the stochastic method use randomly generated variables, and they basically provide optimization methods without the need for any particular insight into the problem. This means that the optimization problem can be treated as a black box. Furthermore, most of the stochastic methods are heuristic (i.e., it cannot be proved that they can find a global optimal solution within a certain amount of finite time). These heuristic methods seem very suitable for trajectory optimization problems. Some of the most important stochastic algorithms are evolutionary algorithm (EA, among which we distinguish: genetic algorithms (GA), differential evolution (DE), and many other forms), ant colony optimization (ACO), particle swarm optimization (PSO). Also, many other variants of these algorithms have been also derived (e.g. extended ant colony optimization (EACO), self-adaptive differential evolution (SADE), etc.). The European Space Agency, during the first years of the 20th century, has extensively studied GA, PSO, MPSO, DE, and ASA, to find the best algorithms for trajectory optimization using some benchmark problems. It has been demonstrated in those years (see for instance Myatt et al. (2004) and Vinkó et al. (2007a)) that MPSO, DE, ASA seem to be promising for trajectory optimization. However, recently it has also been proved that ACO can even outperform these algorithms for certain problems (Schlüter et al., 2012), (Schlüter et al., 2017), (Schlüter, 2014). Before discussing in detail all the methods, a distinction has to be made between single and multi-objective optimization: this will be done in the following sections.

5.1.1. Single-Objective Optimization

In this case, the general optimization method can be mathematically written as:

$$\min_{\mathbf{x}\in\Sigma}(f(\mathbf{x})) \tag{5.1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ are the variables to be optimized, $f(\mathbf{x})$ is the objective function, and Σ is the search space of the variables. This search space is typically confined due to the presence of m equality ($h_i(\mathbf{x})$) and $p - m$ inequality ($g_i(\mathbf{x})$) constraints of the kind:

$$h_i(\mathbf{x}) = 0 \qquad i = 1, \ldots, m \tag{5.2}$$

$$g_i(\mathbf{x}) \leq 0 \qquad i = m, m+1, \ldots, p \tag{5.3}$$

All available optimization techniques try to minimize the objective function $f(\mathbf{x})$ while satisfying the constraints. In the case of single-objective optimization, all the objectives need to be written in one single function. This means that if multiple objectives have to be accounted for, it is necessary to apply some sort of averaging technique to express them in one function (e.g. the simplest method just adds the various objectives). Furthermore, it is possible to transform a constrained single-objective problem, into an unconstrained single-objective optimization problem by writing a new fitness function that is the sum of the previous one plus a certain penalty function that accounts for the constraints violation.

These penalty functions are really problem dependent (Parsopoulos and Vrahatis, 2002), (Yang et al., 1997), and they penalize the cost function in case that some constraints are violated or in case that the final state is not reached. In particular, this way of expressing the constraints is useful for not discarding the individuals that have good performance in terms of cost function, but they slightly violate the constraints: this might produce new individuals that perform well and satisfy the constraints. However, the penalties have to be chosen and tuned accurately: too weak penalties may allow the unfeasible solution to thrive, whereas too strong penalties might prevent the algorithm from finding better individuals. A meta-algorithm exists and has been used in this research for allowing researchers to apply single-objective unconstrained algorithm for constrained problems.

In general, several single-objective optimization algorithms have been tested and studied for a solar sailing polar mission in the past (see, for instance, Spaans (2009) and Garot (2006)). From this, it emerged that DEI (an outdated algorithm that used a differential evolution technique) outperformed GA and PSO for the optimization of the entire trajectory (i.e., geocentric and heliocentric phase). These studies are however outdated and some of the proposed methods have been refined and improved. Besides, some other algorithms have been introduced and have demonstrated amazing performances for trajectory design problems (e.g., ant colony optimization). In this research study, several popular single-objective optimization algorithms are used and benchmarked. These include the following single-objective algorithms for unconstrained problems: standard differential evolution (DE) (Storn and Price, 1997), self-adaptive differential evolution (SADE) (Brest et al., 2006), another differential evolution variant (DE1220)[1], a simple genetic algorithm (SGA) (Oliveto et al., 2007), a particle swarm optimizer (PSO) (Eberhart and Kennedy, 1995) and an artificial bee colony optimizer (Karaboga and Basturk, 2007). Furthermore, we have also used a self-adaptive constraint handling meta-algorithm for permitting the usage of any single-objective unconstrained optimization algorithm (as all the aforesaid ones) to be applied on single-objective constrained problems (Wright and Farmani, 2001), (Farmani and Wright, 2003). Most of these algorithms are well known within the evolutionary computation community and they are often used for space applications. Their thorough description is presented in Appendix C. Their performances will be compared to the newly implemented single-objective ant colony optimizer, which is discussed in detail in Section 5.3.1.

5.1.2. Multi Objective Optimization

Combining multiple objectives in one single function introduces ambiguity in the problem since the search for the optima will rely on how the single-objective functions are defined and how they are weighted. Therefore, it is often preferred to evaluate the various objectives

[1] https://esa.github.io/pagmo2/docs/cpp/algorithms/de1220.html

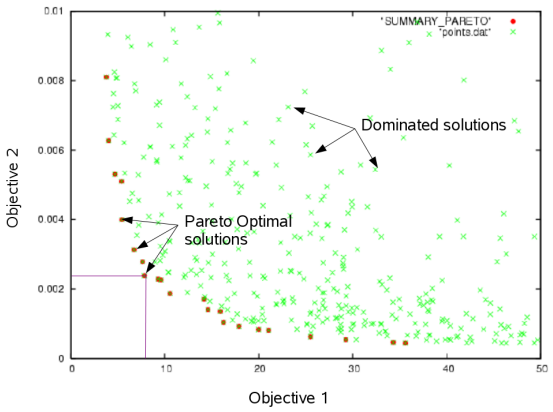


Figure 5.1: Pareto front example in a bi-objective optimization problem [2]

separately, by means of different objective functions, which thus lead us to multi-objective optimization techniques. Assuming that there are M objective functions, the problem can be mathematically formulated as follows:

$$\min_{\mathbf{x}\in\Sigma}(\mathbf{f}(\mathbf{x})) = \min_{\mathbf{x}\in\Sigma}[f_1(\mathbf{x}), ..., f_M(\mathbf{x})]^T \tag{5.4}$$

The problem is generally subjected to the following equality and inequality constraints:

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), ..., h_i(\mathbf{x})]^T = \mathbf{0} \qquad i = 1, ..., m \tag{5.5}$$

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), ..., g_i(\mathbf{x})]^T \leq \mathbf{0} \qquad i = m, m+1, ..., p \tag{5.6}$$

For multi-objective optimization, the problem is thus to simultaneously minimize the components of a vector of functions ($\mathbf{f} = (f_1, ..., f_M)^T$). Each component of this vector is a function of the design variables $\mathbf{x} = (x_1, ..., x_n)^T$. The problem is typically not uniquely solvable, but a set of equally efficient alternative solutions is possible. These solutions, among all the found ones, constitute the Pareto front, when they are plotted in the objective space. In Figure 5.1, we show an example of a Pareto front for a bi-objective example. When comparing individuals i and j, with objective vector functions $\mathbf{f}_i$ and $\mathbf{f}_j$ we affirm that i dominates j in the Pareto sense if and only if:

$$\forall k \in \{1, ..., K\} : f_{i,k} \leq f_{j,k} \qquad \exists k \in \{1, ..., K\} : f_{i,k} \neq f_{j,k} \tag{5.7}$$

The local Pareto set is thus a set of individuals that has the best performance taking all the objectives into account. In case that no further Pareto improvement is possible (i.e., there are no individuals with one better single-objective without having the other objectives worse) Pareto optimality is achieved. The set of Pareto optimal solutions is then called Pareto front. It is clear that, among the population, it is possible to remove the Pareto front and identify the second best Pareto front, and so on. In this way, one is able to rank the various fronts, assuming that all the individuals belonging to the same front have the same fitness value, and thus the individuals can only be evaluated depending on the front to which they belong.

For establishing the Pareto front rank number of each solution, the concept of dominance (expressed in Equation (5.7)) is used. Indeed, the individuals that dominate all the others are checked: in this way, the first front is established. Then these individuals are excluded and the same operation is repeated. In this way, we will obtain N Pareto fronts. As we will see in this chapter, many multi-objective optimization techniques make use of this concept for evolving the population throughout the generations.

Since very promising results of multi-objective optimization techniques have been studied and demonstrated for trajectory optimization problem in the last years (see for instance

[2] `http://www.cenaero.be/Page.asp?docid=27103&langue=EN`, last access: 15 May 2018

Castellini (2008), Zhang and Li (2007a)), and since the main focus of past solar sailing polar missions (see, for instance, Spaans (2009), Garot (2006), Candy (2002)) has been putting together multiple objectives in one single-objective function (due to the limited availability of computational power for running these multi-objective algorithms), we will introduce and apply several multi-objective optimization algorithms for our solar sailing polar mission. This will require another formulation of the optimization problem. It has to be noted that this formulation of the problem, with respect to the one based on a single weighted objective function in which multiple objectives are encapsulated, provides multiple possible optimal solutions in one single simulation run, without the need of reformulating the single-objective function and run the algorithm many times. Indeed, the presence of multiple objectives in a problem often requires to find many optimal solutions (known as Pareto-optimal solutions). The methodology and implementation of MO algorithms are thus completely different from the single-objective ones. In this study, we have used four different multi-objective optimizers: a nondominated sorting genetic algorithm (NSGA-II) (Deb et al., 2002), a multi-objective evolutionary algorithm with decomposition (MOEA/D) (Zhang and Li, 2007b), a nondominated sorting particle swarm optimizer (NSPSO) and a multi-objective ant colony optimizer (MHACO). The first two are popular algorithms with a strong heritage in space applications. Their description is presented in Appendix C. Whereas, the MO ant colony optimizer was entirely conceived and developed in this study: its description is therefore treated extensively in Section 5.3.3. Finally, NSPSO was derived from literature but entirely implemented and slightly modified in this thesis study: its description is presented in Section 5.1.2.1.

5.1.2.1 Non-Dominated Sorting Particle Swarm Optimization (NSPSO)

This algorithm represents a multi-objective extension of particle swarm optimization and its implementation and description have been discussed in several publications (Li, 2003), (Li, 2004), (Fonseca et al., 1993). In these papers, the performances of an implemented multi-objective PSO have been compared to other standard MO algorithms (such as NSGA-II) in several test problems. It seems that this algorithm is capable of being competitive and in some cases superior to these standard MO algorithms. However, while several indications are given in the papers on how to implement this algorithm, its thorough description is not available. Hence, in this thesis study, when we implemented this nondominated sorting particle swarm optimizer (NSPSO) we faced some difficulties in the definition of certain parameters, whose implementation is not exactly presented in the original publications. This has also provoked our implemented algorithm to be slightly different than the original one. For this reason, we have decided to repeat the benchmark of this algorithm on several standard problems and compared them to MOEA/D and NSGA-II. The results of this benchmark are presented in Appendix B.

Having said this, we can move to the mathematical description of this optimizer. Similarly to its single-objective counterpart, NSPSO is a bio-inspired algorithm that tends to guide a certain population towards the most promising area of the search space. However, in PSO a particle is modified only using its personal and global best to produce the offspring (as shown in Equation (C.7)). This represents a serious limitation for extending the algorithm to multi-objective, as in this case, it is not generally possible to define a global best. Also, sharing information among individuals is much more pivotal in the multi-objective case. In NSPSO, an attempt is made to allow such sharing and to modify PSO in such a way that it can also work well for multiple objectives.

For multi-objective problems, it is crucial to have an algorithm that not only converges towards the Pareto-optimal front, but that it also maintains diversity in the population. Hence, similarly to what is done in NSGA-II, NSPSO adopts the nondominated sorting concept to achieve this. This helps us to have a sorting strategy that can rank the various individuals. Moreover, in NSGA-II, the crowding distance concept is used for maintaining diversity in the population. In NSPSO, two different diversity mechanisms are encoded: crowding distance comparison (same as the one adopted for NSGA-II), a widely-known niching method (rey Horn et al., 1994).

First of all, the entire population of NP particles' personal best and their NP offspring are combined to form a $2NP$ set of individuals. On this set, the nondomination sorting technique

is applied to divide them into different nondomination levels. Based on these levels, we only select NP individuals out of this set, to be propagated to the next generation. These particles will result to be the new NP particles' personal best in the future generation. While this process will effectively guide the optimization process towards the Pareto-optimal front, it will however not maintain diversity in the population. We thus need a strategy that can help us to rank the individuals, which belong to the same nondomination rank. For allowing this, two different techniques can be used (depending on the user preferences):

1. Crowding distance assignment: this is the same technique used for NSGA-II. Its thorough description is presented in Appendix C.
2. Niche count: the niche count m_i of a certain individual i is calculated as the number of other particles within a certain σ_{share} distance (computed as the Euclidean distance) from i. This computation is executed dynamically, at every generation, and allows us to count, for every particle i, what is the number of other particles within a σ_{share} distance from it. Although this helps us to determine the more overcrowded particles, it, however, has the drawback to having a user-defined σ_{share} parameter that substantially drives the definition of crowdedness. For this reason, the σ_{share} definition and update is done dynamically, using the method proposed by Fonseca et al. (1993). Thus, the user does not have to worry about selecting or tweaking any σ_{share} parameter. In the case that a bi-objective problem is being optimized, the parameter is selected as follows:

$$\sigma_{share} = \frac{u_2 - l_2 + u_1 - l_1}{NP - 1} \tag{5.8}$$

where NP is the population size and u_i, l_i are the upper and lower bounds, respectively, for each of the two objective function values, computed over the entire population. Once the niche count values are assigned to each individual belonging to each nondomination rank, we can thus rank these individuals and only select the best ones to be preserved for future generations.

The same authors that have proposed the nondomination strategy coupled with either the niching count or crowding distance diversity mechanism, have also decided to introduce a different strategy for ranking the $2NP$ individuals and establish the set of best particles' to be propagated. This is called the maxmin fitness function method and is introduced in Li (2004). For coherence, the multi-objective extension of PSO that makes use of this technique should have had another name (e.g., *maxmin*PSO, or MMPSO), however, we have decided to also encode this strategy in the NSPSO algorithm, and to leave the user the possibility to choose between nondomination ranking with either niche count or crowding distance comparison, and maxmin function strategy.

The maxmin strategy was first introduced in game theory (Luce and Raiffa, 1989). Given two decision vectors $\mathbf{u}$ and $\mathbf{v}$, which belongs to the search space, we can define the min function as the function that obtains the minimal value from the following set: $\{f_i(\mathbf{u}) - f_i(\mathbf{v}) | \forall i \in \{1, \dots, m\}\}$ (where m is the number of objectives). Mathematically, the min function is thus expressed as:

$$min_{i=1,\dots,m}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\} \tag{5.9}$$

Moreover, we define the max function as the maximum over the set of minimal values of all possible pairs of $\mathbf{u}$ and $\mathbf{v}$ (where these two individuals belonging to the search space must be different). Keeping this in mind, we can define the maxmin fitness value for the decision vector $\mathbf{u}$ as follows:

$$f_{maxmin} = max_{j=1,\dots,NP;u\neq v}\{min_{i=1,\dots,m}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\}\} \tag{5.10}$$

We hence have two different loops: first we extract the minimum of the difference in the fitness values of two different decision vectors, over all the objectives. Afterwards, we loop over all the individuals in the population (i.e., NP) and we seek the maximum of these minima. One obvious property of the maxmin function, defined in Equation (5.10), is that for any decision

vector to be a nondominated solution of the current population, it is required that its maxmin fitness value is less than zero. From its definition, it is also clear that the maxmin fitness function can be used for rewarding diversity and penalizing clustering, besides using it for determining the nondominant solutions. We can thus use this function without introducing any other diversity mechanism such as crowding distance comparison or niche counting. More details on the maxmin function and its properties can be found in Balling (2003).

Now that we have discussed both the maxmin strategy and the nondominated sorting strategy, we can explain how future individuals will be generated. For doing this, the same equations used for PSO (i.e., Equations (C.8) and (C.7)) will be used. The only difference is that now there is the possibility to constrain the maximum value of the velocity to a certain user-defined threshold called V_{MAX}. Also, the user has the possibility to tweak a χ parameter that will slightly modify the position update. In particular, in this case, Equation (C.8) becomes:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \chi \mathbf{v}_i \tag{5.11}$$

To summarize, the NSPSO pseudo-code can be expressed in the following steps:

1. A population of NP individuals is first initialized and stored in a list called *PSOList*.
2. Either the maxmin fitness or the nondominated sorting strategy coupled with either niche count or crowding distance comparison is used for storing the individuals in a list called *nonDomPSOList*.
3. Equations (C.7) and (5.11) are used for producing the offspring, which consists of other NP individuals.
4. Both the offspring and the parents are stored in a list of size $2NP$ called *nextPopList*.
5. The maxmin strategy or the nondominated sorting applied with either crowding distance or niche counting method is used for ranking the individuals within the *nextPopList*.
6. After having sorted the individuals, the first NP individuals are stored, to be used as *PSOList* in the future generation.
7. The algorithm goes back to Step 2 and proceeds until a termination criterion is met.

One issue that we encountered when trying to implement this algorithm consists on the fact that in Li (2003) it is claimed that the ω parameter of Equation (C.7) is updated dynamically in the algorithm. However, we have decided to implement a fixed user-defined ω parameter since the mathematical expression for achieving this dynamical behavior is not indicated. This, of course, completely changes the performances of the algorithm, due to the fact that the particles' velocity substantially drives the generation of new offspring and thus the entire optimization process. For this reason, we have also decided to benchmark from scratch this new implementation of the NSPSO algorithm, whose results are presented in Appendix B.

To summarize, the input parameters, besides the number of generations and the population size, to be chosen for starting the optimization process are the following:

1. ω: the particles' inertia weight. It holds $\omega \in [0, 1]$
2. η_1: the magnitude of the force applied to the particle's velocity in the direction of its previous best position. It holds $\eta_1 > 0$.
3. η_2: the magnitude of the force applied to the particle's velocity in the direction of its global best. It holds $\eta_2 > 0$.
4. χ: the velocity scaling factor. It holds $\chi > 0$.
5. V_{MAX}: maximum allowed particle's velocity. It holds $V_{MAX} \in (0, 1]$

6. LSR: the leader selection range parameter. This parameter is used for selecting the leader of each particle among the best $LSRmod(pop)$ individuals (where pop is the population size and $mod()$ represents the modulo operation). Following the same terminology as used for PSO, this means that LSR regulates the neighborhood size in which the best individual is selected. It holds $LSR \in [1, 100]$.

7. $DiversityMechanism$: the diversity mechanism can be chosen between niche count method, crowding distance and maxmin function.

5.2. Performance Metrics

Before diving into the mathematical formulation of the algorithms used, it is first fundamental to discuss the performance metrics that will be used for benchmarking different multi-objective (MO) optimizers. Indeed, while for a single-objective minimization problem it is sufficient to check in how many function evaluations the algorithms have managed to reach a certain fitness value for trading-off different algorithms and establish a performance ranking, which will generally vary depending on the optimization problem; this is not the case for multi-objective problems. In fact, as we have already pointed out, an MO optimization procedure often leads to finding multiple optimal solutions. One possible way of comparing multiple algorithms could be to simply count the number of Pareto optimal individuals at the end of the evolution. However, this does not tell us anything about the closeness of these individuals to the real optimal front (if it is known) and the spread of these individuals within the Pareto front. For circumventing these issues, multiple performance metrics are typically combined. In particular, these are typically classified in three major classes:

1. Convergence metrics: they evaluate how far is the final Pareto front from the real one. Typically, we only know the Pareto optimal front for mathematical problems (that are often used as benchmark problems), whereas we do not have this information for many real world problems. In this case, we limit ourselves to establish what is the best, if any, final Pareto optimal front among the algorithms.

2. Diversity metrics: they evaluate the scatter of solutions in the final Pareto front.

3. Metrics for both convergence and diversity.

More than 50 convergence metrics have been developed until now (Grosan et al., 2003). However, the analysis of all these metrics and their application for benchmarking the MO algorithms used in this study goes beyond the scope of this research. We will thus limit ourselves to only discuss two performance metrics: one that can be used in any problem and is typically used for trading-off performances of different MO algorithms, another one that is very useful for benchmarking performances in two well known test suites.

The first one is the hypervolume indicator (often referred to as Lebesgue measure (Fleischer, 2003) or S-metrics (Zitzler, 1999)). This is a popular quality measure for MO optimizers since it represents both a convergence and a diversity metric.

The second one is called p-distance: this is a particular kind of metric that is only useful for peculiar problems in which the true Pareto front is known and in which the Pareto optimal solutions can be found by minimizing a certain distance function. This latter metrics will only be used for two widespread test suites (i.e., ZDT and DTLZ) that construct their problems employing the aforementioned distance function.

Due to the fact that one single metric might not capable of establishing the quality of solutions both in terms of diversity and convergence, we will combine these two metrics whenever possible. Also, we will always combine them with the number of final Pareto front individuals found, as well as with a graphical representation of the final fronts, whenever possible (i.e., for two or three-objectives problems).

5.2.1. Hypervolume Metric

This metric was first introduced in Zitzler et al. (2002) and it has soon become one of the most used performance metrics for MO algorithms. Indeed, not only does this metric evaluate

both convergence and diversity using a single value, but it also has the property of being strictly Pareto compliant, meaning that the Pareto optimal front has the key characteristic of maximizing the hypervolume value. This means that any dominated set will result in a lower hypervolume value.

Before introducing the mathematical definition of this metric, it is first important to give some definitions. Given a set of $\mathbf{x} \in \mathbb{R}^n$ decision vectors and a set of $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ objective functions, we have already discussed in Section 5.1 that since objectives can be conflicting, we search for a set of good solutions by defining the concept of Pareto dominance. This will hereby be expressed using the symbol: "$\prec$". Given two decision vectors $\mathbf{x}$, $\mathbf{x}$', we say that the former dominates the latter if $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\mathbf{x}')$, and in these cases we write:

$$\mathbf{x} \prec \mathbf{x}' \iff \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}') \tag{5.12}$$

We can thus define the Pareto optimal set as: $\{\mathbf{x} \in \mathbb{R}^n | \nexists\, \mathbf{x}' \in \mathbb{R}^n : \mathbf{x}' \prec \mathbf{x}\}$. The corresponding image in the objective space $\mathbb{R}^m$ is called Pareto front.

Typical definitions of the hypervolume indicator are based on polytopes (Zitzler, 1999). Nevertheless, without introducing these concepts it is still possible to introduce the hypervolume concept (which is nothing more than a generalization of the area, and volume concepts in n-dimensions) in a simpler way. The computation of the hypervolume in 2-dimensions is shown in Figure 5.2.

By taking a sub-set B of the objective functions space and letting Λ denote the Lebesgue measure (which is the common method to measure subsets of n-dimensional Euclidean space, and which corresponds to the concepts of length, area and volume for the case of n=1, 2 and 3, respectively), then the hypervolume (I_H) can be defined as:

$$I_H(B, \mathbf{y}_{ref}) = \Lambda\left(\bigcup_{\mathbf{y} \in B} \{\mathbf{y}' | \mathbf{y} \prec \mathbf{y}' \prec \mathbf{y}_{ref}\}\right), \qquad B \subset \mathbb{R}^m \tag{5.13}$$

where m is the objective space dimension, $\mathbf{y}, \mathbf{y}' \in B$ belong to a sub-set of the overall objective function vectors and where $\mathbf{y}_{ref} \in \mathbb{R}^m$ refers to a reference point that should be dominated by all Pareto optimal solutions.

Albeit recent improvements, the exact computation of the hypervolume for high dimensions is still a bottleneck and quickly becomes unfeasible for objective space dimensions higher than 10. For overcoming this problem, a lot of research has been carried on for approximating the hypervolume computation for high dimensions, while still using exact algorithms for low dimensions (such as 2, 3 or 4). In particular, in Nowak et al. (2014), a study is performed to benchmark the state-of-art hypervolume algorithms. From this study, it is concluded that for two and three dimensions, two exact algorithms shall be used (Guerreiro et al., 2012), (Beume et al., 2009), due to their low computational time. However, for higher dimensions (i.e., from four to ten), the Walking Fish Group algorithm (While et al., 2011) seems to

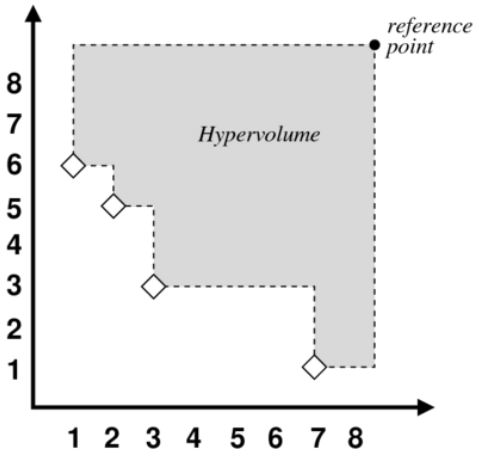


Figure 5.2: Graphical representation of the hypervolume computation in two dimensions [3].

be the most suitable. This algorithm takes advantage of the bounding boxes to compute the exclusive contributions of points that are then used for the hypervolume computation. The asymptotic time does not seem to be very good, but experimental and theoretical evidence shows that is the fastest exact algorithm for high dimensions (Bringmann and Friedrich, 2013), (Priester et al., 2013). For dimensions bigger than 10, however, the computation time increases toward unfeasible values, even for very powerful computers. Typically, this forced researchers to scale down the problems before treating them. In Nowak et al. (2014), it is demonstrated that for the dimensions bigger than 10 the best approximation algorithm in terms of accuracy of the approximation and run-time is the algorithm by Bringmann and Friedrich (Bringmann and Friedrich, 2009), which uses a Monte Carlo-like sampling method together with a racing approach to approximate the hypervolume computation.

To conclude, we will make use of these studies to establish the algorithm to be used for either computing the hypervolume exactly or for approximating it (depending on the objective function dimension). In particular, we will use the following algorithms:

- For $d = 2 \rightarrow$ dimension-sweep exact algorithm.
- For $d = 3 \rightarrow$ Beume exact algorithm.
- For $4 \leq d \leq 10 \rightarrow$ Walking Fish Group exact algorithm.
- For $d > 10$ Bringmann and Friedrich approximation algorithm.

5.2.2. P-Distance Metric

This metric was first introduced in Märtens and Izzo (2013) and it is useful for test-problems (such as ZDT and DTLZ) that are constructed using a distance function that is minimized by all the Pareto-optimal solutions. The authors leverage this aspect for defining a convergence metric that can be useful for evaluating the performance of a certain found final Pareto front. If we call $g(\mathbf{x})$ the distance function of a certain problem, P a set of solutions (which we want to evaluate the performance of) and $\mathbf{x}^*$ any Pareto-optimal decision vector, we then define:

$$\Gamma(P) = \frac{1}{|P|}\sum_{\mathbf{x}\in P}\left(g(\mathbf{x}) - g(\mathbf{x}^*)\right) \tag{5.14}$$

where $|P|$. The lower the $\Gamma(P)$ value, the better the solutions of P are (i.e., meaning that they will be closer to the real Pareto-optimal front). This metric has also the interesting property of reaching a value of zero, in case that all the set of solutions P belong to the real Pareto-optimal front. Also, this metric has the main advantage of not relying on any established reference set. It has, however, the drawback that it can only be applied to mathematical problems that make use of the distance function in their constructions.

5.3. Ant Colony Optimization

Although a single and multi-objective version of an Ant Colony Optimizer (ACO) has been implemented in this thesis study, we have decided to group this class of algorithms in a separate chapter, due to the high relevance of this class of algorithms in this thesis study. Indeed, both the single and multi-objective extension of ACO used for this research have been both theoretically and practically implemented in this thesis study for the first time. In particular, while the single-objective extension is strongly inspired by another ACO first introduced in Schlüter (2012), although several modifications have been introduced in that formulation; its multi-objective counterpart is a completely new idea that merges some multi-objective concepts (e.g. hypervolume metric and nondominated sorting strategy) into the ant colony optimization framework.

In this section, we will first introduce the general framework of ant colony optimization for single-objective problems and some basics definition. We will finally also discuss the multi-objective extensions. The structure of this section is shown in Figure 5.3.

[3] `http://lopez-ibanez.eu/hypervolume`, date of access: August 2019

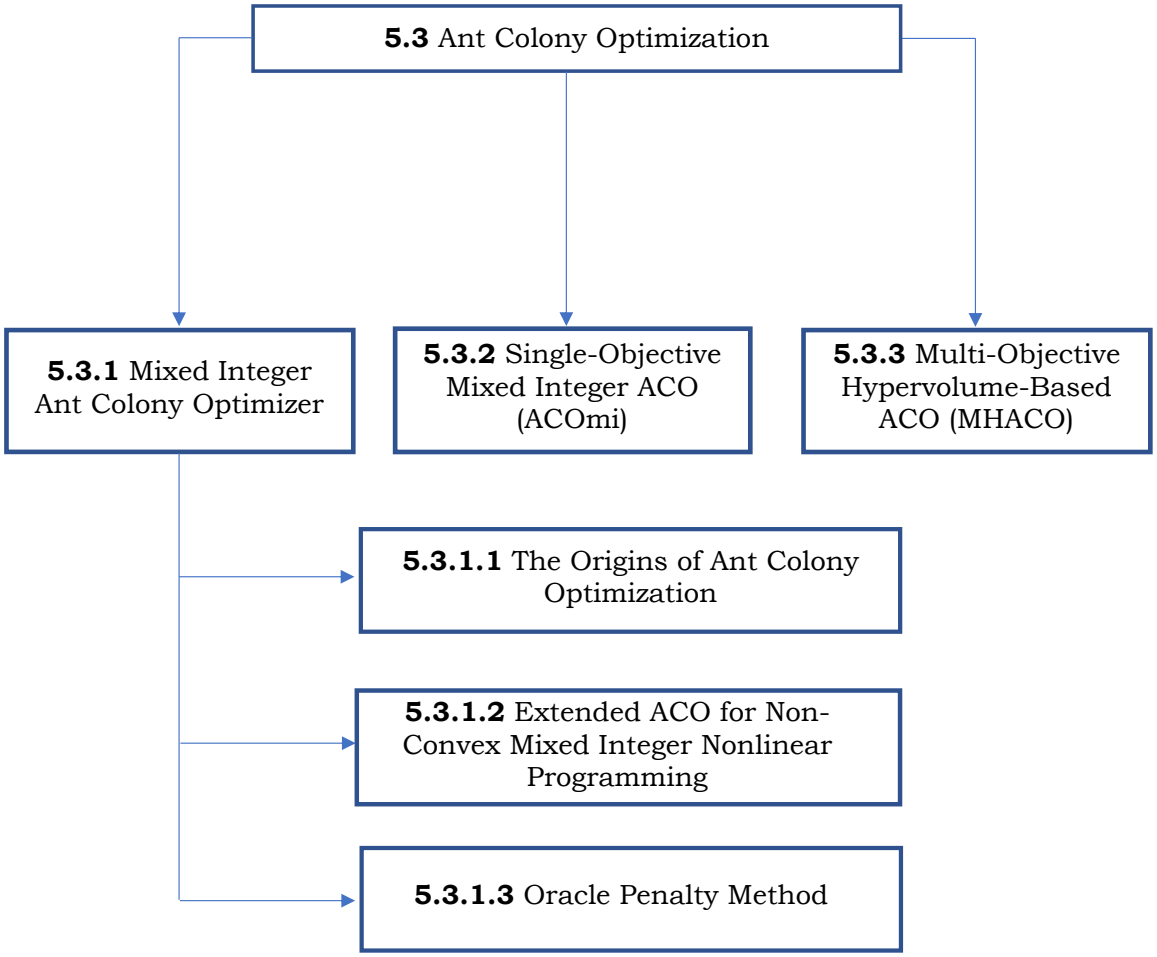


Figure 5.3: Block diagram of Section 5.3.

5.3.1. Mixed Integer Ant Colony Optimizer

Before diving into the implemented algorithm, we will investigate the working principle of the ACO. In particular, we will explore an ACO with an extension to mixed integer search domains, with an Oracle penalty method to handle the constraints. In this thesis study, the ACO principle has been fundamental since we constructed two different algorithms from this principle (one for single-objective and one for multi-objective). For a better understanding of this algorithm, it is first necessary to introduce the basic working principle of ACO. This algorithm was introduced for the first time by Marco Dorigo in his PhD thesis at Politecnico di Milano (Dorigo, 1992). Originally, this algorithm was only used for combinatorial problems, but soon it has become interesting and useful for many other problems and branches (trajectory optimization is one of these). In its rudimental form, this algorithm exploits the behavior of real ant colonies to create artificial ants, which can find the shortest path between two points. We will thus first describe the ant colony optimization origins and the biological mechanism behind it, and we will then discuss into detail one type of ACO: the mixed integer search domain, which is particularly used for trajectory optimization.

5.3.1.1 The Origins of Ant Colony Optimization

ACO is inspired by the natural mechanism through which real ant colonies forage food. These ants explore several paths from their nest to seek food, and they leave pheromone trails along their way for enabling other ants in the colony to follow their path. In this way, the shortest path is soon discovered by the ants and followed by the entire colony. The amount of pheromone released is proportional to the quantity and quality of food found:

this makes the shortest path to the 'best' source of food soon to prevail on the others. This indirect communication between ants through pheromone is called stigmergy and is the basic principle that inspired Dorigo and his colleagues.

As we already pointed out, the first algorithm was mainly developed for combinatorial problems (e.g. the traveling salesman problem), and it was presented as the Ant System (AS). This algorithm can be described in some basic passages.

First of all, the model consists of a graph $G = (V, E)$, where V are two nodes and E two paths. Therefore, v_s represents the nest of the ants, and v_d represents the food source, whereas e_1 and e_2 symbolize two paths with two different lengths. The path e_1 has a length l_1, whereas the path e_2 has a length l_2, and we now assume that $l_2 > l_1$. Hence, e_1 is the shortest path between the nest and the source food, while e_2 is the longest. Besides, we introduce an artificial pheromone value τ_i (where the index i refers to the i^{th} path: in this case i=1,2). This value indicates the strength of the pheromone in the corresponding path. Now, introducing n_a artificial ants, we can describe the behavior of each ant as follows: starting from the nest, each ant chooses the path to follow (in this case between path e_1 and e_2), with a probability:

$$p_i = \frac{\tau_i}{\sum_{i=1}^{i=2} \tau_i} \qquad i = 1,2 \tag{5.15}$$

Therefore, if $p_1 > p_2$ the ants will more probably follow e_1, and vice versa.

When returning back to the nest, the ants follow the same path, and they change the pheromone value associated with that trait. In particular, having chosen the path e_i, the ant then changes the pheromone associated with that path from τ_i to $\tau_i + \frac{Q}{l_k}$, where the positive constant Q is a parameter of the model.

This means that the amount of pheromone added to the path is:

$$\Delta\tau_i = \frac{Q}{l_i} \tag{5.16}$$

This will cause the shortest path to having the highest pheromone increase, and this will soon lead all the ants to choose the shortest path because its probability will increase during the evolution of the algorithm (as it can be seen from Equations (5.15) and (5.16)). As it is, this model is, however, flawed. Indeed, if a short path will not be discovered immediately, but only after a while, the ants will still follow the longer path (that was previously the shortest), due to the amount of pheromone accrued in the meantime. For avoiding this problem, the amount of pheromone released is subject to evaporation over time. This behavior is simulated by changing the pheromone value (τ_i) as follows:

$$\tau_i \Rightarrow (1 - \rho) \cdot \tau_i \qquad i = 1,2 \tag{5.17}$$

where $\rho \in (0, 1]$ is a parameter that controls the evaporation of pheromone.

A simplified ACO algorithm version can be summarized with the following steps:

1. Initialize all of the arcs with a uniform pheromone level.
2. Randomly places ants on the grid.
3. Progress forward, tracing a path by probabilistically selecting the next node based on the relative pheromone levels of surrounding nodes.
4. Erase loops in the path traced.
5. Retrace steps.
6. Globally updates the trails implementing pheromone evaporation according to the value of the ρ parameter.
7. Apply the pheromone increase to the retraced trails ($\Delta\tau_i$).

This algorithm is still subject to some fallacies that are typically solved depending on the specific problem with several specific variants of the ACO algorithm. We will now introduce an ACO algorithm with an extension to mixed integer search domains.

5.3.1.2 Extended ACO for Non-Convex Mixed Integer Nonlinear Programming

Here we will discuss a recent extension of the ACO to mixed integer search domains. This extension was introduced in Schlüter et al. (2009), and is extensively discussed in Schlüter (2012). Also, it is implemented in the software MIDACO for trajectory optimization purposes, and it has been tested on challenging space optimization problems at the European Space Agency and Astrium (Airbus Group) (Schlüter, 2010), (Schlüter et al., 2013).

Before going into details in the algorithm implementation and characteristics, it is first necessary to introduce some mathematical general definitions that link ACO to MINLP (mixed integer non linear programming).

First of all, a feasible set K of a mixed integer optimization problem is introduced, based on equality and inequality constraints.

The mathematical formulation of the general multi-objective MINLP can be written as:

$$\min\big(\mathbf{F}(\mathbf{X})\big) = (f_1(\mathbf{X}), \dots, f_O(\mathbf{X}))^T \tag{5.18}$$

subject to:

$$g_i(\mathbf{X}) = 0 \qquad i = 1, \dots, m_e \tag{5.19}$$

$$g_i(\mathbf{X}) \leq 0 \qquad i = m_e + 1, \dots, m \tag{5.20}$$

$$\mathbf{X}_l \leq \mathbf{X} \leq \mathbf{X}_u \tag{5.21}$$

Thus, the objective is to minimize all the objective functions while satisfying m_e equality constraints and $m-(m_e+1)$ inequality constraints, where the decision variables $\mathbf{X}$ are included between a lower ($\mathbf{X}_l$) and upper ($\mathbf{X}_u$) boundaries. The vector $\mathbf{X}$ is constituted of two different vectors ($\mathbf{X} = (\mathbf{x}^T, \mathbf{y}^T)^T$): a vector of discrete variables of dimension n_{int} ($\mathbf{x} \in \mathbb{R}^{n_{int}}$), and a vector of continuous variables of dimension n_{con} ($\mathbf{y} \in \mathbb{R}^{n_{con}}$).

For introducing the mechanism through which the offspring is generated in the ACO framework, it is useful to introduce some mathematical concepts:

Definition 1

A function $P{:}\mathbb{R} \to \mathbb{R}_0^+$ with:

$$\int_{-\infty}^{+\infty} P(t)dt = 1 \tag{5.22}$$

is called a continuous Probability Density Function (cPDF). Whereas a function $Q{:}\mathbb{Z} \to \mathbb{R}_0^+$ with:

$$\sum_{d=-\infty}^{+\infty} Q(d) = 1 \tag{5.23}$$

is called a discrete Probability Density Function (dPDF).

The ant colony optimization, similarly to other evolutionary algorithms, uses individual ants to explore the search space and applies the concept of 'survival of the fittest' to generate offspring from the current solution. The individual ants are evaluated depending on their constraints violations and objective function values: these two aspects are typically combined in a single penalty function. In Section 5.3.1.3, we will then discuss into details the oracle penalty method employed in this research.

After every generation, the individuals are ranked depending on penalty function values, and the best individuals are chosen to generate future generations stochastically. In this way, the algorithm will likely improve the individuals in future generations. The following definitions are useful for formalizing the meaning of individual, generation, and fitness, in the context of ACO:

Definition 2

An element $(\mathbf{x},\mathbf{y}) \in \mathbb{R}^{n_{con}} \times \mathbb{Z}^{n_{int}}$ is called individual (or ant). Any individual $(\mathbf{x},\mathbf{y})$ is called feasible if and only if $(\mathbf{x},\mathbf{y}) \in K$. Otherwise, is called unfeasible. A set $\mathbf{G} := \{(\mathbf{x},\mathbf{y})^1,$

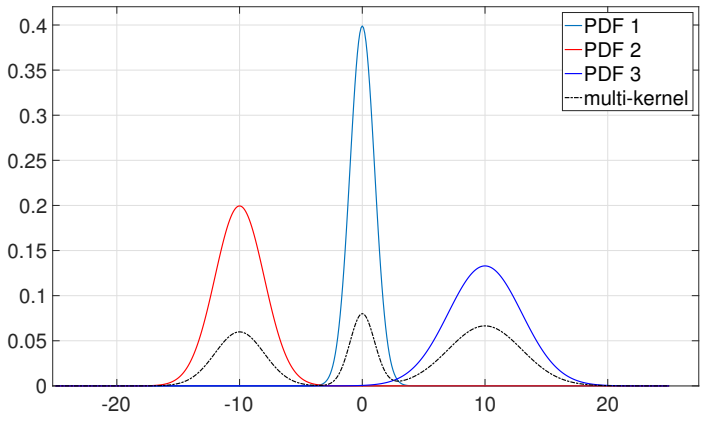


Figure 5.4: Three individual Gauss PDFs and their multi-kernel PDF

$(\mathbf{x},\mathbf{y})^2,\ldots,(\mathbf{x},\mathbf{y})^v\}$ is called a generation of size $v \in \mathbb{N}$, if all the components x_i^l (for $i = 1,\ldots,n_{con}$ and $l = 1,\ldots,v$) are samples of a set of n_{con} cPDF $P^{i=1,\ldots,n_{con}}$ and all the components y_j^l (for $j = 1,\ldots,n_{int}$ and $l = 1,\ldots,v$) are samples of a set of n_{int} dPDF $Q^{j=1,\ldots,n_{int}}$. These individuals belonging to $\mathbf{G}$ are not necessarily feasible, and their feasibility has to be checked at a later phase when evaluating the various ants.

Moreover, the archive in which the best individuals are stored throughout the various generations, as well as the evolutionary operator used for creating new individuals starting from the solution archive, have to be defined. This is done in the following definition:

Definition 3

A set $\mathbf{S} := \{(\mathbf{x},\mathbf{y})^1,\ldots,(\mathbf{x},\mathbf{y})^K\}$ is called a solution archive of size K if all individuals of $\mathbf{S}$ are ordered regarding their fitness (such that the first one is the best and the last one is the worst).

A function $\mathcal{E} : \left(\mathbb{R}^{n_{con}} \times \mathbb{Z}^{n_{int}}\right)^K \to (\mathbb{R}^{n_{con}} \times \mathbb{Z}^{n_{int}})^v$ that creates a generation of v individuals based on K individuals of a solution archive $\mathbf{S}$ is called an evolutionary operator.

Note that the solution archive $\mathbf{S}$ contains K n-dimensional solution vectors ($\mathbf{s}^1 = (\mathbf{x},\mathbf{y})^1,\ldots,\mathbf{s}^K = (\mathbf{x},\mathbf{y})^K$), their corresponding K objective function values, constraints violations and the penalty function values. In general, K and v are two parameters independent from each other, but it always holds $K < v$.

We will now introduce a specific evolutionary operator that is used for producing the offspring. The ACO operator used is based on n multi-kernel Gauss probability density functions $\mathcal{G}^{h=1,\ldots,n}$, where $n = n_{con} + n_{int}$. One multi-kernel PDF $\mathcal{G}^h$ is a weighted sum over K individual Gauss PDF. It can thus be written as:

$$\mathcal{G}^h(t,\omega,\mu,\sigma) = \sum_{k=1}^{K} \omega_k^h \frac{1}{\sigma^h\sqrt{2\pi}} e^{-\frac{(t-\mu_k^h)^2}{2(\sigma^h)^2}} \qquad h = 1,\ldots,n_{con}+n_{int} \tag{5.24}$$

where the term:

$$\frac{1}{\sigma^h\sqrt{2\pi}} e^{-\frac{(t-\mu_k^h)^2}{2\sigma^{h\,2}}} \tag{5.25}$$

represents a single Gauss PDF. In Figure 5.4 an example of three PDF and their multi-kernel PDF is shown.

The version of the multi-kernel PDF has a different implementation for the integer domain: indeed, for that domain a discretized version is applied, which accumulates the probability

given by $\mathcal{G}^h(t,\omega,\mu,\sigma)$ around an integer d in the interval: $[d-\frac{1}{2},d+\frac{1}{2}]$. Hence, we can define the already discussed cPDFs and dPDF (there will be one for each continuous and discrete variable) as:

$$P^i(t)=\mathcal{G}^i(t,\omega,\mu,\sigma) \qquad i=1,\dots,n_{con} \tag{5.26}$$

$$Q^i(d)=\int_{d-1/2}^{d+1/2}\mathcal{G}^{n_{con}+j}(t,\omega,\mu,\sigma)dt \qquad j=1,\dots,n_{int} \tag{5.27}$$

From Equations (5.26) and (5.27), it is possible to see that each PDF is identified by three parameters: $\{\omega_k^h,\mu_k^h,\sigma^h\}$. Hence, these three parameters are responsible for the evolution of the search process of the ACO and they are of extreme importance. Indeed, the search process is performed through $P^i(t)$ and $Q^j(d)$ for each variable, and these parameters can be seen as the pheromone values (in analogy with the biological process). The parameters ω_k^h are only dependent on the size K of the solution archive **S**, and they behave as weights for each PDF in the multi-kernel PDF $\mathcal{G}^h(t,\omega,\mu,\sigma)$. Indeed, as we already pointed out, the archive **S** is ordered in terms of performance (the best ones are the first). Therefore,the weights are calculated as:

$$\omega_k^h=\frac{K-k+1}{\sum_{j=1}^{K}j} \tag{5.28}$$

Thus, we have as many weights as the solutions in the archive (which are K in total), and these weights are useful for computing the pheromones for each of the continuous and discrete variables in the problem. It is important to notice that a property of the weights is that:

$$\sum_{k=1}^{K}\omega_k^h=1 \tag{5.29}$$

The parameter μ_k^h are the means of each PDF within the multi-kernel PDF. They can be computed from the ants: $(\mathbf{x},\mathbf{y})^{k=1,\dots,K}$ contained in **S** in the following way:

$$\mu_k^h=\begin{cases}x_h^k & \text{if } h=1,\dots,n_{con}\\ y_{h-n_{con}}^k & \text{if } h=n_{con}+1,\dots,n_{int}\end{cases} \tag{5.30}$$

The standard deviations σ^h are related to each individual PDF within the multi-kernel PDF. The method to calculate them is based on the concept of maximum distance (D_{max}) and minimum distance (D_{min}) between each variable dimension h, in all the solution archive $(\mathbf{x},\mathbf{y})^{k=1,\dots,K}$ (e.g. for h=1 then $x_1^1,\dots,x_1^K$ are taken into consideration). These distances are computed as:

$$D_{min}^h=\begin{cases}min\{|x_h^p-x_h^q|:p,q\in\mathbb{N},p\neq q\le K\} & \text{if } h=1,\dots,n_{con}\\ min\{|y_{h-n_{con}}^p-y_{h-n_{con}}^q|:p,q\in\mathbb{N},p\neq q\le K\} & \text{if } h=n_{con}+1,\dots,n_{int}+n_{con}\end{cases}$$

$$D_{max}^h=\begin{cases}max\{|x_h^p-x_h^q|:p,q\in\mathbb{N},p\neq q\le K\} & \text{if } h=1,\dots,n_{con}\\ max\{|y_{h-n_{con}}^p-y_{h-n_{con}}^q|:p,q\in\mathbb{N},p\neq q\le K\} & \text{if } h=n_{con}+1,\dots,n_{int}+n_{con}\end{cases}$$

The standard deviations σ^h are then computed as:

$$\sigma^h=\begin{cases}\dfrac{D_{max}^h-D_{min}^h}{\#gen} & \text{if } h=1,\dots,n_{con}\\ max\left\{\dfrac{D_{max}^h-D_{min}^h}{\#gen},\dfrac{1}{\#gen},\left(1-\dfrac{1}{\sqrt{n_{int}}}\right)/2\right\} & \text{if } h=n_{con}+1,\dots,n_{con}+n_{int}\end{cases} \tag{5.31}$$

where $\#gen$ is the number of generations reached.

In the case that the search space is discrete, the standard deviation will never be smaller than $max\left\{\frac{1}{\#gen}, \left(1-\frac{1}{\sqrt{n_{int}}}\right)/2\right\}$, even if $D_{max}^{h} - D_{min}^{h} = 0$. This situation may happen in the case that all the solution vectors in $\mathbf{S}$ have converged to the same integer. This condition is enforced to sample the multi-kernel PDF with a sufficient standard deviation, which allows exploration of other integers, nearby the found ones. Indeed, if this procedure was not implemented the algorithm would have been less flexible, and would have more easily got stuck.

To summarize, the solution saved in the solution archive will generate new solutions through the multi-kernel PDF. This is done by considering the importance of the solutions in the solution archive (i.e., for a solution $\mathbf{s}_k$, the lower the index k the better the solution in terms of the objective function and penalty function) and the values of the several solution vector components of each member of the archive. This generates a triplet of parameters (weight, mean and standard deviation), which are then used for generating new ants from the solution archive. These new solutions will be evaluated and compared with the solutions in the solution archive. If the new ants perform better than the one stored in $\mathbf{S}$, then the solution vectors in the solution archive are replaced starting from the tail of the vector (e.g. solution $\mathbf{s}_g$ is replaced before $\mathbf{s}_f$, if $g > f$). This replacement process corresponds to the biological process of pheromone evaporation.

In particular, the incremental construction of new individuals (i.e., ants) is done by first choosing the mean μ_k^h for every h. This choice is not performed completely randomly, but it is carried out depending on the weights ω_k^h: this causes, for instance, the mean μ_1^h to be more likely chosen, whereas μ_K^h has the lowest probability to be selected. Furthermore, a random number is produced starting from the selected mean, using the related standard deviation (σ^h). By doing this for all the dimensions of the ants, a new ant is created, and this allows another objective function and penalty function evaluation to be performed, which also allows us to rank the solution and establish whether it has to replace another solution in the solution archive or not. This is carried out throughout all the generations.

An aspect to be noted is that Equation (5.24) only defines the multi-kernel PDF $\mathcal{G}^h(t, \omega, \mu, \sigma)$, however, for a numerical implementation of the algorithm, a mathematical technique is necessary to actually produce the stochastic sample from the given PDF.

To summarize, the general framework of ACO for MINLP can be explained through the following steps:

1. Generate the first generation $\mathbf{G}^1$ of ants: the first generation will have v individuals: $(\mathbf{x},\mathbf{y})^1, \ldots, (\mathbf{x},\mathbf{y})^v$ based on a uniform cPDF for the continuous variables $\mathbf{x}$ and on a uniform dPDF for discrete variables $\mathbf{y}$.
2. Repeat the steps from 3 to 5 until the stopping criteria are met.
3. Select K best individuals from the current generation, based on their fitness value (comprehensive of the objective function and penalty function) and save them as solution archive $\mathbf{S}$.
4. Apply the evolutionary operator (described above) on the solution archive to produce the next generation of v ants.
5. Introduce all the ants created to the solution archive and check whether they perform better than the lowest ranked individual in $\mathbf{S}$. If this is the case, then discard those individuals from the archive that have lower fitness and introduce the better new ones in $\mathbf{S}$.

5.3.1.3 Oracle Penalty Method

The oracle penalty method was first introduced in Schlüter (2010). As any other penalty method, its objective is to transform an originally constrained problem into an unconstrained one, by inserting penalty functions in the original objective function. In this research, this penalty method is employed for the SO ACO.

In general, the oracle penalty method is especially meant for stochastic metaheuristics optimization problems. Furthermore, its strength is that it only requires one parameter (called the 'oracle') to be tuned: making the penalty functions easy to understand and handle.

We will first introduce the basic oracle penalty function, and then we will introduce some modifications and adjustments for showing some extensions to the basic method.

The basic idea is to transform the objective function $f(\mathbf{X})$ into an equality constraint to be added to the others in the form of:

$$g_0(\mathbf{X}) = f(\mathbf{X}) - \Omega = 0 \tag{5.32}$$

where Ω is a parameter called oracle. In this transformed problem, the objective function results to be redundant and a new objective function ($\overline{f}(\mathbf{X})$) can be declared as constant zero function. Hence, the transformed problem can be expressed as:

$$\begin{aligned} &\text{minimize } \overline{f}(\mathbf{X}) \equiv 0 \\ &\text{subject to}: g_0(\mathbf{X}) = f(\mathbf{X}) - \Omega = 0 \ \omega \in \mathbb{R} \\ &g_i(\mathbf{X}) = 0 \qquad\qquad i = 1, \dots, m_e \in \mathbb{N} \\ &g_i(\mathbf{X}) \leq 0 \qquad\qquad i = m_e + 1, \dots, m \in \mathbb{N} \end{aligned} \tag{5.33}$$

Assuming now that $\mathbf{X}^*$ is the solution that defines the global optimum of the MINLP, then an oracle parameter of the kind: $\Omega = f(\mathbf{X}^*)$ would imply that a solution of Problem 5.33 is directly the global optimal solution of the MINLP. The algorithm will hence start to equally search to minimize $g_0(\mathbf{X})$. However, we would like the algorithm not only to minimize the objective function, but also to maintain the constraints violated as little as possible. For doing this, the concept of residual function is used. A residual function measures the violation of the constraints by applying a norm function over all the m constraint violations of the MINLP problem (i.e., if there is no constraint violations, the residual function will be zero). In our research, we have decided to employ an l^2 norm as residual function. This means that the residual function would have the following definition:

$$l^2 = res(\mathbf{X}) = \sqrt{\sum_{i=1}^{m_e} |g_i(\mathbf{X})|^2 + \sum_{i=m_e+1}^{m} \min\{0, g_i(\mathbf{X})\}^2} \tag{5.34}$$

The penalty function defined by the basic version of the oracle penalty method consists in a function that uses the information on the residuals and on the objective function, to establish the penalty. In particular, the penalty function ($p(\mathbf{X})$) can be defined as:

$$p(\mathbf{X}) = \mathcal{A}_\Omega(f(\mathbf{X}), res(\mathbf{X})) \cdot |f(\mathbf{X}) - \Omega| + (1 - \mathcal{A}_\Omega(f(\mathbf{X}), res(\mathbf{X}))) \cdot res(\mathbf{X}) \tag{5.35}$$

where $\mathcal{A}_\Omega(f(\mathbf{X}), res(\mathbf{X}))$ is defined as:

$$\mathcal{A}_\Omega(f(\mathbf{X}), res(\mathbf{X})) = \begin{cases} 1 - \dfrac{1}{2\sqrt{\dfrac{|f(\mathbf{X}) - \Omega|}{res(\mathbf{X})}}} & \text{if } res(\mathbf{X}) \leq |f(\mathbf{X}) - \Omega| \\ \dfrac{1}{2}\sqrt{\dfrac{|f(\mathbf{X}) - \Omega|}{res(\mathbf{X})}} & \text{if } res(\mathbf{X}) > |f(\mathbf{X}) - \Omega| \end{cases}$$

For conciseness, from now on we will define α as: $\alpha = \mathcal{A}_\Omega(f(\mathbf{X}), res(\mathbf{X}))$.

It is clear that α is always limited between 0 and 1, and it balances the penalty function depending on the value of both $|f(\mathbf{X}) - \Omega|$ and $res(\mathbf{X})$. Indeed, if the residuals ($res(\mathbf{X})$) are 'less violated' it results that $0.5 \leq \alpha \leq 1$, and the penalty function will focus on penalizing the transformed objective function. On the other hand, if $|f(\mathbf{X}) - \Omega|$ is smaller than $res(\mathbf{X})$, then $0 \leq \alpha \leq 0.5$ and this will cause the penalty function to penalize more the residuals, increasing their weight.

A crucial aspect that constitutes a substantial pitfall of the basic oracle penalty function, is that the penalty function, and thus the search for the optimum, is strongly related to the value of Ω. This means that we need information on the global optimal objective value to make this strategy effective. In Schlüter (2010) an extension is mathematically explained and introduced to solve this issue. This extension led to the definition of an extended oracle penalty function.

This penalty function is of the form:

$$p(\mathbf{X}) = \begin{cases} \alpha \cdot |f(\mathbf{X}) - \Omega| + (1-\alpha) \cdot res(\mathbf{X}) & \text{if } f(\mathbf{X}) > \Omega \text{ or } res(\mathbf{X}) > 0 \\ -|f(\mathbf{X}) - \Omega| & \text{if } f(\mathbf{X}) \leq \Omega \text{ and } res(\mathbf{X}) = 0 \end{cases} \tag{5.36}$$

where α is computed as:

$$\alpha = \begin{cases} \dfrac{|f(\mathbf{X}) - \Omega| \cdot \dfrac{6\sqrt{3}-2}{6\sqrt{3}} - res(\mathbf{X})}{|f(\mathbf{X}) - \Omega| - res(\mathbf{X})} & \text{if } f(\mathbf{X}) > \Omega \text{ and } res(\mathbf{X}) < \dfrac{|f(\mathbf{X}) - \Omega|}{3} \\ 1 - \dfrac{1}{2\sqrt{\dfrac{|f(\mathbf{X}) - \Omega|}{res(\mathbf{X})}}} & \text{if } f(\mathbf{X}) > \Omega \text{ and } \dfrac{|f(\mathbf{X}) - \Omega|}{3} \leq res(\mathbf{X}) \leq |f(\mathbf{X}) - \Omega| \\ \dfrac{1}{2}\sqrt{\dfrac{|f(\mathbf{X}) - \Omega|}{res(\mathbf{X})}} & \text{if } f(\mathbf{X}) > \Omega \text{ and } res(\mathbf{X}) > |f(\mathbf{X}) - \Omega| \\ 0 & \text{if } f(\mathbf{X}) \leq \Omega \end{cases} \tag{5.37}$$

Now that the penalty function is extended, it is necessary to explain how the oracle parameter, Ω, has to be handled during the optimization procedure, to steer the algorithm towards the search of the global optimum. The oracle parameter Ω is initialized with a very big value (i.e., $\Omega^1 >> f(\mathbf{X})$, for any $\mathbf{X}$). This will cause the search strategy to focus only on the residual for the first optimization runs, up to the point that a feasible solution is found. Indeed, after the first optimization run, the oracle parameter follows the following update strategy:

$$\Omega^i = \begin{cases} f^{i-1} & \text{if } f^{i-1} < \Omega^{i-1} \text{ and } res^{i-1} \leq \overline{res} \\ \Omega^{i-1} & \text{else} \end{cases}$$

where i refers to the current generation number and $\overline{res}$ is a user-defined tolerance that defines the residuals feasibility threshold. Hence, if no feasible solution has been found so far (i.e., solutions such that $res(\mathbf{X}) \leq \overline{res}$) the oracle parameter will be maintained large enough. On the other hand, if a feasible solution has been found, then the oracle parameter is updated with the latest feasible solution, which has the lowest objective function value. It has to be noted that if a reasonable value for a feasible solution is known, then one can initialize Ω^1 using a value reasonably lower than the currently known objective function value.

From our definitions, it is clear that the oracle penalty method only works for single-objective optimization. For this reason, this method was only used for the SO implementation of ACO and another sorting strategy was employed to rank the solutions in the MO extension. In the next two sections, we will discuss both the SO and MO extensions, as well as their input parameters and working principles.

5.3.2. Single-Objective Mixed Integer ACO (ACOmi)

This algorithm is constructed based on the principles introduced in Section 5.3.1. In particular, the same oracle penalty method and pheromone values as explained in that section are used. However, some parts have been modified for improving the algorithm: indeed, when implementing the basic version explained above, it was found that premature convergence was achieved, thus causing the algorithm to get stuck even for very simple SO test problems.

For this reason, the following modifications were added to the algorithm:

1. Using Equation (5.31), the standard deviations were soon driven to zero, thus causing the algorithm not to generate offspring far enough from previous individuals, and to get stuck soon during the evolution process. For solving this, a new user-defined parameters has been introduced: *NGenMark*. Basically, instead of #*gen*, in Equation (5.31), *GenMark* was used so that the equation becomes:

$$\sigma^h = \begin{cases} \dfrac{D^h_{max} - D^h_{min}}{GenMark} & \text{if } h = 1, \dots, n_{con} \\ max\left\{ \dfrac{D^h_{max} - D^h_{min}}{GenMark}, \dfrac{1}{GenMark}, \left(1 - \dfrac{1}{\sqrt{n_{int}}}\right)/2 \right\} & \text{if } h = n_{con}+1, \dots, n_{con}+n_{int} \end{cases} \tag{5.38}$$

   This parameter is the same as the number of generations. However, when the *GenMark* parameter reaches the *NGenMark* value, it is restarted again from 1 and it is increased again by one as the generation counter increases by one as well. For instance, let us assume that the author chooses *NGenMark* = 7, then, this means that the *GenMark* parameter will be increased until 7, when the population is evolved until the 7th generation. Afterwards, the *GenMark* parameter is started again from 1 and then increased until 7, when the generation value reaches 14. This is repeated again with the same logic for the entire evolution process. In this way, (for continuous variables), as long as the $D^h_{max} - D^h_{min}$ parameter is different than zero, σ^h is ensured not to reach very small values if the user decides so (by selecting a low value for the *NGenMark* parameter). Of course, there is also the risk that *NGenMark* is chosen so small that the consequent standard deviations are too large and the algorithm thus struggles to converge. For this reason, this parameter often requires tweaking and tuning.

   In particular, two parameters are used to allow the user to monitor the spread of the individuals stored in the solution archive and thus establish how to set *NGenMark*. This is strictly related to the standard deviation values since very spread individuals might require a lower standard deviation value (hence a higher *NGenMark*), whereas very cluttered individuals might require a higher standard deviation value (hence a lower *NGenMark*), which may ensure a wider search in the variables' domain. For allowing the user to determine this more easily, the flatness in the variables (i.e., *dx*) and in the penalties (i.e., *dp*) of the solution archive are stored for each generation. These are defined as:

$$dx = \sum_{i=1}^{n} \left(|x_{w,i} - x_{b,i}| \right) \tag{5.39}$$

$$dp = |p_w - p_b| \tag{5.40}$$

   where p_w is the penalty function value of the last (i.e., the worst) individual in the solution archive, whereas p_b of the first (i.e., the best), both computed as defined in Equation (5.36). Also, n is the variables' dimension, i is the i^{th} variables' component, $x_{b,i}$ indicates the components of the first decision vector (i.e., the best in the solution archive) and $x_{w,i}$ the components of the last decision vector (i.e., the worst in the solution archive).

2. In Section 5.3.1, we have discussed the multi-kernel gaussian distribution properties and its use for the ant colony optimization algorithm. However, in practice, we have decided to implement a different strategy for generating these individuals. This strategy uses the same concept as the multi-kernel distribution, but employing a different method. Also, it allows the user to establish whether to focus more on the first individuals of the archive, for producing the offspring. This strategy was first introduced in Bernardo Jr and Naval Jr (2010).

   For presenting its implementation, it is first necessary to redefine the weights. Indeed, the weights used in the multi-kernel gaussian function, instead of being defined as shown in Equation (5.28), are defined as follows:

$$\omega_k^h = \frac{1}{qK\sqrt{2\pi}} e^{-\frac{(k-1)^2}{2q^2K^2}} \tag{5.41}$$

   where k refers to the k^{th} individual in the solution archive, whose members range from 1 to K (which represents the solution archive size). Also, q is a new user-defined parameter that substantially regulates the weights' definition. Indeed, if q is set to high values, then the resulting weights will be of similar magnitude and every individual in the solution archive will have a similar probability of being selected. Whereas if q is chosen to be very small, then the algorithm will focus more on the very first individuals of the solution archive (which are considered to be the best). Besides, Equation (5.29) does not hold anymore for the redefined weights.

   As a consequence, the sampling process is not done anymore by directly using the multi-kernel Gaussian distribution function shown in Equation (5.24), but it is performed in a more practical way. This is carried out by only selecting one of the Gaussian functions of the multi-kernel PDF, whose selection is done by taking into account the probability p_k^h of choosing the k^{th} Gaussian function, for each variables' component h. This probability is computed as:

$$p_k^h = \frac{\omega_k^h}{\sum_{k=1}^{K} \omega_k^h} \tag{5.42}$$

   These probability functions have the property of having their sum equal to one. Mathematically, this means that:

$$\sum_{k=1}^{K} p_k^h = 1 \tag{5.43}$$

   The process of sampling the chosen Gaussian distribution is then executed as follows: from the probability values the cumulative probability is extracted, and each individual is assigned with that cumulative distribution function value. Afterwards, a random number between 0 and 1 is generated using a uniformly distributed random generator, and the Gaussian function is chosen accordingly.

   For instance, let us assume that there are five individuals in the archive and that their variables' dimension is one (i.e., $h = 1$). Assuming that the user chooses q=1, the consequent weights (computed using Equation (5.41)) will be: $\omega_1 = 0.07979$, $\omega_2 = 0.07041$, $\omega_3 = 0.06389$, $\omega_4 = 0.060227$ and $\omega_5 = 0.057938$. Hence, the resulting probability function values (computed as explained in Equation (5.42)) will be: $p_1 = 0.24014$, $p_2 = 0.21192$, $p_3 = 0.19229$, $p_4 = 0.18127$ and $p_5 = 0.17438$. Also, their corresponding cumulative distribution functions are: $cdf_1 = 0.24014$, $cdf_2 = 0.45206$, $cdf_3 = 0.64435$, $cdf_4 = 0.82562$ and $cdf_5 = 1$. Thus, when the random number generator will generate a number num between 0 and 1, this will determine which probability density function will be chosen. In particular:

- if $0 < num < cdf_1$, then the Gaussian function of the first individual will be chosen.
- if $cdf_1 < num < cdf_2$, then the Gaussian function of the second individual will be chosen.
- if $cdf_2 < num < cdf_3$, then the Gaussian function of the third individual will be chosen.
- if $cdf_3 < num < cdf_4$, then the Gaussian function of the fourth individual will be chosen.
- if $cdf_4 < num < cdf_5$, then the Gaussian function of the fifth and last individual will be chosen.

For how the weights are defined, it is clear that the higher an individual is placed in the rank, the higher will be the probability that its Gaussian function is chosen. Also, the influence of the user-defined q parameter can be clearly seen: if this parameter is set to 0.5, the consequent cumulative distribution functions are: $cdf_1 = 0.38163$, $cdf_2 = 0.61310$, $cdf_3 = 0.76999$, $cdf_4 = 0.89389$ and $cdf_5 = 1$. As it can be seen, the first individuals will now be way more likely to be chosen (e.g. the first individual has now 38% probability to be chosen, whereas when $q = 1$, it only had 24% probability to be selected).

As one may point out, in the first phase of the algorithm optimization process, we would rather have more evenly distributed weights, so that all the individuals in the archive are used for the generation of the offspring. Whereas when the generations' number is high and the evolution process is mature, we would like to focus more on the very first individuals in the archive, to possibly achieve convergence. For taking this into account another user-defined parameter was introduced, called $threshold$. This parameter is an integer that can range between 1 and the generation number. When the generation's value reaches the $threshold$ value, then the q value is changed from the user-defined choice to a value of 0.01. This small value for q ensures that when the generation reaches the $threshold$ value, the algorithm will focus on the very first individuals of the archive, for generating the offspring. Also, if this behavior is not wanted by the users, they are also allowed to switch-off this dynamic behavior of q, and leave the user-defined choice of q for the entire evolution.

3. The last novelty of the algorithm is the introduction of an accuracy parameter (called acc) that makes sure that the individuals' penalty functions are diverse enough. Indeed, this parameter works as follows: before a new individual is accepted within the solution archive, it has not only to outperform at least the last individual, but it also has to have a penalty function value that is at least acc away from the outperformed individual in the solution archive. In this way, we make sure that the individuals in the solution archive (whenever possible) have penalty function values that are a acc value distant from each other. Of course, if the acc parameter is set to zero, then there are no distance constraints for adding new individuals in the archive.

5.3.3. Multi-Objective Hypervolume-Based ACO (MHACO)

This algorithm was developed both theoretically and practically in this thesis study and it represents a completely new strategy for MO optimization. Its idea comes from a fusion between three different concepts: nondominated sorting, hypervolume metric, and ant colony optimization. In particular, the nondominated sorting strategy and the hypervolume metric are used for ranking the individuals and their offspring, whereas the ant colony optimization idea is the key to understanding how new individuals are generated. Therefore, this algorithm represents a multi-objective extension of ACOmi, the ant colony optimizer introduced in Section 5.3.1.

The algorithm strategy can be summarized as follows:

1. The initial population of size NP is randomly generated within the box-bounds of the variables. Also, a solution archive of size $ker < NP$ is also generated using the individuals of the initial population.

2. If the generation number is higher than 1, then a merged list of $NP + ker$ individuals is created, where the individuals belonging to the solution archive and the offspring of the previous generation are collected.

3. The fast nondominated sorting strategy (the same as NSGA-II) is used for sorting the individuals into different nondomination ranks. This process is thoroughly discussed in Appendix C.

4. The hypervolume metric is computed for each individual at each nondomination level.

5. A sorting strategy is implemented: in case the individuals have different nondomination ranks, the lower nondomination level is preferred. In case that the nondomination level is the same, then the hypervolume metric is used to rank the individuals. In particular, the higher the hypervolume value, the better the individuals are considered and they are thus placed in a higher position in the ranking. Mathematically, this can be expressed using a hypervolume-comparison operator (i.e., $<_h$). According to this operator, an individual i is considered better than an individual j (and thus placed higher in the ranking) if the following is verified:

$$i <_n j \text{ if } (i_{\text{rank}} < j_{\text{rank}})$$
$$\text{or } ((i_{\text{rank}} = j_{\text{rank}}) \text{ and } (hv(i) > hv(j)))$$

   where the $hv(i)$ represents the hypervolume contribution of the i^{th} individual.

6. Based on the results of the sorting strategy, if some solutions of the solution archive are outperformed by new individuals (in terms of nondomination ranks or hypervolume values), then the solution archive is updated.

7. Once the ranking is done, it is possible to generate new offspring. This is done using the multi-kernel gaussian distribution already discussed in Section 5.3.1. In particular, the same strategy as ACOmi is used for handling and generating the pheromone values. The user is thus left to choose the three parameters (i.e., $threshold$, $NGenMark$ and q) to tune the pheromone values, and to thus steer the optimization.

8. The algorithm goes back to Step 2 and repeats itself. It is thus clear that at the first iteration, only the initial population will be ranked, whereas, for all the others, both the solution archive and the population are sorted.

One important aspect of this algorithm is the hypervolume value of each individual in the nondominated front. Indeed, as we have already pointed out in Section 5.2.1, this is strictly dependent on the choice of the reference point, which needs to always be dominated by all the other individuals. For achieving this, the reference point is generated by taking the maximum fitness, for each component, from all the individuals in the merged list. Then, its coordinate is increased by 1% in each coordinate (for ensuring that the hypervolume value is different than zero for all the individuals). The result is a point that is strictly dominated by all other points for each nondominated front and that is thus ideal for being used as a reference point.

If, for instance, we have three individuals in the merged list, in a problem with three dimensions, with the following fitness vectors: $\{100, 10, 8\}$, $\{15, 20, 7\}$ and $\{2, 18, 90\}$. Then, these individuals belong to the same nondominated front and the reference point will be created by increasing by 1% the coordinates of the following vector: $\{100, 20, 90\}$.

The algorithms used for the hypervolume computation are those indicated in Section 5.2.1: they will thus change depending on the dimension of the problem.

Overall, the algorithm has thus the following input values to be chosen and tuned (besides the generation and population sizes):

1. ker: the dimension of the solution archive (same use as ACOmi).

2. q (convergence speed parameter): same use as ACOmi.

3. $threshold$ (threshold parameter): same use as ACOmi.

4. $NGenMark$ (standard deviations convergence speed parameter): same use as ACOmi.

5. $evalstop$ (evaluation stopping criterion): if a positive integer is assigned here, the algorithm will count the runs without improvements (in terms of the ideal point), if this number will exceed the $evalstop$ value, the algorithm will be stopped and will return the evolved population until that moment. The ideal point is a fictitious point that has, in each component, the minimum value of the objective functions of the input points.

6. $focus$ (focus parameter): this parameter is used for making the search towards the minimum greedier and more likely to get stuck in local minima (same use as ACOmi).

As we can observe, by designing from scratch the MO extension for the ACO, we have also managed to keep the number of input parameters to only six and to remove the oracle parameter. Also, among these six, we have seen that only $threshold$ and $NGenMark$ are critical to be tweaked and tuned (especially when the population and generation sizes are changed). While the other parameters are typically set to certain values: in particular, the ker is typically chosen to be equal to the population size, q is set to 1, $evalstop$ and $focus$ are typically not activated (unless the user has a specific time or convergence constraints). Of course, this is just a general rule and may not be applicable to all the problems: depending on each problem, the user should thus pay attention to all these parameters and decide whether to tune them, in case that the optimization results are not satisfactory.

Furthermore, since this algorithm represents an absolute novelty w.r.t. literature, it requires further study to understand how to set these parameters depending on which type of optimization problem is handled.

Concerning the computation speed, MHACO has the possibility to parallelize the fitness computation of the current individuals on multiple cores, thus making the optimization process considerably faster for difficult problems (especially for machines with many cores). However, similarly to ACOmi, for employing this strategy, it is required that the problem is thread-safe.

5.4. Problem Definition

In this section, we will describe the characteristics of the optimization problems studied in this research. As we have already mentioned, the scope of this study is to design a solar sailing mission for a journey in a polar orbit around the Sun. This is done by approaching the problem with different strategies: first with a single-objective strategy and then with a multi-objective one. Furthermore, in the SO framework, the problem is approached in two different ways: first the geocentric and heliocentric phases are optimized separately in two optimization problems and then concurrently in one single problem.

In Sections 5.1.1 and 5.1.2, we have already discussed how SO and MO problems can be formulated. In the following sections, we will focus on the characteristics of our specific problem. In particular, we will first discuss the problem dimension in Section 5.4.1, and its objectives and constraints in Section 5.4.2. Finally, in Section 5.4.3 we will discuss the optimization approach.

5.4.1. Problem Dimension

When optimizing a problem, its dimension is crucial for the optimization outcome. The problem dimension corresponds to the decision vector length: hence, more variables to be optimized will result in a longer decision vector and bigger search space, which typically complicates the optimization process.

In our case, we have different problem dimensions, depending on the type of problem. For SO, we have the following variables:

1. α_{E1}: first cone angle of the geocentric phase (see Section 4.3.2 for its definition). It holds $\alpha_{E1} \in [0°, 90°]$.

2. α_{E2}: second cone angle of the geocentric phase (see Section 4.3.2 for its definition). It holds $\alpha_{E2} \in [0°, 90°]$.

3. α_{E3}: third cone angle of the geocentric phase (see Section 4.3.2 for its definition). It holds $\alpha_{E3} \in [0°, 90°]$.

4. α_{S1}: spiral inwards phase cone angle (see Section 4.3.1 for its definition). It holds $\alpha_{S1} \in [-90°, 0°]$.

5. α_{S2}: circularization phase cone angle (see Section 4.3.1 for its definition). It holds $\alpha_{S2} \in [-90°, 90°]$.

6. α_{S3}: orbit cranking phase cone angle (see Section 4.3.1 for its definition). It holds $\alpha_{S3} \in [0°, 90°]$.

7. α_{S4}: spiral outwards phase cone angle (see Section 4.3.1 for its definition). It holds $\alpha_{S4} \in [0°, 90°]$.

8. T_{launch}: launch date (i.e., Julian date in which the sailcraft will be launched into the initial GTO orbit). It holds: $T_{launch} \in$ [2022-01-01, 2028-01-01], where the launch date is defined in yyyy/mm/dd.

9. ω_0: argument of perigee of the initial GTO orbit. It holds: $\omega_0 \in [0°, 360°]$.

10. R_1: distance at which the circularization phase will start. It holds: $R_1 \in [0.3 \text{ au}, 0.7 \text{ au}]$.

11. R_2: distance at which the cranking phase will start. It holds: $R_2 \in [0.26 \text{ au}, 0.3 \text{ au}]$.

For the MO problem, the same variables are also present, with the addition of the sail area, which will now vary in the following range: $A \in [5000\text{m}^2, 24000\text{m}^2]$. Hence, the problem dimension is 11 for SO and 12 for MO. Concerning the objectives, while in the SO mission the flight time is the only objective, for the MO case, there are four different objectives. These include the time of flight, the sail mass and two constraints treated as objectives (i.e., the final time constraint and the final orbit constraint, whose description is presented in Section 5.4.2). Moreover, for the SO case, another optimization strategy for analyzing whether this problem can be solved in a better way by splitting the problem into two different ones with a lower dimension was also investigated. Its general description is presented in Section 5.4.3 and the results of this different strategy compared with the original one are discussed in Chapter 8.

5.4.2. Objectives and Constraints

The aim of this study is to either minimize the time of flight (tof), for the SO case, or the time of flight and the sail mass (m), for the MO case. Both of these shall be minimized while satisfying the constraints. The time of flight is computed by simply counting the required time for the sail to reach the final circular orbit with 90° inclination and 0.4 au distance from the Sun. Whereas the mass is computed by multiplying the sail loading by its area.

While making its journey to the Sun, the sail has to fulfill several constraints. In particular, the following constraints are active:

1. The sailcraft shall not crash on any of the planets. For enforcing this, we have stopped and penalized (with a death penalty) any simulation where the sailcraft goes below a safe distance from each planet. These distances (in terms of altitude) were chosen to be equal to 100 km for the Earth, 1500 km for Venus and Mercury and 300 km for the Moon. Also, the sailcraft cannot go below 0.22 au from the Sun (otherwise the materials and instruments are not capable of working well anymore, due to the Sun's heat and magnetic field).

2. The sailcraft shall reach a final circular orbit at 0.4 au from the Sun, with an inclination of 90°.

3. The sailcraft shall reach the final orbit before the 1st of January 2033, for scientific reasons.

The first constraints were maintained as death penalties however, the other two were transformed to allow the algorithms to handle them in a more efficient way: the implemented procedure for transforming them is presented in Sections 5.4.2.1 and 5.4.2.2. Also, in the MO problems, the constraints 2 and 3 (the transformed versions) are turned into objectives to be minimized since our employed MO algorithms cannot handle constraints. Thus, in the MO problem, we will handle four different objectives.

5.4.2.1 Final Orbit Constraints

The final orbit constraints cannot be treated as death penalties, since a final trajectory with an almost circular orbit and parameters that are near to the ideal ones (e.g. 0.4 au of semi-major axis and 90° of inclination) might still be an appealing mission scenario. However, it has also to be considered that a variation of 1% in the inclination is different than a variation of 1% in the semi-major axis. For taking this into account, we have turned these constraints into one single ΔV constraints. This means that we take into account how much it would cost in terms of ΔV to correct the final orbit and reach the ideal one. Ideally, we would like to achieve an orbit with $\Delta V = 0$, meaning that we place the sailcraft into an orbit with zero eccentricity, 0.4 au of semi-major axis and 90° of inclination. However, whenever this is not done, we compute the cost in terms of ΔV for correcting this orbit, so that a correction of inclination will have a different effect than a correction in the eccentricity or in the semi-major axis. In this way, we ensure a more physical representation of the problem. In particular, we consider as if an impulse maneuver should be done for correcting these orbits since we know the mathematical equations for computing the ΔV in such cases. As indicated in Wakker (2015), we can compute this ΔV (in the optimal case in which the propellant consumption for doing the maneuver shall be minimized) as follows:

$$\Delta V_i = 2\sqrt{\frac{\mu_s}{a_f}}\sin(\Delta i/2) \tag{5.44}$$

$$\Delta V_{a,e} = \begin{cases} V_{c,p}(\sqrt{1+e_t} - \sqrt{1+e_f}) + V_c(1-\sqrt{1-e_t}) & \text{if } r > r_{a,f} \\ V_c(\sqrt{1+e_t} - 1) + V_{c,a}(\sqrt{1-e_t} - \sqrt{1-e_f}) & \text{if } r_{p,f} < r < r_{a,f} \\ V_c(\sqrt{1+e_t} - 1) + V_{a,f} - V_{c,a}\sqrt{1-e_t} & \text{if } r < r_{p,f} \end{cases} \tag{5.45}$$

where r is the radius of the ideal orbit (i.e., 0.4 au) and V_c is its circular velocity (computed as: $V_c = \sqrt{\mu_s/r}$), e_f and a_f are the final eccentricity and semi-major axis, reached by the sailcraft when the simulation is stopped, $r_{a,f}$ and $r_{p,f}$ are the apocenter and pericenter radii of the final orbit when the simulation is stopped (computed as: $r_{a,f} = a_f(1+e_f)$ and $r_{p,f} = a_f(1-e_f)$), $V_{c,p}$ and $V_{c,a}$ are the circular pericenter and apocenter velocities of the reached orbit when the simulation is stopped (computed as: $V_{cp} = \sqrt{\mu_s/r_{cp}}$ and $V_{ca} = \sqrt{\mu_s/r_{ca}}$), $V_{a,f}$ is the apogee velocity (computed as $V_{a,f} = \sqrt{\mu/a_f(1-e_f)/(1+e_f)}$), μ_s is the Sun's gravitational parameter (in m^3/s^2) and Δi is the required inclination change.

Also, e_t is the eccentricity of the transfer orbit that is used for going from the reached elliptical orbit to the final ideal one: this is computed differently depending on the cases. In the case that $r > r_{a,f}$, then the impulse is performed at the pericenter of the reached orbit and it thus holds $e_t = (r - r_{p,f})/(r + r_{p,f})$. In the case that $r_{p,f} < r < r_{a,f}$, then a two-impulses maneuver with one of the impulses applied in the apocenter of the final orbit is performed: hence $e_t = (r_{a,f} - r)/(r_{a,f} + r)$. In the case that $r < r_{p,f}$, then the impulse is performed at the apogee of the reached orbit: it thus holds $e_t = (r_{a,f} - r)/(r_{a,f} + r)$. For changing eccentricity and semi-major axis, we have decided to implement these impulsive strategies because, as explained in Wakker (2015), they represent the optimal ones (in terms of propellant consumption) to reach a circular orbit from an elliptical one, in an impulsive maneuver scenario.

Our final constraint will thus be:

$$g_1 = \Delta V = \Delta V_i + \Delta V_{a,e} \tag{5.46}$$

The implemented optimization algorithms will thus try to maintain $g_1 \leq 0$, whenever possible. Of course, a ΔV smaller than zero does not make sense, and the algorithms will thus effectively try to push the sailcraft towards the ideal orbit when trying to satisfy this constraint.

5.4.2.2 Final Time Constraint

This constraint is handled by transforming the ideal final time (i.e., T_{ideal}) and the simulation final time (i.e., T_{final}) in Julian seconds, and by computing their difference. In particular, the following is done:

$$g_2 = \begin{cases} T_{final} - T_{ideal} & \text{if } T_{final} > T_{ideal} \\ 0 & \text{otherwise} \end{cases} \tag{5.47}$$

where the ideal final time corresponds to the Julian date of 1st January 2033, expressed in Julian seconds.

Similarly to what happened for the final orbit constraint, the optimization algorithms will try to satisfy $g_2 \leq 0$, which will effectively mean to only maintain the orbits that fulfill the final time constraint, if possible.

5.4.3. Optimization Approach

As we have already mentioned above, all the variables are box-bounded (meaning that they vary within fixed lower and upper boundaries). The first population is generated randomly, using a uniformly distributed random generator: afterwards, this population is fed to the optimization algorithm that performs the evolution process and improves the individuals in the population. Every comparison between different algorithms is run three times and starting with the same initial populations (thus using the same seed for the random number generators) so that the results are based on the same initial population sets and are thus more representative of the differences between the algorithms. As we have to deal with both single and multi-objective problems, both SO and MO algorithms have been implemented. Also, the SO part was optimized using two different strategies:

1. Separate geocentric and heliocentric phases: in this case, the optimization process was split into two different problems. The first one is a single-objective unconstrained problem where the sail has to escape the Earth's gravitational pull in the fastest possible time (i.e., minimizing tof). In this case, the variables will only be 5: α_{E1}, α_{E2}, α_{E3}, T_{launch} and ω_0. Also, no constraints are present, besides the death penalties that are activated when the sailcraft reaches too low altitudes around the Earth and/or the Moon: in these cases, the simulation is stopped and the individual is discarded by assigning an extremely high tof to its orbit. Applying several SO algorithms for optimizing this, we will eventually obtain the best variables to optimize the geocentric orbit. We will then be ready to use the final condition of the best geocentric orbit for optimizing the heliocentric phase separately. In this second sub-problem, the objective is still the tof, but there are now the final orbit and final time constraints indicated in Equations 5.46 and 5.47 (besides the death penalties, in case the sailcraft reaches too low altitudes over the Earth or Venus or Mercury or the Sun). For this sub-problem, we will thus deal with six variables: α_{S1}, α_{S2}, α_{S3}, α_{S4}, R_1 and R_2. In this case, the optimization algorithms will not return a single best solution (unless both the constraints will be satisfied and the tof minimized), but the optimal solution will be a trade-off between the constraints violations (particularly, the ΔV one) and the tof. For this reason, as we will see in Chapter 7, this problem can be treated as an SO problem but studied and evaluated as a bi-objectives one, where feasible candidates in terms of final time constraint are compared in terms of tof and ΔV.

2. Concurrent geocentric and heliocentric phases: in this framework, the geocentric and heliocentric phases are treated as one single optimization problem with 11 variables and the same constraints as the separate case.

Concerning the MO case, besides adding the mass as objective, the two constraints (i.e., those expressed in Equations (5.46) and (5.47)) are also treated as such: thus constructing a four objectives unconstrained problem, which can be treated using NSGA-II, MOEA/D, NSPSO and MHACO. Of course, the death penalties related to the minimum allowable distances to the various celestial bodies are still active.

Chapter 6

Software

Both the trajectory simulation and optimization procedure mentioned in this thesis study will be implemented in the form of software models. Therefore, in Section 6.1, we will first describe the software architecture of the implemented models. Then, in Section 6.2, we will talk about the external software used for this research, as many of the optimization algorithms as well as environment models and numerical methods have been created using external software packages. Finally, in Section 6.3 the numerical methods used for the integrator and propagator, as well as their trade-offs, will be discussed.

6.1. Software Architecture

The previously discussed trajectory simulation and the optimization procedure, have to be performed using certain software. In the following section, we will discuss the software architecture for both these aspects.

6.1.1. Trajectory Simulation

As far as concerns the trajectory simulation, the software has to provide the user with the following aspects:

1. Environment.
2. External forces (i.e., perturbations and gravitational attraction of the main attractor).
3. Propagation and integration schemes.
4. Solar-sail vehicle model.

Whenever some of the aforesaid models were not already implemented in the selected software, they had to be implemented manually during the research.

We will assume that the environment does not encompass the vehicle, which is treated conceptually separately, as a vehicle model. An example of some relevant solar system body environment models that will be useful for our case is:

1. Ephemeris of the planets, the Moon and the Sun.
2. Gravity field of the celestial bodies.
3. Atmospheric model of the Earth.
4. Radiation model.

The software to be chosen shall be equipped with these environment models, or their implementation shall be possible.

Moreover, the solar-sail vehicle model shall include the following aspects:

1. Mass.
2. Area.
3. Attitude (i.e., cone and clock angles).
4. Sail characteristics (i.e., front and back non-Lambertian coefficients, front and back emissivity coefficients, reflectivity coefficient and specular reflection coefficient).

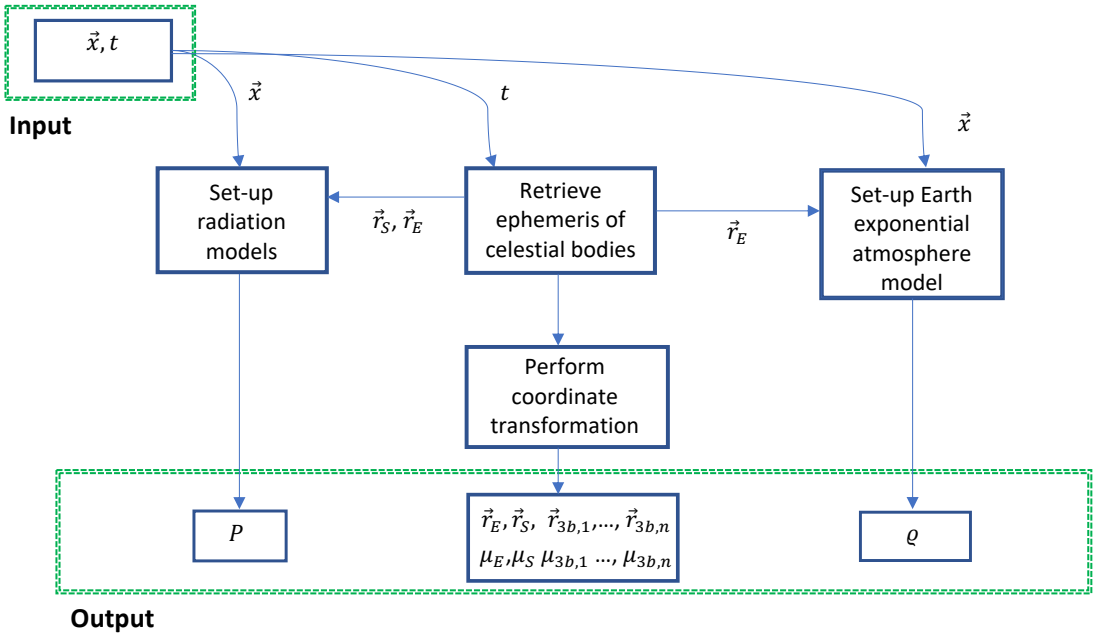


Figure 6.1: Environment architecture.

From a given attitude, the model shall be able to give, as output, the resulting non-ideal sail force. Since there is no loss of propellant, the mass of the vehicle will be kept constant throughout the mission, and thus no integration procedures are required to compute the mass.

To summarize, the software architecture related to the simulation model can thus be decomposed into the following sub-models:

1. Environment architecture.
2. External forces and vehicle architecture.
3. Guidance architecture.
4. Propagation architecture.

These architectures will be explained in the following sections using workflows. All the equations and variables specified in the following sections have correspondence with those used in this report unless otherwise stated.

6.1.1.1 Environment architecture

The environment architecture is shown in Figure 6.1. In the workflow, the two input values are the state vector of the sail (**x**) and the corresponding time (t). The indices *3b,n* refer to the third body gravitational perturbation of the n-body. These can be Mercury, the Moon, Venus, the Earth, and the Sun. On the other hand, the subscripts E and S refer to the Earth and the Sun, as these two bodies will act as central gravitational bodies in two different phases of the mission. Besides, the radiation model shall be set-up: in this way we can retrieve the radiation pressure value (P) of the Sun acting on the sailcraft. Finally, the Earth's exponential atmosphere model is set-up, and this helps us to determine the density (ρ) of the air at the altitude in which the sail is located w.r.t. the Earth.

6.1.1.2 External forces and vehicle architecture

The external forces and vehicle architecture can be seen in Figure 6.2. In particular, the total force due to the radiation pressure (i.e., $\mathbf{F}_{tot}$) is computed as shown in Equations (3.30) and

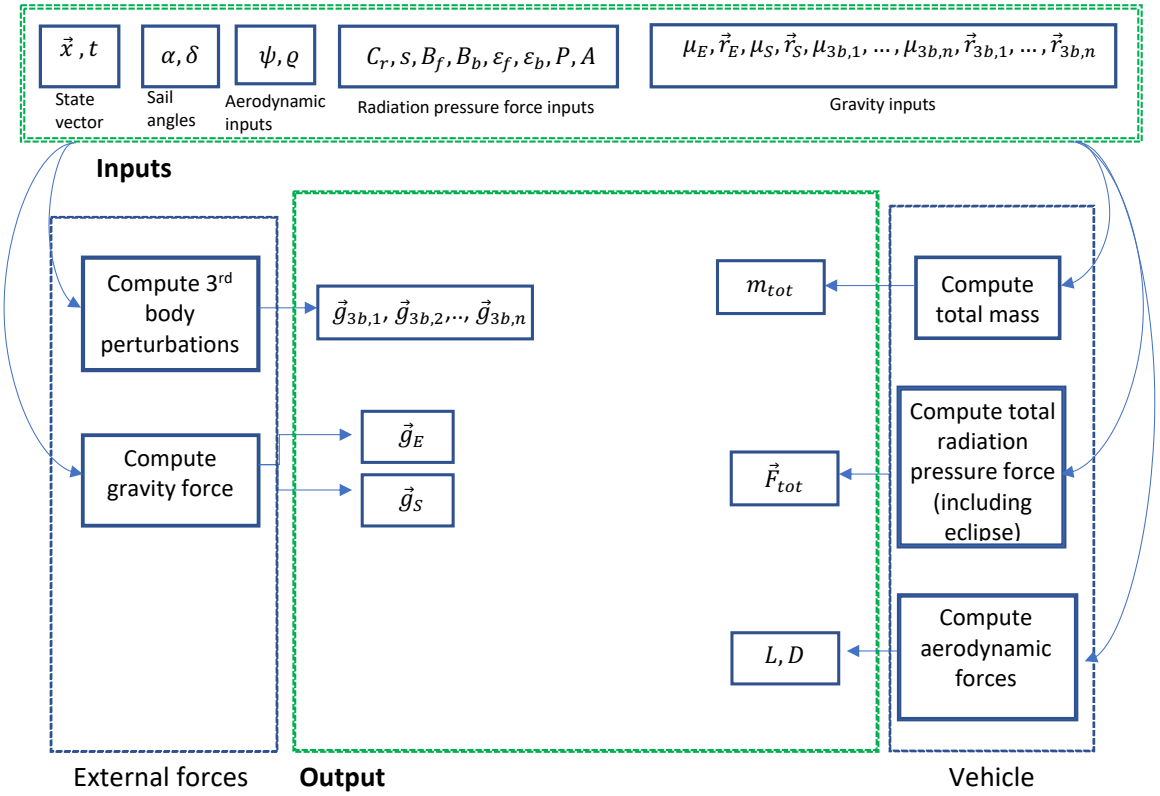


Figure 6.2: External forces and vehicle architecture.

(3.31). The only difference is that the eclipse is taken into account in the simulation model, which causes this force to go to zero when the sailcraft is shadowed w.r.t. the Sun.

In the workflow, we display several input blocks, with the following meanings:

1. Compute 3rd body perturbations: this block requires as input the state vector and the gravity inputs and it returns as output the various gravitational accelerations exerted by these bodies on the sailcraft.

2. Compute gravity force: this block requires as input the central bodies (i.e., Earth or Sun, depending on whether the sailcraft is in the geocentric or heliocentric phase) positions and gravitational parameters. Also, the state vector has to be fed as input to this block. This block then returns the gravity field of the central body.

3. Compute total mass: this block requires the sail area and sail loading as inputs and returns the total mass of the sail as output.

4. Compute total radiation pressure force (including eclipse): this block is useful for computing the total non ideal radiation pressure force exerted on the sail. For doing so, the radiation pressure force inputs are required, as well as the state vector, sail angles and position of the Sun.

5. Compute aerodynamic forces: this block requires as input the aerodynamic inputs, the state vector, and the sailcraft area and returns as output the lift and drag forces acting on the sail.

6.1.1.3 Guidance architecture

It is important to notice that the sail has to be properly steered to minimize the mission time and cost. Hence, in the simulation model, it is important to have a solar-sail guidance system

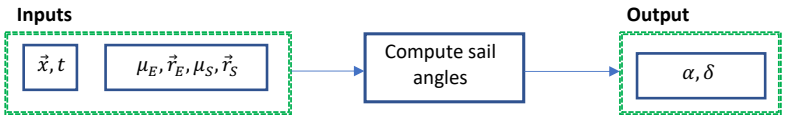


Figure 6.3: Guidance architecture.

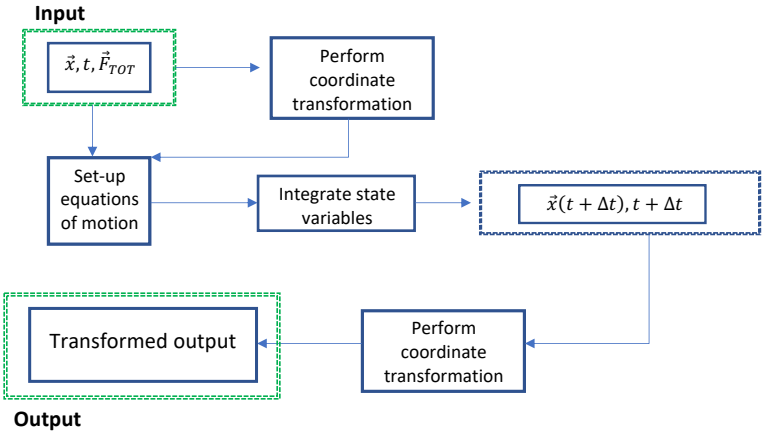


Figure 6.4: Propagation architecture.

that is capable of maneuvering the sailcraft depending on its position w.r.t. to the Earth and the Sun. The guidance architecture is shown in Figure 6.3.

6.1.1.4 Propagation architecture

Finally, inside the simulator, it is crucial to have a propagation module that is capable of taking the current state of the sailcraft and propagate it in time, depending on the forces acting on the body. The propagation architecture is shown in Figure 6.4. In this figure, we notice that the variable $\mathbf{F}_{TOT}$ is passed as input: this is the total force, as a summation of all the forces acting on the sailcraft (central body gravitational force, third-body perturbations, solar-sail force, aerodynamic forces, etc.). Also, for integrating the state vector and retrieve the state of the sailcraft, it is necessary to first set-up the equations of motion in the simulator. For doing this, a propagation model has to be established: this will be an outcome of this research study, as its choice influences both the CPU time and the quality of the solutions.

6.1.1.5 Simulation architecture

Having discussed all the aspects needed for constructing a simulation model, we can now integrate all the different models to create a simulation architecture. This is shown in Figure 6.5.

As can be seen in the figure, the simulation model presents an iterative nature, since it has to be continuously updated throughout the whole propagation time. Also, it can be seen that a block called termination conditions has been added as input: this includes all the conditions that, if violated, make the simulation end. For instance, if the sail approaches the Sun too closely or if the propagation time exceeds a certain maximum value, the simulation is interrupted. In this way, on the one hand, it is ensured that the simulation stops when the desired conditions are reached. On the other, it is also ensured that the simulation is stopped if certain constraints are violated, thus avoiding to waste CPU time.

6.1.2. Optimization Model

In Figure 6.6, the general architecture of the complete optimization model is shown. In this figure, we see that the simulation model has been enclosed inside the simulator block. Therefore, we take for granted that the optimization problem has already been set-up (either for SO

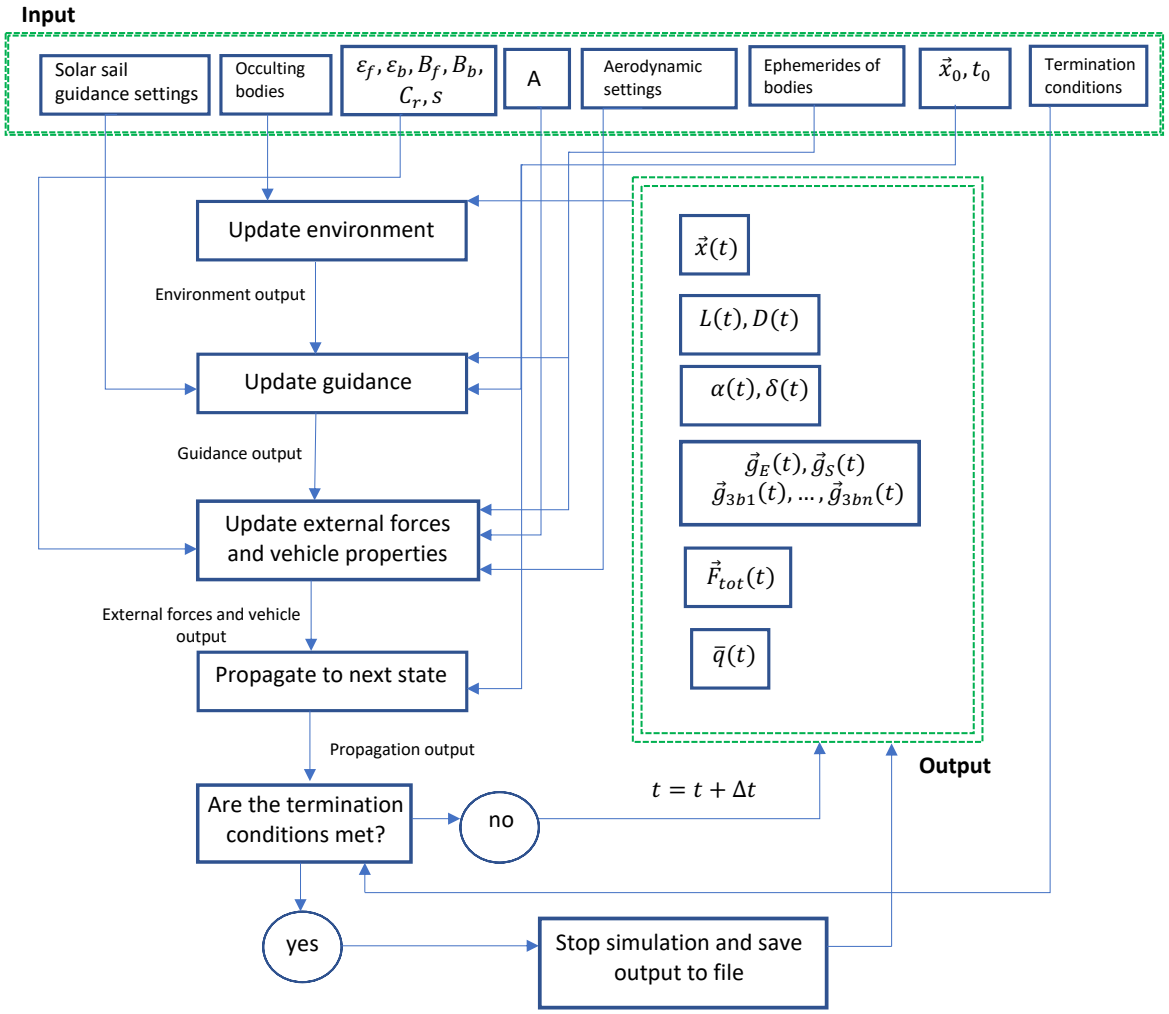


Figure 6.5: Simulation architecture.

or for MO) and that the simulator is ready for being employed in the optimization scheme by the user. As we can observe from the architecture, the algorithm input parameters, the number of individuals and the number of generations are necessary to construct the optimization method. Once this has been done, and once the initial population is randomly generated using a uniform random distribution and is evaluated in the simulator (i.e., a specific problem to be optimized), the objectives and constraints values can then be stored. Once this is done the optimization algorithm takes this population and its corresponding objectives and constraints, and evolves them to the next generation (in case that the stopping criteria are not met). This procedure is repeated until either the stopping criteria are met or the final generation number is reached. Once the evolution stops, the optimization algorithm will return the final population that has been evolved.

Of course, each optimization algorithm has its own working principle and a different evolution strategy. Nevertheless, as we will explain in Section 6.2, many of the optimizers used are implemented in external software and it is thus superfluous to discuss their specific structure here. However, the ant colony optimizer (for both SO and MO) represents a novelty of this research and requires a more detailed discussion. In Figure 6.7, the software architecture of the implemented ant colony optimizers (i.e., ACOmi and MHACO) is shown.

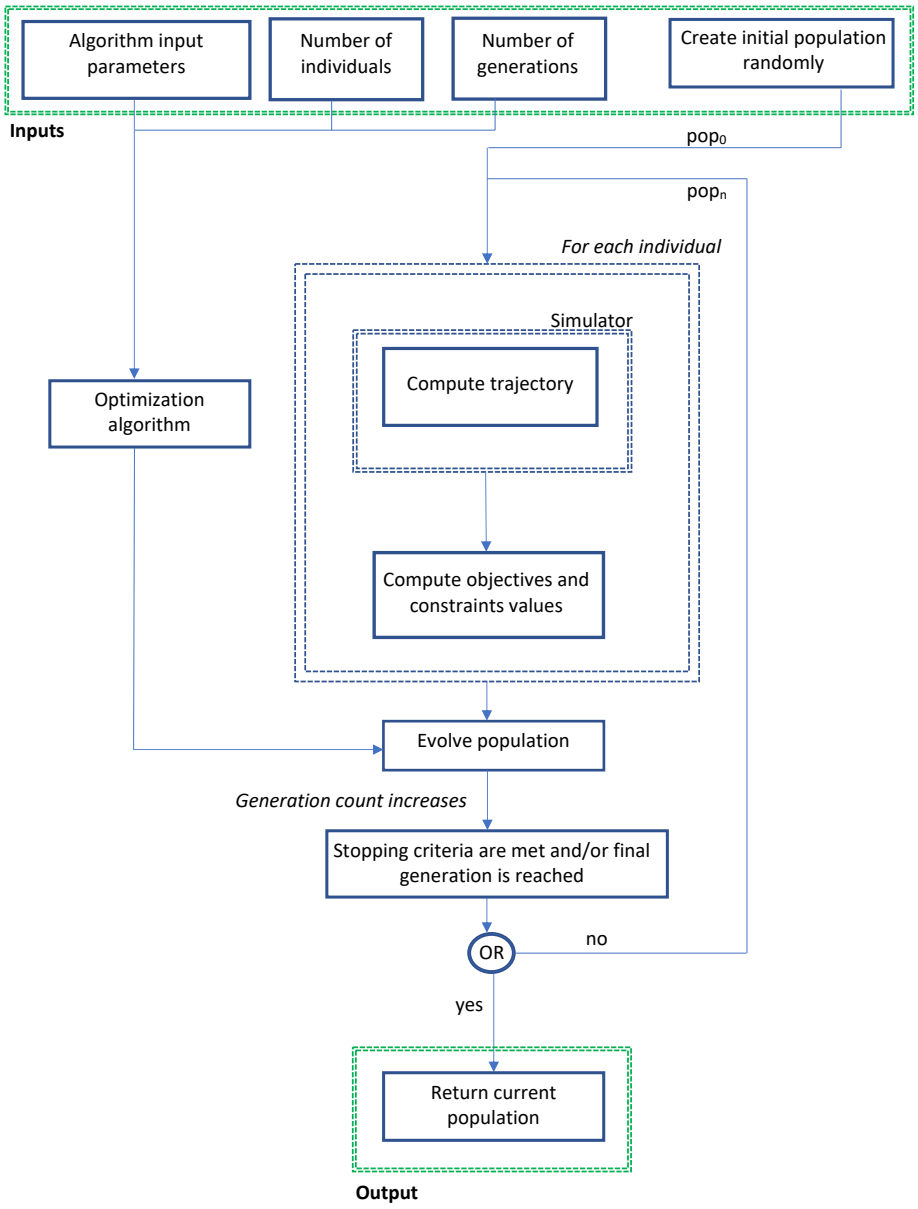


Figure 6.6: Architecture of the general optimization model.

6.2. External Software

Most of the software models are created using external software packages: whenever necessary (i.e., for the optimization algorithms and problems ACOmi, MHACO, NSPSO, and WFG, as well as for the non-ideal radiation model of the solar-sail) external software have been extended for including new models of our interest. In the following sections, we will discuss the external software used for both the simulation model, in Section 6.2.1, and the optimization procedure, in Section 6.2.2.

6.2.1. Simulation Model

In this section, we will discuss several different external software used for this research. In Section 6.2.1.1, we will first discuss a trajectory simulation software (i.e., Tudat) used for constructing and simulating all our solar-sail problem formulations. Then, in Section 6.2.1.2, we will introduce Pykep: a trajectory simulation software used during the verification and validation phase, as it makes several global trajectory optimization problems available for

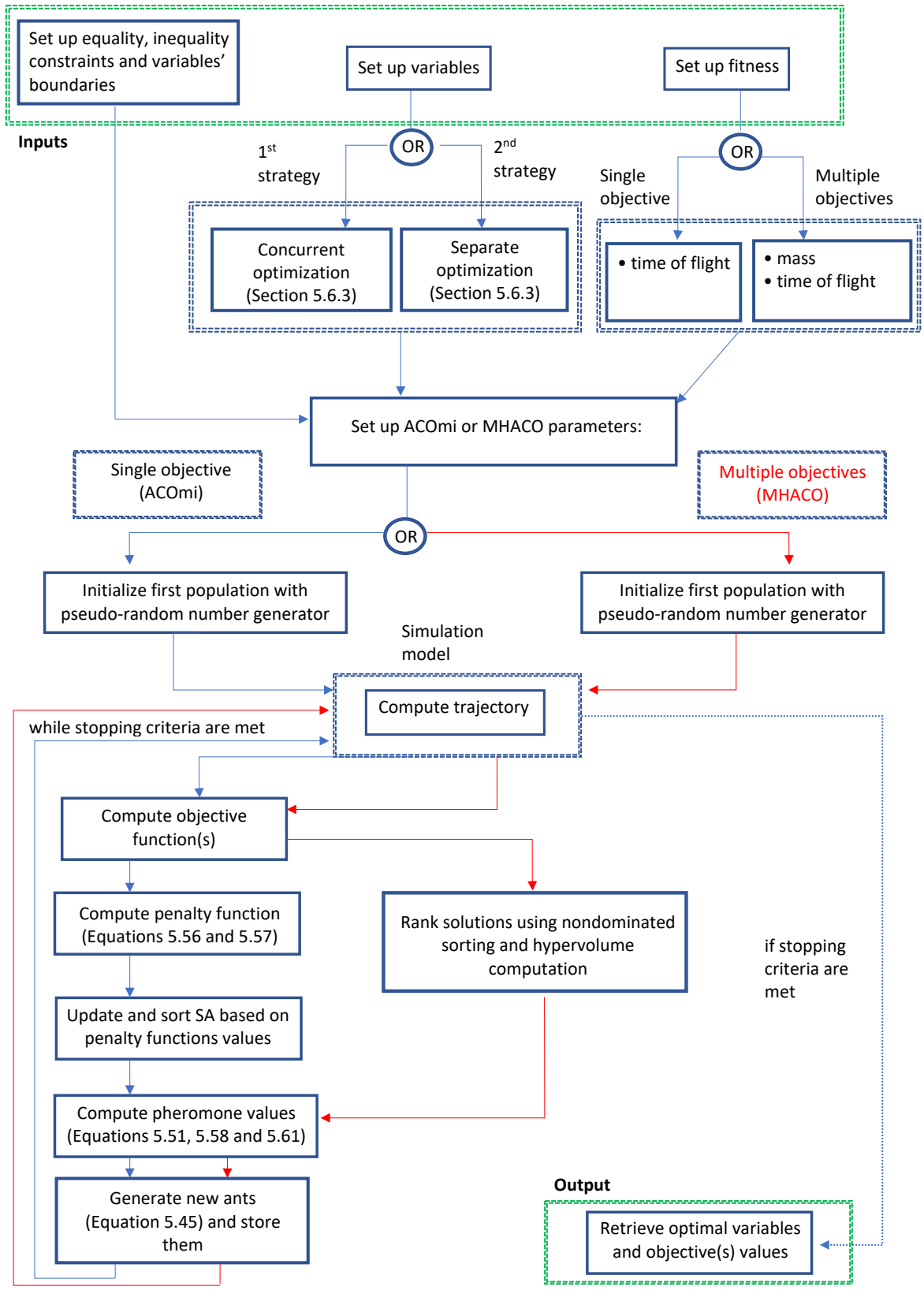


Figure 6.7: Architecture of the ant colony optimizers for both SO and MO.

use.

6.2.1.1 Tudat

For setting-up the simulation model of this research, the TU Delft Astrodynamics Toolbox (i.e., Tudat) was used. This is an open source software toolbox, which has been developed and maintained by TU Delft students and staff of the Aerospace Faculty. Everything is programmed in C++, and it is offered the possibility of a wide choice between environment models and numerical models. The software offers the following environment models for the trajectory simulation:

1. Ephemeris: defines the state of any celestial body as a function of time (Dynamical Barycentric Time seconds since J2000 is default). These ephemerides are tabulated and retrieved from SPICE [1].

2. Gravity models: defines the gravity field of the body, in terms of its gravitational potential and associated quantities.

3. Atmospheric model: defines the atmospheric properties (density, temperature, etc.) as a function of the relative position of the vehicle and time. This will be useful for modeling the atmosphere of the Earth during the geocentric phase (i.e., in particular during the drag phase).

4. Aerodynamic coefficient interface: defines the aerodynamic properties of the body, such as its aerodynamic coefficients as a function of some set of independent variables.

5. Radiation pressure interface: defines the radiation pressure properties of the body. At the moment of this research study, Tudat was only capable of simulating a cannon ball radiation pressure model for the Sun. For this reason, it was necessary to implement an extension for the environment, guidance and force settings, to compute the solar-sail non-ideal force. As explained in Appendix B, this contribution has been thoroughly validated and verified and also included in the open-source software for allowing future researchers and students to use it.

6.2.1.2 Pykep

Pykep is a scientific library developed by the Advanced Concepts Team (ACT) at the European Space Agency (ESA) (Izzo, 2019). Its purpose is to provide basic astrodynamics tools for aerospace researchers. The library is entirely written in C++ and also interfaced in Python. Although this software does not offer the variety of environment and numerical models of Tudat, which is the main reason why the former was preferred in this research, however, it offers a wide library of space problems inspired from real space missions (such as Cassini, Rosetta, Messenger, etc.). These problems are already implemented as optimization problems[2] and can directly be coupled with any optimizer. We have thus decided to make use of this software for testing the new algorithms developed in this research on space applications. The outcome of these benchmarks on several space problems are thoroughly discussed and presented in Appendix B.

6.2.2. Optimization Procedure: PaGMO

The optimization model employed in this research makes use of the global optimizers found in the PaGMO library (Biscani and Izzo, 2019). PaGMO (C++) or PyGMO (Python) is a scientific library for massively parallel optimization. It offers a wide range of global and local optimizers, as well as well-known test problems. We have made use of this library for both the global optimization techniques and various test problems. PaGMO and PyGMO are typically used for solving constrained, unconstrained, continuous and integer, single-objective and multiple objective optimization problems. Whenever new algorithms had to be implemented

[1] `https://naif.jpl.nasa.gov/naif/spiceconcept.html`, date of access: August 2019
[2] `https://esa.github.io/pykep/documentation/trajopt.html`, date of access: August 2019.

(e.g. ACOmi, MHACO, and NSPSO) or new test problems had to be tested (e.g. WFG), which were not available in PaGMO, it was decided to integrate these methods within the software, so that future researchers can also use them in their work. PaGMO is indeed developed in a full Free/Libre and Open-Source Software (FLOSS) philosophy. This means that the source code is openly shared and available and people are encouraged to contribute to the PaGMO project. Of course, there are several tests and verification procedures that have to be undergone before new optimization algorithms or test problems are implemented. This can be a downside, as it requires a lot of testing and debugging time, however, it has resulted to be extremely useful in our study, as it has allowed us to thoroughly verify and validate all the newly implemented methods. This software was conceived by the ACT and it has been extensively used for space applications (Izzo, 2010), (Izzo, 2007), (Vinkó and Izzo, 2008), (Biscani et al., 2010).

PaGMO provides a long list of optimizers[3] and test problems[4]. Each optimizer requires several input parameters:

1. A box-bounded optimization problem (which can typically be mixed-integer, constrained or unconstrained, single-objective or multi-objective, depending on the type of optimizers to be used).
2. The dimension of the continuous search space (i.e., continuous variables' dimension).
3. The dimension of the integer search space (i.e., integer variables' dimension).
4. The number of objective functions.
5. The number of equality constraints.
6. The number of inequality constraints.
7. The lower and upper bounds of each variable.

Besides, each optimizer requires a set of specific input parameters: their choice was performed in two manners. In some cases, these parameters were selected based on the suggestions of the authors of these algorithms (whose formulation and original papers are referred to in Appendix C). In some other cases, if this information was not available or if better sets of input parameters were found, these input parameters were changed during the verification and validation phase. In particular, in case an algorithm was clearly outperformed by the others, its input parameters were modified in an attempt to improve its performances. The test problems solved in the verification and validation phase and the performances of the algorithms on these tests can be further analyzed in Appendix B.

As a result of the aforesaid two strategies, the input parameters for all the SO and MO algorithms available in PaGMO were selected. We will now list their values, for each of the implemented methods. In particular, concerning artificial bee colony (ABC) only the maximum number of trials for abandoning a source has to be chosen. This was set to 20.

Conversely, when considering the simple genetic algorithm (SGA), the following parameters were chosen:

1. CR: crossover probability. This was set to 0.9.
2. m: mutation probability. This was set to 0.02.
3. p_m: mutation strategy. This was set to polynomial mutation.
4. p_s: selection strategy. This was set to be tournament selection.
5. p_c: crossover strategy. This was chosen to be exponential.
6. DI: distribution index. This was set to be 1.

[3] `https://esa.github.io/pagmo2/docs/algorithm_list.html`, date of access: August 2019.
[4] `https://esa.github.io/pagmo2/docs/problem_list.html`, date of access: August 2019.

7. ST: tournament size. This was set to be 2.

Regarding the particle swarm optimizer (PSO), the following parameters were selected:

1. ω: particles' inertia weight. This was set to 0.7298.
2. η_1: magnitude of the force applied to the particle's velocity, in the direction of its previous best position. This was set to 2.05.
3. η_2: magnitude of the force applied to the particle's velocity in the direction of the best position in its neighborhood. This was set to 2.05.
4. V_{MAX}: maximum allowed particle velocity, as a fraction of the box-bounds. This was set to 0.5.
5. N: neighborhood parameter, which handles the width of the neighborhood, for each particle. This was set to 4.

Moreover, for the differential evolution optimizer (DE), the following parameters were chosen:

1. F: weight coefficient. This was set to 0.8.
2. CR: crossover probability. This was set to 0.9.

While for the self-adaptive variants of DE (i.e., SADE and DE1220), the parameters are self-tuned and do not need to be set by users.

Concerning the MO optimizers, PaGMO makes available two different algorithms: MOEA/D and NSGA-II. In the first case, the following input parameters were chosen:

1. WG: weight generation. Method used to generate the weights. In our case, the weights are generated using the "grid" method.
2. D: decomposition method. This was set to be Chebyshev.
3. N: size of weight's neighborhood. This was set to be 20.
4. CR: crossover parameter. This was set to be 1.
5. F: parameter for the differential evolution operator. This was set to 0.5.
6. η_m: distribution index used for polynomial mutation. This was set to 20.
7. P: probability that the neighborhood is considered at each generation, rather than the entire population. This was set to 0.9.
8. L: maximum number of copies reinserted in the population. This was set to 2.

Whereas concerning the NSGA-II algorithm, the following input parameters were selected:

1. CR: crossover probability. This was set to 0.95.
2. η_c: distribution index for crossover. This was set to 10.
3. m: mutation probability. This was set to 0.01.
4. η_m distribution index for mutation. This was set to 50.

The output of all these optimizers consist of the population of individuals throughout the generations, together with their objective functions and constraint violations values.

This research has extensively contributed to PaGMO's software, by implementing a new single objective ant colony optimizer[5] (i.e., ACOmi), a multi-objective test-suite[6] (i.e., WFG), a completely new MO ant colony optimizer [7] (i.e., MHACO), as well as a MO particle swarm optimizer[8] (i.e., NSPSO).

[5] https://esa.github.io/pagmo2/docs/cpp/algorithms/gaco.html#_CPPv4N5pagmo4gacoE, date of access: August 2019.

[6] file:///Users/giacomoacciarini/Develop/pagmo2/doc/sphinx/_build/html/docs/cpp/problems/wfg.html#_CPPv4N5pagmo3wfgE, date of access: August 2019.

[7] https://github.com/esa/pagmo2/pull/326, date of access: August 2019.

[8] https://esa.github.io/pagmo2/docs/cpp/algorithms/nspso.html, date of access: September 2019.

6.3. Numerical Methods

In this section, we will discuss a trade-off for the numerical methods used in this research. In particular, we will first benchmark the propagator to be used, in Section 6.3.1. Afterwards, in Section 6.3.2, we will also discuss the reason why a Runge-Kutta Fehlberg technique has been chosen as the integrator scheme.

6.3.1. Propagator Selection

Numerical integration has started to play a pivotal role in solving the orbital mechanics problem, in favor of analytics and semi-analytics techniques, which were used in the past. However, through the transformation of the state variables in the equations of motion, it is possible that the overall dynamical stability of the integration is improved and better results are obtained. In particular, by expressing the differential equations as $\ddot{\mathbf{r}} = \sum \mathbf{F}/m$ we usually assume that Cartesian state variables (i.e., Cowell propagator) will be used for propagating the orbit. However, these are just one out of many possible formulations: there are many equivalent representations to express the dynamics. Although these other representations are equivalent, some of them may bring advantages with respect to the others (e.g. in terms of accuracy of the solution, or efficiency of the computation, or stability: i.e., singularities are avoided). Hence, it is worthy to discuss different propagation schemes and their advantages and disadvantages. Usually, Cartesian coordinates are used for their simplicity and numerical properties, but also other formulations exist, which can become more attractive for certain kinds of problems. Some of the most popular propagation schemes for space applications are:

1. Cowell.
2. Encke.
3. Kepler elements.
4. Modified equinoctial elements.
5. Unified state model.

All these propagators make use of the sets of state variables discussed in Section 3.2.

Every propagation scheme can be developed starting from the general formulation of the equations of motion as:

$$\frac{d\mathbf{x}}{ds} = \mathbf{f}(\mathbf{x}, s; \mathbf{p}, \mathbf{u}) \tag{6.1}$$

where $\mathbf{x}$ is the state variables vector; s is the independent variable (which is usually the time t); $\mathbf{p}$ are the environmental parameters (e.g. the gravitational parameter); and $\mathbf{u}$ are the control parameters. The choice of the propagation scheme influences the state variables $\mathbf{x}$ and the independent variable s.

The first propagation scheme, and the most straightforward one is the Cowell propagator, which makes use of the Cartesian elements as state variables, and the time as the independent variable (i.e., $\mathbf{x} = (\mathbf{r}^T, \dot{\mathbf{r}}^T)^T$ and $s = t$). For very long and perturbed missions its large values for the state derivative and its variations make difficult to adapt the time step and make the method subject to large numerical error. Nevertheless, due to its robustness (i.e., it is singularity free) and simplicity, it will be used as a benchmark for the propagator selection.

The Encke propagator selects as independent variable the time and as dependent variable the variation from a perturbed and an osculating orbit (i.e., $\mathbf{x} = (\Delta\mathbf{r}^T, \Delta\dot{\mathbf{r}}^T)^T$ and $s = t$). The Encke propagator brings some advantages with respect to Cowell: the state derivative and their variations are now small. However, the method loses its advantages over time when $\Delta\mathbf{r}$ becomes of similar magnitude to $\mathbf{r}$. Also, the mathematical representation of the differential equations is more complex. Hence, we understand that for very long and perturbed simulations (such as our solar sailing mission), this method presents drawbacks, which can seriously compromise the results: we thus exclude this method from our propagators' choice.

Another possibility is to use the Keplerian elements: in this case, the true anomaly is the only element that varies fast during the propagation, whereas the other elements are usually subject to slower variations. However, although this representation mostly solves the issues of the aforesaid propagators, it is not singularity free. Indeed, there are singularities not only for $e = 0$ and $i = 0$, but also for hyperbolic orbits at large distances. For this reason, also this propagator is discarded for our case.

Starting from this set of Keplerian elements the orbital elements have been reformulated to prevent singularities. This has led to another propagator scheme: modified equinoctial elements (MEE). This formulation, together with the unified state model (USM) seems to be quite attractive for the optimization of solar sailing missions. Both of these techniques use time as an independent variable. Also, three different types of USM exist. In Section 3.2, the state variables corresponding to each of these propagators are introduced and discussed. Furthermore, in Appendix A the four possible representations of the equations of motion using the three types of USM and the MEE propagators are mathematically discussed.

In this section, we limit ourselves in doing a trade-off for testing the three USM propagators and the MEE propagator against Cowell. In Mooij (2012), USM, MEE, and Cowell were compared for a solar sailing polar mission. The objective of the paper was to assess the advantages and disadvantages of these methods, in terms of accuracy and computation time.

In particular, these three methods were compared for the same solar sailing polar mission described by Candy (2002), and they were assessed for a model in which an evolutionary algorithm is used as optimizer, and both SO and MO optimizations are tested. This long duration low-thrust mission, which also makes use of an optimizer, is ideal for testing these propagators in terms of CPU load, number of function evaluations and accuracy of the results. Minimizing the CPU load per each run can bring significant enhancements: we would gain a lot of time to be used for tuning the optimizer, for instance. This low-thrust mission has indeed a duration around 10 years, and thus the CPU time is in general large (i.e., around twenty seconds per function evaluation). The MEE has turned out to have a superior performance w.r.t. Cowell in terms of stability. Indeed, the integration could continue from four up to ten times longer than the Cartesian model before the solution starts to deviate. The USM model has demonstrated to be exceptionally better than the Cartesian elements in terms of CPU time (usually one order of magnitude lower): this makes the USM very appealing for these kinds of optimization problems.

The results found in Mooij (2012) were found by using the same optimization method, and by testing four types of integration methods of the Runge-Kutta Fehlberg family

It is clear that these results heavily depend on the environment, integration, guidance and optimization models used. Therefore, although the missions are similar, slight variations in these models can cause the results to deviate. For these reasons, we have decided to perform this trade-off again, by applying it to the solar-sail mission with the geocentric and heliocentric phases optimized concurrently. Furthermore, it was decided to compare the propagators for a single-objective mission using the Delta-V Strategy (already introduced and discussed in Chapter 5). The ACOmi optimizer was used for optimizing each of these missions with different propagators, whereas the chosen integrator was Runge-Kutta Fehlberg 56 for all the runs, with an integration tolerance of 10^{-9} for the geocentric phase and 10^{-11} for the heliocentric phase.

The different propagators are confronted in an optimization framework where 100 individuals are evolved over 40 generations. The quality of the solutions in terms of constraints violations and fitness minimization when the different propagators are used will be analyzed. Also, a study of the different time required for running the optimization with a different propagator will be carried out, as well as an analysis of the number of final best solutions provided by each propagator.

In Figure 6.8, the Pareto fronts are shown for the best individuals over three runs. As can be seen, the results are plotted in a 2-dimensional graph in which the x-axis represents the ΔV (i.e., the constraint violation value in the final orbit), whereas the y-axis the time of flight (in Julian Days). Also, in the figure, we referred to the USM with quaternions as USM7, to the USM with exponential mapping as USM6e and to the USM with Rodrigues parameters

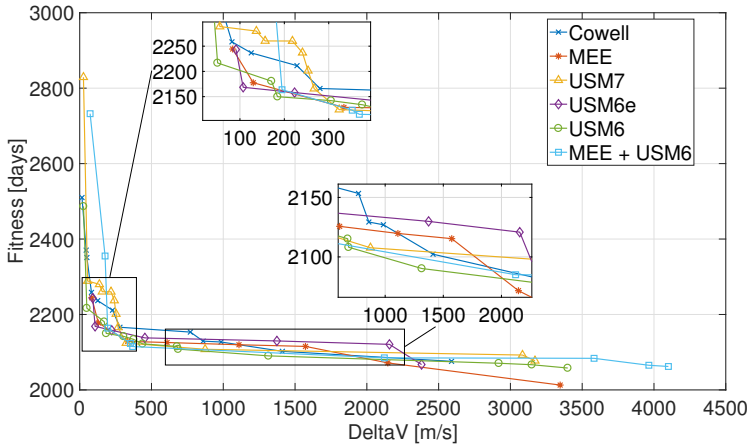


Figure 6.8: Pareto fronts of the best of all the three runs.

Table 6.1: Propagator trade-off: CPU time (i.e., Time), average number of best individuals (i.e., N) and their feasibility ratio (i.e., Feasibility).

	Cowell	**MEE**	**USM7**	**USM6e**	**USM6**	**hybrid**	*Units*
Time	26	18	27	31	29	20	[h]
N	13	8	8	9	11	9	[-]
Feasibility	85.9	100	89	100	97	100	[%]

as USM6. We can use this figure to compare and discuss the performances of the different propagators. However, this information has to always be coupled with the time required for each propagator to perform the optimization. This information is displayed in Table 6.1. There, the mission with two different propagators (i.e., MEE for the geocentric phase and USM Rodrigues for the heliocentric phase) is referred to as "hybrid". Moreover, the time represents the average wall clock time over three runs. Together with the time information, in the table we can also observe the average number of individuals that belong to the best Pareto fronts and their average feasibility ratio (i.e., ratio between number of feasible individuals and the total number of individuals, where the feasibility is established depending on whether the solution violates the final time constraint or not). Moreover, it has to be considered that only the individuals that managed to complete the orbit were considered for the statistics: this means that all the individuals that could not escape the Earth or crashed in any celestial body were excluded, as they also represent infeasible individuals. In particular, it was found that 25% of the overall individuals managed to complete the mission for MEE, USM7, USM6e and the hybrid one, and around 30% for Cowell and USM6.

As can be seen, the best propagator in terms of time is the MEE, with nearly half of the time w.r.t. the worst (i.e., USM6e). The only other propagator that is competitive with this time is the hybrid one, with only two hours more required, which corresponds to an approximate increase of 10%. However, by looking at the Pareto fronts, the situation is slightly different, and it results in a bit more difficult to identify a clear winner. When the best results over three runs are compared, we notice that the hybrid propagator explores the domain in the most spread out way, although it does not reach the minimum values of ΔV as USM6 and USM7. The USM6e seems to be the worst also in this case: its front, indeed, not only results to be the worst in terms of spread but also the worst in terms of Pareto points (i.e., all its points result to be dominated by other propagators except for one). By inspection, the USM6 propagator seems to be the best in terms of Pareto front (i.e., almost all its points seem to be Pareto dominant w.r.t. the other fronts), with the only exception of high ΔV, where its performances seem to be outweighed by MEE.

If we compare the number of Pareto individuals and their feasibility ratios (shown in Table 6.1), we observe that the Cowell propagator and the USM6 propagator have the highest average number of Pareto individuals per run. However, the average feasibility value of the

Cowell propagator is around 85%, which means that only 8.5 out of 10 of those Pareto points can actually be seen as such since the others violate the final time constraint and are thus not feasible solutions.

In conclusion, inspecting the Pareto fronts in Figure 6.8, and by looking at the numbers in Table 6.1, we can conclude that the best propagators for such a mission (considering not only the best individuals in the front but also the time required for running the optimization with different propagators) are USM6 and MEE. Another interesting candidate could also be the hybrid propagator, however, its Pareto fronts' shapes, as well as its number of Pareto individuals, do not seem to add strong advantages to the MEE propagator, which would justify the increased wall-clock time. Moreover, comparing USM6 and MEE, we can observe that their overall best Pareto fronts are very similar and a clear winner cannot be identified. The USM6 is able to find more Pareto fronts individuals, although the MEE has a smaller wall-clock run time (18 hours against 29).

All things considered, we have decided to use the MEE as a propagator for all the runs of the mission. Indeed, the performances of the other propagators do not seem to bring significant advantages that would justify longer simulation times. Therefore, this propagator will be used for all our optimization runs discussed in Chapter 7, unless otherwise stated.

6.3.2. Integrator Selection

Once the equations of motion are set-up, we need to solve them. To do this, we need numerical techniques to integrate the state derivative: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$. Since we will perform this operation numerically, assuming that $\mathbf{y}$ is the analytical solution and $\bar{\mathbf{y}}$ is the numerical solution, we will always have a residual error $\boldsymbol{\varepsilon} = \bar{\mathbf{y}} - \mathbf{y}$. Our objective is to select numerical integrators able to reduce as much as possible this error, while still maintaining the computational effort limited. The following integration methods represent the most popular choices for space applications:

1. Runge-Kutta (RK) variable step-size integrator.
2. Adams-Bashforth (AB) variable step-size/order integrator.
3. Bulirsch-Stoer (BS) variable step-size integrator.

First of all, the Adams-Bashforth integrator was excluded. Indeed, it has the natural tendency to lower the step-size during the integration, and this prevents it to integrate accurately high eccentric orbits (usually, this method is used for very low or moderate eccentric orbits) (Montenbruck and Gill, 2012). Since in our case we are dealing with orbits that can range from $e = 0$ to $e > 1$ we have to discard this method.

We are thus left with two different integration schemes: either a Runge-Kutta variable step-size integrator or a Bulirsch-Stoer variable step-size integrator. Research by Mooij (2012), has discussed a thorough trade-off between four different RK variable step-sizes integrators (i.e., RK Fehlberg 4(5), RK Fehlberg 5(6), RK Fehlberg 6(7) and RK Fehlberg 7(8)): in such a benchmark study, different relative and absolute integration tolerances have been tested and studied for a solar sailing polar mission. Due to the similarities of such a mission to our problem, we have thus decided to take this study as a baseline for the selection of our integration scheme. Besides, the BS method does not result to be ideal for problems that are either stiff or non-smooth and for differential equations that have singular points inside the interval of integration. Although almost all our propagator schemes do not have robustness problems (i.e., no singular points), the smoothness of the problem is not ensured and we thus prefer an RK Fehlberg technique, which is more stable for such problems (Kiusalaas, 2013).

All things considered, we have thus decided to select the RKF 5(6) integration scheme: this method, according to Mooij (2012), has demonstrated to have favorable properties for this particular type of mission. In particular, for balancing between function evaluations and accuracy, the RKF 5(6) method was chosen with a relative integration tolerance of 10^{-9} for the geocentric phase and 10^{-11} for the heliocentric phase, while the maximum step-size was left free to vary between 10^{-5} and 10^{4} s, as this was pointed out to be more advantageous. This

integration scheme and its setting are available for the use in Tudat and have been employed for this whole research study. In the following section, the description of the Runge-Kutta Fehlberg scheme will be presented.

Runge-Kutta Fehlberg Methods

Among the Runge-Kutta methods, there are a number of methods called adaptive. These produce an estimate of the local truncation error for a single step, for changing the step-size as desired. Hence, the working principle of these methods is to change the step-size based on an error evaluation at each step. The integration procedure of these methods can be represented as follows:

$$\mathbf{x}_{n+1}^{*} = \mathbf{x}_n + h \sum_{i=1}^{s} b_i^{*} \mathbf{k}_i \tag{6.2}$$

where the time is discrete and each time step can be related to the previous one as: $t_n = t_{n-1} + h$ and where $\mathbf{x}_{n+1}$ and $\mathbf{x}_n$ are the state vectors at the times t_{n+1} and t_n, respectively. For integrating the equations, it is enough to have the initial conditions $\mathbf{x}_0$, since the $\mathbf{k}_i$ factors can be written as:

$$\begin{aligned} &\mathbf{k}_1 = \mathbf{f}(t_n, \mathbf{x}_n) \\ &\mathbf{k}_2 = \mathbf{f}(t_n + c_2 h,\ \mathbf{x}_n + h(a_{21}\mathbf{k}_1)) \\ &\dots \\ &\mathbf{k}_s = \mathbf{f}(t_n + c_s h,\ \mathbf{x}_n + h(a_{s1}\mathbf{k}_1 + a_{s2}\mathbf{k}_2 + \dots + a_{s,s-1}\mathbf{k}_{s-1})) \end{aligned} \tag{6.3}$$

Moreover, the error can be computed as:

$$\mathbf{e}_{n+1} = \mathbf{x}_{n+1} - \mathbf{x}_{n+1}^{*} = h \sum_{i=1}^{s} (b_i - b_i^{*}) \mathbf{k}_i \tag{6.4}$$

where the c_i, $a_{i,j}$, b_i and b_i^{*} coefficients have known values that can be found in the original article where these methods were introduced (Fehlberg, 1964).

6.4. Verification and Validation

In this thesis study, the verification and validation (V&V) of the implemented software was done at two levels. First of all, each unit was verified separately. Furthermore, also the whole integrated system was verified. This was done for both the simulation model and the optimization procedure. All the methods and results of the verification and validation phases are discussed and presented in Appendix B. Concerning the simulation model, the purpose of the V&V phase was to make sure that all the implemented models were working as expected: for doing this, we have first checked that each unit of the simulation model was producing the expected results (for instance, we checked that the generated solar radiation pressure force was correct and that the software was free of bugs). Moreover, we have also validated the integrated system by reproducing the entire mission scenario of a previously studied solar sailing polar mission to the Sun.

Furthermore, concerning the optimization procedure, we first verified that the implemented methods were free of bugs and that the algorithms were fully covered by the tests. Afterwards, we also made sure that the performances of the implemented algorithms were not clearly outperformed by other popular and already verified optimization algorithms (e.g. differential evolution, genetic algorithms, etc.). By doing so, we have also tweaked and tuned both the standard algorithms and the implemented ones. This has allowed us to establish a set of competitive input parameters for all the implemented algorithms used in this research, over all the studied test problems. In this way, when optimizing the mission of our interest, we were able to use the results of the V&V phase in the choice of the input parameters.

Chapter 7

Results

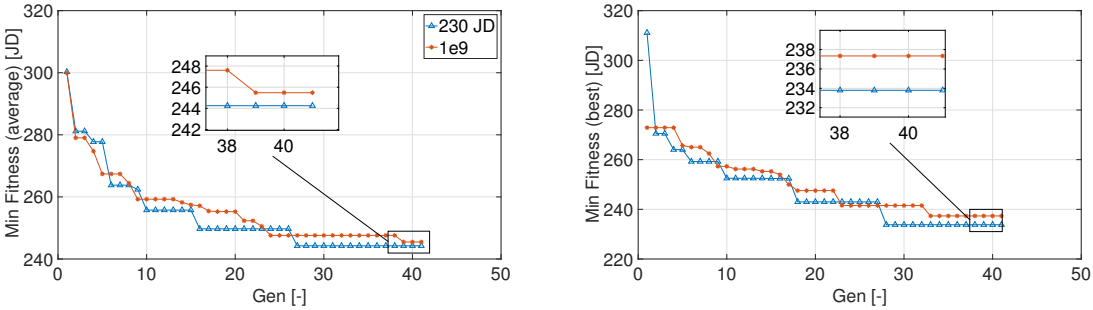
In this chapter, we will present and comment on the results of this thesis study. In particular, we will first discuss the algorithms tuning strategy, in Section 7.1. We will then apply the knowledge derived from this to perform optimizations of both the single-objective and multi-objective formulations of the solar sailing polar mission, introduced in Chapter 5. This is done in Sections 7.2 and 7.3, respectively. In particular, we will analyze and benchmark different orbits coming from different problems and solved various optimization algorithms. This will be done not only for both single and multi-objective types of mission but also for the geocentric and heliocentric phases both separately and together. Furthermore, a Monte Carlo local refinement will be performed and discussed for each optimized problem to understand whether nearby solutions can further improve the best results. Then, in Section 7.4, the random seed influence on the optimization results will be addressed. Finally, the optimal trajectory will be discussed in Section 7.5, by not only showing the trajectory to be flown but also displaying the evolution of some key variables during the sail journey to the Sun.

7.1. Algorithm Tuning

In this research study, three new algorithms were introduced: ACOmi, MHACO, and NSPSO. While NSPSO has been almost entirely derived from literature. However, the other algorithms have been developed in this thesis study and the choice for the values of their input parameters has to be discussed. Hence, while NSPSO requires a little tuning (only on those parameters that have not been subjected to previous studies), the other two algorithms require a more thorough investigation. In the following sections, we will deal with the tweaking and tuning of all these three methods. The problem in which these algorithms will be tested is the solar-sail problem of our interest. In details, we will first discuss the ACOmi tuning in Section 7.1.1, for then tackling the MHACO and NSPSO tuning in Sections 7.1.2 and 7.1.3, respectively. This study will be particularly useful for our research, as it will allow us to establish a set of input parameters to be used for optimizing the solar sailing polar mission and benchmark its optimization results against other well-known algorithms. A thorough description of both the single and multi-objective solar-sail problems used in the thesis study can be found in Section 5.4.

7.1.1. ACOmi Tuning

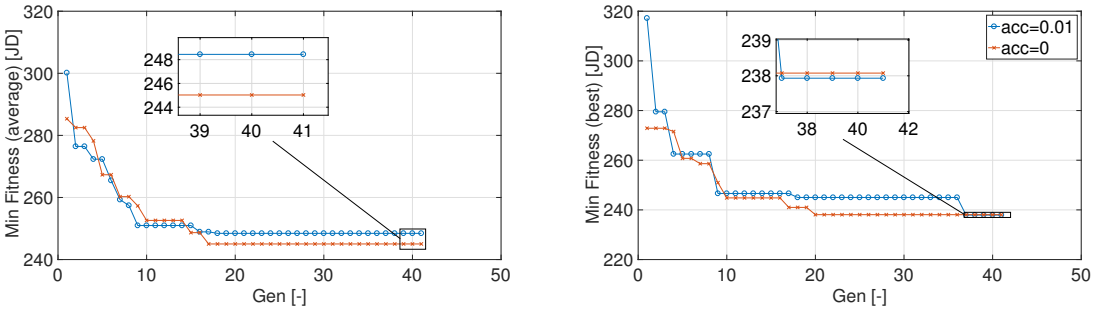
As we have already pointed out in Chapter 5, the ACOmi algorithm has several input parameters that can be chosen. Some parameters pertain to the convergence speed (i.e., q, $NGenMark$, $threshold$): their values, in principle, should be tweaked and tuned, however, since these parameters have been introduced in this research and have been thoroughly studied in several test problems discussed in Appendix B, we have come to the conclusion that their values can be kept fixed at $NGenMark = 7$, $threshold = 25$ and $q = 1.0$. Also, the $focus$ parameter has been maintained fixed at 0. We are thus left with only three parameters to tweak: ker, $oracle$ and acc. Two of these parameters (i.e., ker and $oracle$) were introduced in the original formulation of the extended ant colony optimizer (Schlüter, 2012). As reported there, the $oracle$ parameter should preferably be chosen as near as possible to the global optimum. Whenever this global optimum is not known, it should be chosen to be equal to 10^9. In our case, we do not precisely know the location of the global optimum. However, we know from previous studies that a total geocentric time of flight of 230 Julian days should be near the global optimum (Candy, 2002), (Garot, 2006), (Spaans, 2009). We have thus decided to perform a benchmark study where an optimization with an oracle parameter of 230 JD



(a) Average fitness for 100 individuals and 40 generations.

(b) Best fitness for 100 individuals and 40 generations.

Figure 7.1: Average and best fitness values for two different oracle parameters.



(a) Average fitness for 100 individuals and 40 generations.

(b) Best fitness for 100 individuals and 40 generations.

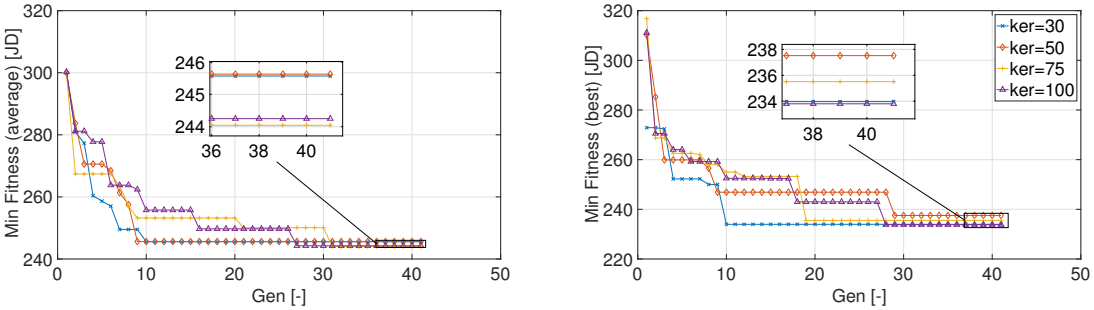
Figure 7.2: Average and best fitness values for two different accuracy parameters.

was compared with another one with an oracle parameter of 10^9 JD. In Figure 7.1, we show the best and average fitness values as a function of the generation number. Each optimizer was run with a fixed set of input parameters: $ker = 100$, $acc = 0.0$, $q = 1$, $threshold = 25$, $NGenMark = 7$, $focus = 0$. Also, each run was performed three times with three controlled seeds, so that the randomness could be removed and the optimizers could start from the same initial population. The results in the figure are representative of the best and average over three runs.

As we can see, the advantages of choosing a smaller oracle parameter (i.e., closer to the global optimum) are evident: runs with an oracle parameter of 230 JD not only reach the best and average minimum fitness overall, but they also are almost always below the curve of the high oracle parameter, meaning that they even reach the best values with fewer function evaluations. In our thesis study, we have thus decided to keep the oracle parameter to 230 JD.

Similarly to what has been done for the oracle parameter, we have decided to implement a similar tuning strategy for the accuracy parameter. This parameter, as explained in Section 5.3.2, regulates the distance between penalty function values stored in the archive. We are thus interested to see whether the removal of such distance (i.e., setting $acc = 0$) could be helpful to our case, or if a certain distance shall be maintained. We have thus checked the average and best fitness progression of two optimization algorithms: one with $acc = 0$ and another one with $acc = 0.01$. The other input parameters were selected to be: $ker = 100$, $oracle = 10^9$, $q = 1$, $threshold = 25$, $NGenMark = 7$, $focus = 0$. Each algorithm was run three times with three controlled seeds. The best and average results are shown in Figure 7.2

Although the best overall value is found for $acc = 0.01$ (i.e., 237.93 JD against 238.07 JD), the average runs with $acc = 0$ show a better convergence behavior both in terms of convergence speed and average minimum value. Also, their absolute minimum value is not



(a) Average fitness for 100 individuals and 40 generations.

(b) Best fitness for 100 individuals and 40 generations.

Figure 7.3: Average and best fitness values for four different solution archive sizes.

very far and is within a 0.5 JD distance. For all these reasons, we have decided to select an accuracy value of zero (i.e., $acc = 0$) for our research.

Finally, we also tuned the ker parameter to see what is the value that performs the best in terms of fitness (i.e., time of flight). We have benchmarked four different ker values: 30, 50, 75, 100. The average and best results over three runs with controlled seeds are shown in Figure 7.3.

As we can observe, the results with $ker = 100$ are the best in terms of the best fitness and the second best in terms of average fitness. Nevertheless, we cannot establish a clear winner both in terms of the best fitness values and convergence speed to reach these values. Indeed, lower ker sizes seem to reach lower fitness values faster, but they also seem to get stuck more easily. This behavior could be foreseen as lower ker sizes mean that the algorithm will focus more on generating new individuals around a few best ones. This will, of course, boost the convergence speed while weakening the diversity. It will thus be easier to get stuck in local minima, but it will also be easier to converge faster towards a certain minimum.

All things considered, we have thus decided to prefer a ker size of 100, which corresponds to be the same as the population size. Therefore, as a general rule for this research, we will always maintain a solution archive size as big as the population size (e.g. if population size is 200, then $ker = 200$).

7.1.2. MHACO Tuning

In this section, we will discuss how we tweaked and tuned the input parameters of the MO MHACO algorithm. In principle, each optimization problem requires input values to be chosen accordingly. However, this often means very long computation time as the problem has to be run over and over for each different input set. Therefore, very often the user prefers self-adaptive algorithms (i.e., algorithms that can adapt their input parameters throughout the optimization process) or algorithms where little tweaking and tuning are required. Although MHACO has been tested on several problems, and its capabilities have been thus studied in diverse sets of problems (which are presented in Appendix B), it is still a new algorithm and we thus do not know whether its input parameters have to be thoroughly adjusted for each problem, or if little changes have to be made. Thus, we have decided to use the knowledge acquired from the validation tests and some algorithm insights coming from the general construction of the algorithm to narrow down a set of different input parameters to be traded-off for the solar sailing polar mission. This will help to establish an ideal set of input parameters for MHACO, which can then be used for when, in Section 7.3, this algorithm will be compared with other MO optimizers in the solar-sail mission. As we have already pointed out in Section 5.3.3, this algorithm has six input parameters. Most of them have a similar working principle as those of its single-objective counterpart (i.e., ACOmi). As a matter of fact, for most of the problems q can be set to 1 and the focus parameter to 0. Also, in our case, we are not interested in stopping the optimization process before the end of the generations, and we thus do not trigger the $evalstop$ parameter. We are thus left with only three parameters

Table 7.1: nine tested different sets of input parameters for MHACO algorithm.

Options	$NGenMark$	$threshold$	ker
Option 1	7	25	pop_{size}
Option 2	10	25	pop_{size}
Option 3	12	25	pop_{size}
Option 4	14	25	pop_{size}
Option 5	12	35	pop_{size}
Option 6	7	35	pop_{size}
Option 7	7	35	20
Option 8	8	35	40
Option 9	8	35	20

Table 7.2: Joint hypervolume values w.r.t. the reference point, for MHACO algorithm with nine different options for the input parameters.

Options	pop= 56	pop=120	pop=220
Option 1	477372.67×10^6	486743.41×10^6	472589.17×10^6
Option 2	478128.68×10^6	485813.03×10^6	481644.95×10^6
Option 3	474189.01×10^6	485813.03×10^6	488377.64×10^6
Option 4	477862.30×10^6	488159.23×10^6	486466.86×10^6
Option 5	468965.80×10^6	489236.12×10^6	478474.98×10^6
Option 6	467075.85×10^6	473465.33×10^6	474939.19×10^6
Option 7	478805.27×10^6	501915.54×10^6	500235.87×10^6
Option 8	484182.78×10^6	484680.88×10^6	498216.61×10^6
Option 9	494756.29×10^6	499735.20×10^6	506573.98×10^6

to tweak: ker, $threshold$ and $NGenMark$. Based on our verification and validation experience with the algorithm, we have decided to test five different $NGenMark$ values and two different $threshold$ values. In particular, $NGenMark$ was varied between 7, 8, 10, 12 and 14, whereas $threshold$ was varied between 25 and 35. Furthermore, the ker parameter was set to either a small number (i.e., 20 or 40) or it was chosen as big as the population size: this was done because for the SO ant colony optimizer that we developed, when the ker parameter was traded-off, it was verified that either a small value or the biggest possible (i.e., as big as the population size) typically perform better. We thus constructed nine different input sets: each of these is shown in Table 7.1.

Each algorithm was run three times with controlled seeds (for removing randomness and for having the same initial population set) with three different population sizes (i.e., 56, 120 and 220 individuals). The hypervolume values and the front progressions have been used to determine the best input parameters for the algorithm among the ones tested. The hypervolume values have been computed on all the three final runs, for each population size. While the reference point was set to be the same for all the different algorithms and population sizes: so that the results can be comparable for the different population sizes. Its coordinates have been chosen to be: (500 kg, 3×10^8 s, 10×10^3 m/s, 0.001 s), where the first coordinate represents the sailcraft mass (m), the second the total time of flight (tof), the third the final ΔV and the fourth the final time constraint violation (ΔT_{final}). Of course, since the problem has four objectives, the reference point is four-dimensional.

The results of these nine options in terms of joint hypervolume values of the Pareto-optimal solutions found with the different population sizes, over three runs, are reported in Table 7.2, and shown in Figure 7.4.

Also, in Figure 7.5, the best Pareto front individuals number as a function of the number of generations is shown. This represents an average over the three runs and is plotted for three different population sizes (i.e., 56, 120 and 220).

By looking at these results, some conclusions can be drawn. First of all, in terms of joint hypervolume values, Option 9 performs as the best for two population sizes (i.e., 56 and 220),

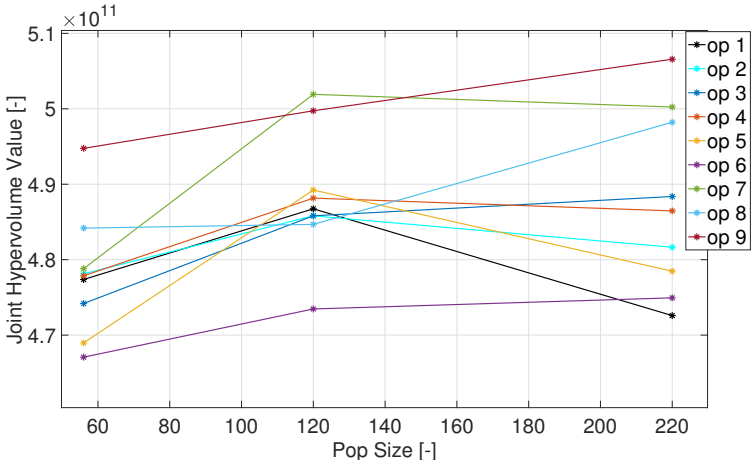
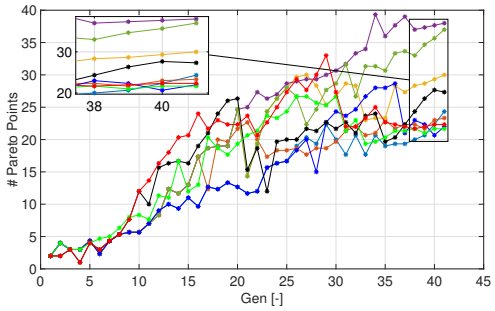
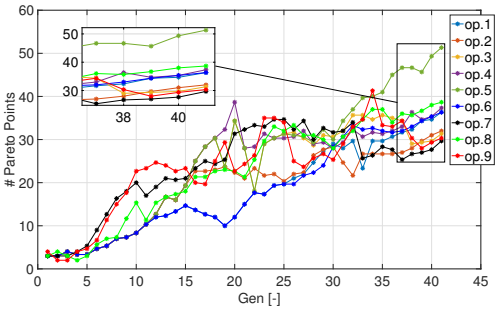


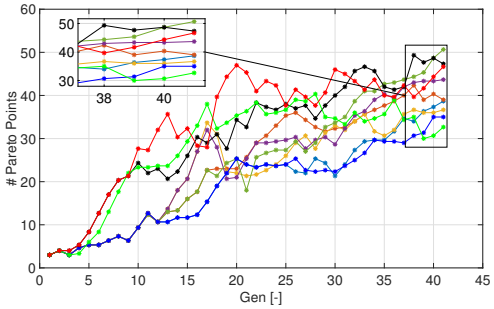
Figure 7.4: Graphical representation of the joint hypervolume values reported in Table 7.2. These values are shown for all the nine options and as a function of the population sizes.



(a) 56 individuals and 40 generations.



(b) 120 individuals and 40 generations.



(c) 220 individuals and 40 generations.

Figure 7.5: Average number (over three runs) of Pareto points throughout the generations for nine different MHACO algorithm input sets. The results are shown as a function of the number of generations.

while for the population size of 120, Option 7 outperforms it, although Option 9 is still the second and not far from the best hypervolume value. Therefore, the lower the *ker* values are, the better the solutions seem to be. However, when looking at the front progression the results are completely different. Indeed, there it seems that Option 5 performs in the best way (whereas in terms of hypervolume value is one of the worst for all the population sizes). This might suggest that the algorithm gets stuck in local optimal fronts in the case of Option 5, whereas for the other cases the algorithm still keeps searching. As a further step for a future study, it could be interesting to verify whether this situation changes or stabilizes when the generations are increased, to also better understand whether the aforesaid hypothesis could correspond to the truth.

The greatest contributors to the hypervolume values, for the population sizes of 56, 120

and 220 are the following individuals, respectively: (129.12 kg, 1988 JD, 3543.21 m/s, 0 s), (178.18 kg, 1972.81 JD, 3543.21 m/s, 0 s) and (129.35 kg, 1975.56 JD, 1335.96 m/s, 0 s). It has to be noted, however, that the greatest contributors are not necessarily the best solutions. Indeed, the single contributions also account for how cluttered the points are. A solution that is located with many others in the front, for instance, will result in a smaller contribution to the hypervolume w.r.t. another one far away from the others. This means that the greatest contributor is often a point quite isolated in the front, which does not always result to be the best for our physical solution. For instance, by looking at the greatest contributors, in this case, we notice that they all have a quite high ΔV and mass, but a low tof. This is related to the fact that a few solutions exist with these characteristics in the Pareto-optimal front. Nonetheless, if we analyze the data of the best front (i.e., the one of Option 9 for 220 individuals), we find, for instance, an individual that has: (104.09 kg, 2076 JD, 97.46 m/s, 0 s). This individual will generate an orbit with nearly 20 kg less mass and an almost perfect final operational orbit w.r.t. that of the greatest contributor, although it will require nearly 100 more days for reaching that final operational orbit. It is clear that determining which of these solutions is better depends on the specific characteristics of the problem. Therefore, a trade-off between the different individuals in the front and their objectives values shall be made by the mission analysis team. Indeed, it is always useful to look at the entire front while analyzing and comparing the individuals within that front, rather than taking the greatest hypervolume contributor as the best solution. This aspect will be treated in more detail in Section 7.3, where a few candidate best solutions will be selected and benchmarked for the solar sailing polar mission.

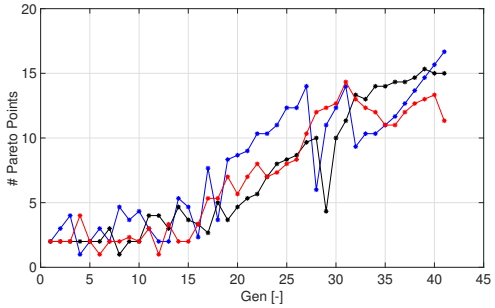
7.1.3. NSPSO Tuning

Similarly to what has been done in Section 7.1.2, we will now tweak and tune the NSPSO algorithm for figuring out what are the ideal parameters to be used in our optimization process. Since this algorithm is only used for MO problems, the tuning phase will be performed on the solar-sail MO problem discussed in Section 5.4. Nevertheless, before this tuning and tweaking phase on the solar-sail mission of our interest, the verification and validation of the algorithm on several MO test problems have been carried out. The results of these tests are discussed in Appendix B. These tests, together with the theoretical studies coming from literature, have been pivotal for both having an idea of the possible performances of this algorithm when compared to NSGA-II and MOEA/D, and for understanding how to select the input parameters of the algorithm, depending on the type and dimension of the problem.

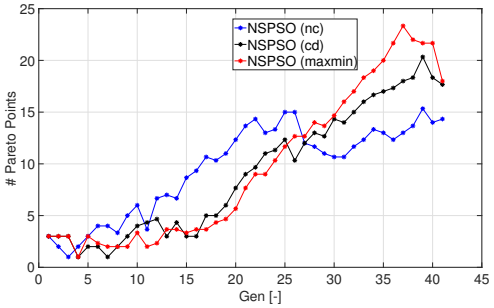
As a consequence, the only NSPSO input parameter that will be tweaked is the employed diversity strategy. As explained in Section 5.1.2.1, this algorithm has three possible strategies to be chosen: niche count, crowding distance and maxmin function. Each of these will substantially change the algorithm behavior: for our problem, it is thus important to understand which one performs the best. As far as concerns the other parameters (i.e., ω, χ, η_1, η_2, V_{MAX}, LSR): their value has been chosen either according to the original paper where NSPSO was introduced and discussed or according to the results we obtained from the test problems. The input parameters used for this benchmark study are: ω =0.4, χ =1.0, η_1=2.0, η_2 =2-0, V_{MAX}=0.5, LSR =50.

For tweaking the three different diversity strategies used, we followed the same methodology used for MHACO: we monitored the front progression and the hypervolume values of these algorithms when they are run with three population sizes (i.e., 56, 120 and 220). Each population size is run three times, with three different controlled seeds: so that the randomness is canceled and the initial populations of each algorithm with different diversity strategies are always the same when the seed is the same. The average (over three runs) front progressions for the three population sizes are shown in Figure 7.6. While in Table 7.3 the joint hypervolume computations are shown for all the population sizes and algorithm strategies. In particular, these hypervolume values have been computed for all the three final population sets of each population size and each strategy.

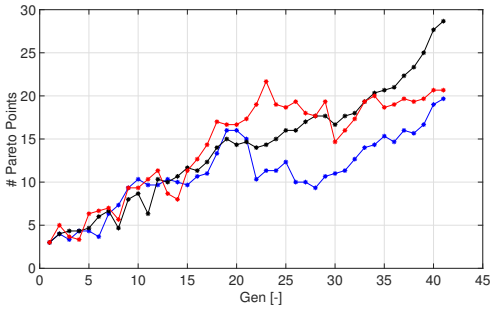
Also, the reference point has been chosen to be the same for all the different algorithms and population sizes: so that the results can be comparable also across different population sizes. The reference point is set to be the following: (500 kg, 3×10^8 s, 5000 m/s, 0.001 s),



(a) 56 individuals and 40 generations.



(b) 120 individuals and 40 generations.



(c) 220 individuals and 40 generations.

Figure 7.6: Average number of Pareto points throughout the generations for NSPSO algorithm with three different diversity strategies (niche count, crowding distance, maxmin function).

Table 7.3: Joint hypervolume values w.r.t. the reference point, for NSPSO algorithm with three different diversity strategies (niche count (NC), crowding distance (CD), maxmin function (MaxMin)).

Diversity Strategy	pop= 56	pop=120	pop=220
NC	210216.75×10^6	228613.80×10^6	220412.32×10^6
CD	220412.32×10^6	219146.81×10^6	224427.84×10^6
MaxMin	211557.15×10^6	220291.27 ×10^6	223243.22×10^6

where the first coordinate represents the sailcraft mass (m), the second the total time of flight (tof), the third the final (ΔV) and the fourth the final time constraint violation (ΔT_{final}).

As we can see from the hypervolume values, the NSPSO algorithm with niche count has the overall best performance, which happens for 120 individuals, but it is outperformed by the other two diversity strategies when looking at the 56 and 220 population sizes alone. Indeed, for both 56 and 220 individuals, NSPSO with crowding distance performs in the best manner. Furthermore, we observe a generally improving trend (in terms of hypervolume values) when the population size is increased, although this is not always true (e.g. NSPSO with crowding distance worsens from 56 to 120 individuals). The individual that contributes the greatest to the hypervolume for NSPSO with niche count and 120 individuals is the following: (m = 104 kg, tof = 2202.63 JD, ΔV = 114.43 m/s, ΔT_{final} = 0 s). While if we check the greatest contributors for the highest hypervolume values of both 56 and 220 individuals (which are both generated by NSPSO with crowding distance), we find the following two points: (m = 197.55 kg, tof = 2426.04 JD, ΔV = 119.61 m/s, ΔT_{final} = 0 s) and (m = 104 kg, tof = 2543.93 JD, ΔV = 97.453 m/s, ΔT_{final} = 0 s), respectively.

When confronting the average Pareto fronts progressions (over three runs), the NSPSO with crowding distance seems to perform very well. Indeed, as we can see from Figure 7.6, it always reaches the highest final numbers of Pareto points values (as it happens for 120 and 220 individuals), or the second highest (as it happens for 56 individuals). Also, it seems to be the algorithm that increases more steadily the number of Pareto points throughout the generations, while the others either display a more accentuated wobbling, or they lose their

Table 7.4: Optimizers trade-off in the geocentric phase (for three different function evaluation sizes: 2050, 4100, 6150 and 8200, by keeping the generations number fixed to 41).

	Fevals	**ABC**	**DE**	**DE1220**	**PSO**	**SADE**	**SGA**	**ACOmi**
tof_{best} [JD]	2050	232.13	242.20	241.53	230.43	236.80	238.96	231.51
	4100	238.18	241.63	231.35	231.21	238.67	239.04	233.80
	6150	234.01	230.98	232.63	233.16	233.31	235.07	229.11
	8200	232.28	231.52	235.22	232.33	234.09	229.56	232.61
tof_{ave} [JD]	2050	236.81	245.78	242.20	232.63	240.94	246.35	235.77
	4100	243.35	244.16	237.80	231.82	241.64	244.67	234.42
	6150	235.21	236.92	236.08	233.56	235.66	238.22	232.88
	8200	233.09	235.24	237.40	232.46	237.32	236.90	237.88

rising rate at higher generations, thus displaying a flatter curve for high generations (as it seems to be the case for NSPSO with niche count, especially for 56 and 220 individuals). All things considered, we have thus decided to choose NSPSO with crowding distance for population sizes of 56 and 220 individuals, for making the comparison and trade-off against MHACO, NSGA-II and MOEA/D and establish the overall best solutions of these algorithms in the MO solar sailing polar mission. While for the case of 120 individuals, NSPSO with niche count was chosen to be compared with the other algorithms. The results and discussions of such benchmarks are presented in Section 7.3.

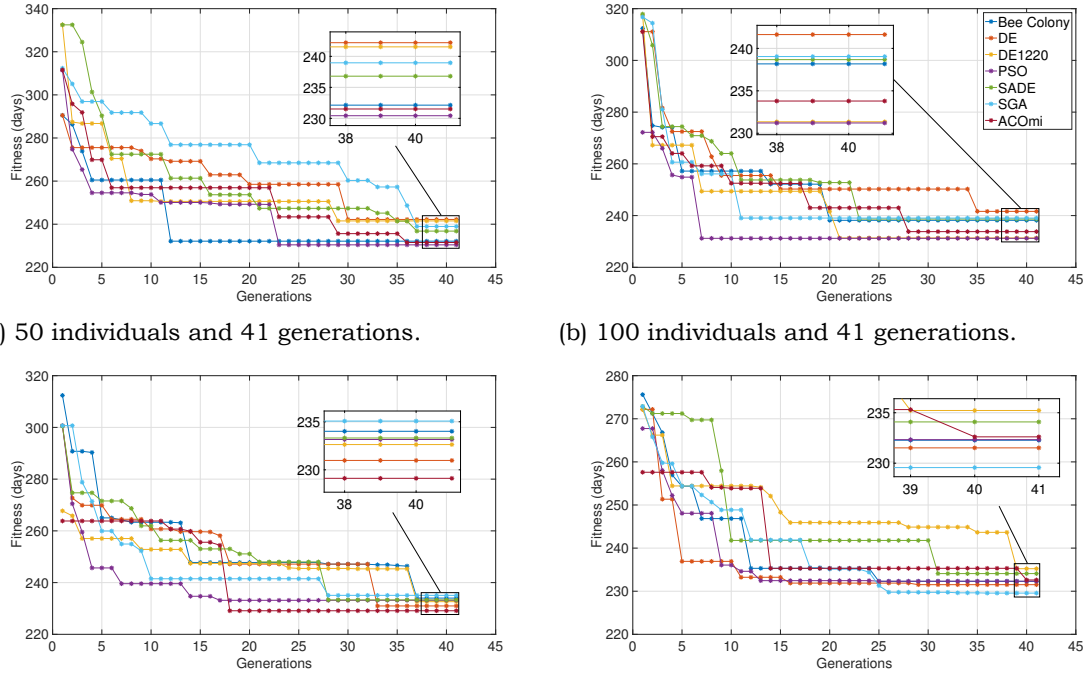
7.2. Single-Objective

In this section, we will discuss the results of the optimization procedure applied to the SO solar-sail mission. As we have already mentioned in Chapter 5, in such a mission we will only optimize the time of flight (tof). In particular, we will first treat the SO problem as two separate problems: an unconstrained SO problem for the geocentric phase and a constrained SO problem for the heliocentric phase. These two optimization procedures are presented in Sections 7.2.1 and 7.2.2, respectively: in both cases, the tof is the objective to be minimized. Then, in Section 7.2.3 we will optimize the geocentric and heliocentric phase as a unique SO constrained problem. Finally, in Section 7.2.4, the results coming from the two different strategies will be compared and traded-off. All these optimizations will be executed using several different SO optimizers. Most of these are well known and have been extensively used in the space sector (Vinkó and Izzo, 2008), (Izzo, 2010), (Izzo, 2007).

7.2.1. Geocentric Phase

In this section, we only study the geocentric phase. Therefore, we have an SO unconstrained problem with box-bounded variables. In this framework, we use several different optimizers to find the best sets of these variables, which minimize the time of flight. In particular, we run each optimizer for three times for removing the randomness that is inherent in these metaheuristic algorithms. Furthermore, every run is executed with the same seed, so that all the algorithms always start from the same initial population. As we can observe from the plots in Figure 7.7, six different optimizers are used (i.e., Artificial Bee Colony (ABC), Differential Evolution (DE), Differential Evolution 1220 (DE1220), Particle Swarm Optimization (PSO), Self-Adaptive Differential Evolution (SADE), Simple Genetic Algorithm (SGA), mixed integer Ant Colony Optimizer (ACOmi)): each of these was compared in terms of minimum tof value, as a function of the generations used for evolving the algorithm. Also, four different population sizes were implemented (i.e., 50, 100, 150 and 200). The results, shown in the figures, represent the best runs of each of the optimizers. The absolute best optimal value of time of flight is found using the ant colony optimizer (ACOmi), which also performs very well in general, showing a steady improving behavior with often the best or one of the best final optimal values of tof. The final average and best values of the different algorithms for different population sizes are shown in Table 7.4.

It has to be noted that all the algorithms have executed their number of function evaluations in 41 generations and with a population size of 50, 100, 150 and 200 individuals. The



(a) 50 individuals and 41 generations.

(b) 100 individuals and 41 generations.

(c) 150 individuals and 41 generations.

(d) 200 individuals and 41 generations.

Figure 7.7: Optimization of geocentric phase for different population sizes and optimization algorithms

only exception is the artificial bee colony, that executes twice the number of the individuals function evaluations per generation. Hence, if a population size of 50 is chosen, the algorithm will perform 100 of function evaluations after one generation. For avoiding to have confusing results, we have uniformed the ABC's plot to the others by splitting the number of individuals per generation by two. Indeed, the main driver for the algorithm performance both in terms of accuracy and computation time is the function evaluation value: this is found by multiplying the number of individuals by the number of generations and thus results to be the same for all our tested algorithms.

As we can observe from Table 7.4, in general, the results do not always improve when the function evaluations are increased. The reason is that most of these algorithms manipulate the population for evolving new individuals: hence, by increasing the population size, we change the behavior of the algorithm, which, in turn, might reduce its convergence speed or worsen its performance. This aspect can be clearly seen in Figure 7.8, in which the best results for the various algorithms (in terms of minimum tof) are plotted as a function of the function evaluations (when the number of generations is kept fixed).

On the other hand, if the same population size is maintained and the generations are increased, we might expect to have either the same or better results. This is not, however, the case for the algorithms that have an internal mechanism that makes use of the number of generations: in this case, increasing the generation number does not strictly mean an improvement in performance since the algorithm changes its behavior (this is the case for ACOmi, for instance). For demonstrating this behavior, we have decided to run PSO and ACOmi with the same number of function evaluations (i.e., 2050, 4100, 6150, 8200), but fixing the number of population to 50 individuals and only varying the number of generations (i.e., 40, 80, 120 and 160). The results are shown in Table 7.5.

In particular, we observe that while PSO improves, on average, when the generations are increased (with the exception for the case of 41 generations and 50 individuals, which shows the minimum for PSO in terms of the time of flight), this does not happen for ACOmi. Moreover, we notice that PSO has a more regular behavior when the generations are increased and the population size is kept fixed, contrary to what happened when the generations were kept fixed and the population size was varied. However, the best result still remains that

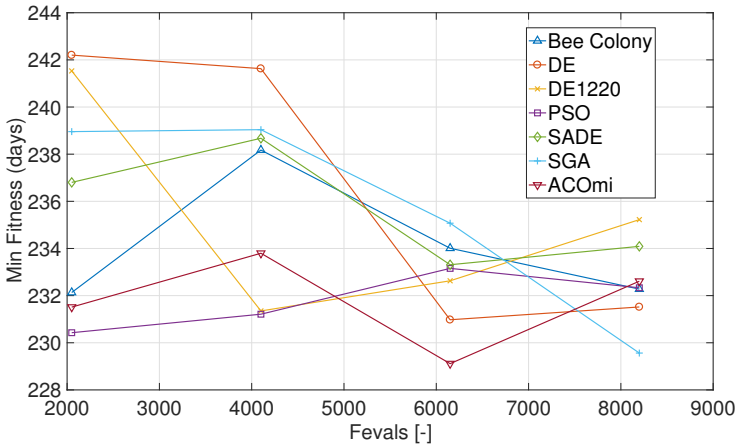


Figure 7.8: Best final time of flight of the optimization algorithms as a function of the function evaluations (for a generation size fixed to 41).

Table 7.5: Optimizers trade-off in the geocentric phase (for three different function evaluation sizes: 2050, 4100, 6150 and 8200, by keeping the population size fixed to 50 and varying the generation sizes between 40, 80, 120 and 160).

	Fevals	**PSO**	**ACOmi**
tof_{best} [JD]	2050	230.43	231.51
	4100	231.23	236.64
	6150	231.04	235.92
	8200	230.92	231.91
tof_{ave} [JD]	2050	232.63	235.77
	4100	233.90	239.35
	6150	232.04	237.07
	8200	232.01	233.41

of 40 generations and 50 individuals. Hence, for both PSO and ACOmi an increase in the function evaluations has not brought an increase in the performances (in both cases in which either the generations or the population sizes were increased). For ACOmi this kind of unpredictable behavior can be explained by looking at the mathematical formulation of the optimizer. Indeed, this algorithm has an internal mechanism based on two parameters (i.e., the $threshold$ and the generation mark, $NGenMark$, input parameters) that are strictly related to the number of generations. Indeed, if the threshold is chosen to be 20, for instance, then the algorithm will start focusing more on the best individuals of the archive already from the 20th generation. This might be powerful if we are running 200 individuals for 40 generations (because, perhaps, the very best individuals found at the 20th generations are already candidate solutions, and it thus might result worthy to focus on improving them, rather than keeping the search more spread). However, if we are running 50 individuals for 160 generations (which corresponds to the same function evaluation number) this might not be ideal since it might cause the algorithm to get stuck on local optima.

In conclusion, as we have already pointed out, from the various results (by testing different population sizes and by also varying the generations' size), we can conclude that the absolute best results are found with the ant colony optimizer (with 150 individuals and 41 generations). Moreover, we have observed that PSO seems to perform very well overall, as it reaches the third best absolute minimum value at only 2050 function evaluations, and it displays a quite low average among the different function evaluation sets tested. We have thus decided to test these two algorithms with a higher population size of 300 individuals (by maintaining 41 generations), to see whether these results could improve even further if much bigger function evaluations are performed. Surprisingly, as can be seen in Table 7.6, for ACOmi we do not observe any improvement with respect to the absolute best and average

Table 7.6: PSO vs ACOmi for 300 individuals and 41 generations.

	Fevals	PSO	ACOmi
tof_{best} [JD]	12300	229.55	235.15
tof_{ave} [JD]	12300	231.26	237.69

Table 7.7: Best solution found with ACOmi optimizer (150 individuals and 41 generations).

Symbol	**ACOmi Best**
T_{launch}	755970192 s
ω_0	359.41°
α_{E1}	26.06°
α_{E2}	7.68°
α_{E3}	0.08°
tof	229.11

best, whereas for PSO we find an absolute best of 229.55 Julian Days, which is less than 1% worse than the absolute best found with ACOmi using 150 individuals.

Analyzing the results, it seems that increasing too much the number of individuals does not bring substantial improvements for the ant colony optimizer. This can be explained by noticing that together with the population size, we are also increasing the kernel size (i.e., the number of individuals stored in the solution archive). This means that more individuals are used for generating new ants for future generations, and this might cause these individuals to be more spread and less focused around the found optimal solution. Even though this might be an advantage in some cases (i.e., when there are multiple local minima or when the minimum is very hard to find), this is however not ideal when we want to reach a certain area where the minimum is located and then narrow down the search to a smaller portion of the domain. Indeed, such a big kernel size causes the search always to be quite spread, thus partially impeding local convergence.

Due to the fact that many function evaluations still have not been able to improve the overall best found solution with ACOmi for only 6150 function evaluations, we have decided to proceed to the local Monte Carlo improvement using such a solution. In the following section, this local refinement will be discussed: its purpose is to try to enhance the best solution by slightly varying the optimal solution's variables.

7.2.1.1 Local Refinement: Geocentric Phase

As we have already pointed out, the best overall solution is found using ACOmi with 150 individuals and 41 generations. In Table 7.7, the time of flight (in Julian Days) together with all the optimized variables corresponding to it, is shown. The time shown in the table (i.e., T_{launch}) is expressed in Julian seconds; this corresponds to the following launch date: 2023-12-16 at 03h:43m:12s [1].

By running this solution separately in the simulator, we verified that it is a feasible solution, and that the sailcraft safely (i.e., without impacting the Moon and without decreasing its altitude w.r.t. the Earth too much) escapes the Earth gravity. However, before proceeding to the heliocentric phase with this solution, we have decided to perform a local refinement using a Monte Carlo technique around this best solution, for checking whether this is actually the best, or if a faster trajectory can be found in the nearby search space. In particular, each variable was box-constrained in a 1% range and a Monte Carlo run was performed using a uniform distribution for varying each variable. This distribution was chosen because we do not have any physical insights that can allow us to use any particular distribution. The runs were performed three times with 15,000 function evaluations each and the resulting times of flights as a function of the number of function evaluations are shown in Figure 7.9.

As can be seen, the solutions found with the Monte Carlo technique are quite well spread,

[1] https://nsidc.org/data/icesat/glas-date-conversion-tool/date_convert/, date of access: August, 2019

Table 7.8: Best solution found with MC local refinement of the geocentric phase.

Symbol	MC Best
T_{launch}	755940988.8 s
ω_0	358.737°
α_{E1}	25.92°
α_{E2}	7.69°
α_{E3}	0.08°
tof	228.45 JD

besides two gaps that form between 230 and 250 Julian Days and slightly below 260 Julian Days. In this histogram only the feasible solutions are shown, meaning that the solutions that have violated the conditions imposed to the orbit (i.e., crashing onto the Moon's surface or going too near to the Earth or not reaching the sphere of influence of the Earth within a limited amount of time) are excluded. However, by checking the average number of feasible solutions over three runs, we observe that these are 98.62% of the total. Hence, these gaps might be caused by the fact that within those variable bounds (i.e., 1% of the optimal value), there are no solutions that produce those times of flights' values. In any case, further investigation of these gaps is not very interesting for our study, since they happen to be at quite high fitness values w.r.t. the optimal values of interest. Performing the MC refinement, we managed to improve the tof by 0.29%. Indeed we managed to find a new optimum with a tof value of 228.45 Julian Days. Its corresponding best variables are shown in Table 7.8. With this solution, we can thus extract the initial conditions to be used for the optimization of the heliocentric phase, which will be performed in the following section.

7.2.2. Heliocentric Phase

As we have pointed out in the previous section, the best found solution of the geocentric phase can be used for extracting the initial conditions to be used for the heliocentric phase. In particular, the final ephemeris time and the final sailcraft state variables' vector of the geocentric phase are needed for this to happen. By simulating the geocentric phase with the best found variables, we find the initial conditions for the heliocentric phase shown in Table 7.9, where T_{i} indicates the initial ephemeris time in second (which corresponds to the following date: 2024-07-31 09h:45m:41s), and it is found by summing the tof and the launch time in seconds of the best geocentric phase. The variables x_{i}, y_{i}, z_{i}, $\dot{x}_{\text{i}}$, $\dot{y}_{\text{i}}$, $\dot{z}_{\text{i}}$ represent the final position and velocity of the sailcraft when the geocentric phase is stopped, expressed in Cartesian coordinates. The propagator used for this mission is the USM Rodrigues propagator. For the

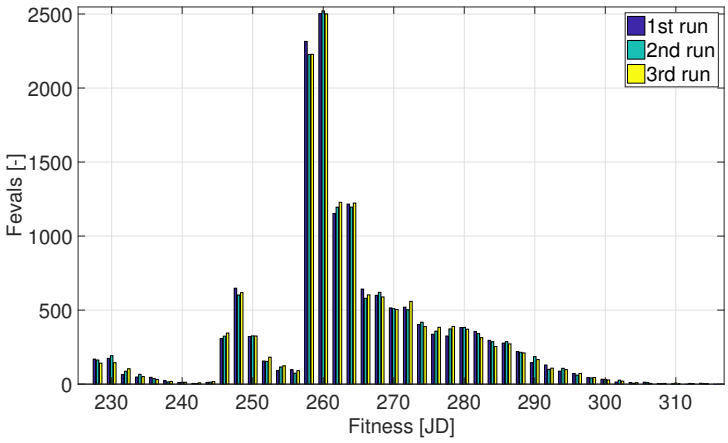


Figure 7.9: 1% local MC refinement within for the best solution in the geocentric phase (time of flight vs function evaluations for three different runs).

Table 7.9: Initial conditions for the heliocentric phase.

Symbol	Initial Conditions
T_i	775691141.52 s
x_i	94909925.09 km
y_i	-118495784.90 km
z_i	75147.28 km
$\dot{x}_i$	23.15744 km/s
$\dot{y}_i$	19.41567 km/s
$\dot{z}_i$	0.12347 km/s

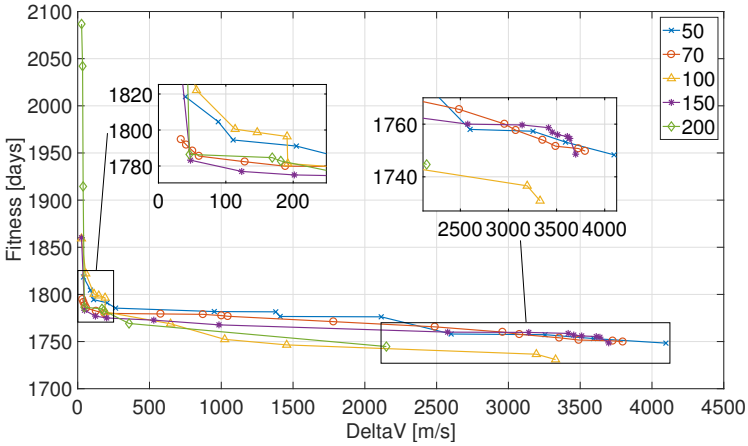


Figure 7.10: ACOmi population size trade-off for the heliocentric phase.

heliocentric case, we have decided to apply the USM Rodrigues propagator (instead of the MEE propagator) because we have observed from the propagator trade-off study in Section 6.3.1, that the USM Rodrigues propagator is the best in terms of quality of solutions (for the solar sailing polar mission of our interest), but it has the main disadvantage that it has a way higher wall-clock time. In this case, we are not worried about this second aspect, since the heliocentric phase is less computationally expensive than the geocentric phase (i.e., less than one second per function evaluation versus nearly twenty), thus making the wall-clock time difference between propagators negligible.

In this case, the problem is a SO constrained one, however, all the algorithms used in the geocentric phase (except for ACOmi) cannot handle constraints. Nonetheless, as already pointed out in Chapter 5, a meta-algorithm exists to modify these algorithms in such a way that constraints can also be handled. Hence, this meta-algorithm is applied for three single-objective algorithms (i.e., ACOmi, PSO and DE1220) for allowing their application to this problem. These algorithms have been selected due to their good performances demonstrated in the optimization of the geocentric phase, across different population sizes. The results found with these algorithms will be confronted with those found with ACOmi alone. It has to be noted that ACOmi can be applied both without and with the meta-algorithm: this allows us to confront these two techniques and establish whether it is better to use ACOmi alone, or to couple it with the meta-algorithm. The constraints to be handled are the final time violation and the ΔV (which represents one of the two constraints, that we would ideally like to be zero). Before confronting these algorithms, a study is made for establishing the correct population size to be used for ACOmi, when used alone. In particular, the generation size is kept fixed at 41 and the population size is varied between 50, 70, 100, 150, 200. The results of this trade-off are plotted in terms of the time of flight (in Julian Days) versus the ΔV. These are shown in Figure 7.10.

Only the solutions that fulfill the final time constraint (i.e., final time violation equals to zero) are represented in these Pareto fronts, whereas the others are excluded. It can be seen

from this plot that higher population sizes offer more advantages in this optimization. Indeed, if for instance we observe the area on the left, the two most interesting points (i.e., those that keep the tof at low values while also minimizing the ΔV) are found with a population size of 200 (i.e., tof=1786 JD and ΔV=46.03 m/s) and of 150 (i.e., tof=1783 JD and ΔV=47.55 m/s). One could notice that the nearby solution with 70 individuals (i.e., tof=1785 JD and ΔV=60.27 m/s) is not much worse than the aforementioned ones, and due to the fewer function evaluations, should be preferred. However, this can be objected with two main points:

1. The heliocentric phase is much less computationally expensive than the geocentric phase, due to the absence of atmosphere and less dynamically varying environment (i.e., the wall-clock time for 2050 function evaluations is only 500 seconds against 40,000 seconds of the geocentric phase, and for 8200 function evaluations is only 2500 seconds against 180,000 seconds of the geocentric phase). This means that while the function evaluations may be a bottleneck for the geocentric phase, so that lower population sizes might be preferred, even though they slightly worsen performances, this is not the case for the heliocentric phase, in which the computation time problem is almost absent, due to very fast simulations.

2. For the entire (ΔV, tof) space, we do not observe behavior of low population sizes comparable to that of higher sizes. The 150 and 200 individuals runs seem to explore the domain more efficiently, by reaching more and better solutions in key parts of the graph (i.e., minimizing the objective and the constraint both separately and concurrently).

For these reasons, we have decided to opt for a population size of 200 individuals for the heliocentric phase optimization.

Having done this population size trade-off, we could then compare the three algorithms coupled with the meta-algorithm for handling the constraints (i.e., PSO, DE1220, and ACOmi) and ACOmi alone. Each run was executed three times to remove the intrinsic randomness of these metaheuristic techniques and the same function evaluations were used (i.e., 20,000). Also, each run was performed with a controlled seed, to have the same initial population (generated randomly) for all the algorithms. This aspect is further explained and analyzed in Section 7.4.

The results of this study in terms of ΔV and tof are shown in Figure 7.11. Concerning the ACOmi algorithm alone, for having the same number of function evaluations, a population size of 200 individuals was selected, with 100 generations. While for the other three, 50 individuals with 20 generations and 20 iterations were chosen for the algorithms and the meta-algorithm (the meta-algorithm description can be found in Appendix C).

The figure represents the best Pareto front over three runs of all the solutions of the algorithms. As can be seen, all the algorithms seem to converge to the same front, which means that we probably converged towards the absolute best front. The only exception is the ACOmi optimizer alone, which does not seem to be competitive with the others, reaching a worse Pareto front. In particular, almost all its individuals are dominated by those of the other three algorithms. The DE1220 algorithm seems to have the best performances for low values of ΔV (i.e., below 400 m/s), thus making its solutions very interesting. For instance, two Pareto dominant solutions, one with tof=1776 JD and ΔV=23.67 m/s, and the other one with tof=1775 JD and ΔV=35.64, seem to be very appealing. Conversely, for higher ΔV's the ACOmi algorithm coupled with the meta-algorithm seems to perform in the best way, also managing to reduce the most the fitness value (finding a minimum tof of 1707 JD but for the highest ΔV of 3293 m/s). Another interesting solution of ACOmi is that of tof=1754 JD and ΔV=403.5 m/s, which seems to be an acceptable trade-off between a low time of flight and a reasonably low ΔV violation.

From Figure 7.11, we can extract several best solutions: indeed, contrary to the geocentric phase, in this case, we cannot identify a single best solution, but the best solutions will be a trade-off between ΔV and time of flight values. Due to this, we decided to identify three candidate best solutions (two that minimize the two objectives separately and one that minimizes both the objectives concurrently). These three candidate solutions (i.e., Candidate 1, 2 and 3; coming from PSO, DE1220 and ACOmi, all with the meta-algorithm, respectively), together with their corresponding variables and constraint violation values are shown in Table 7.10.

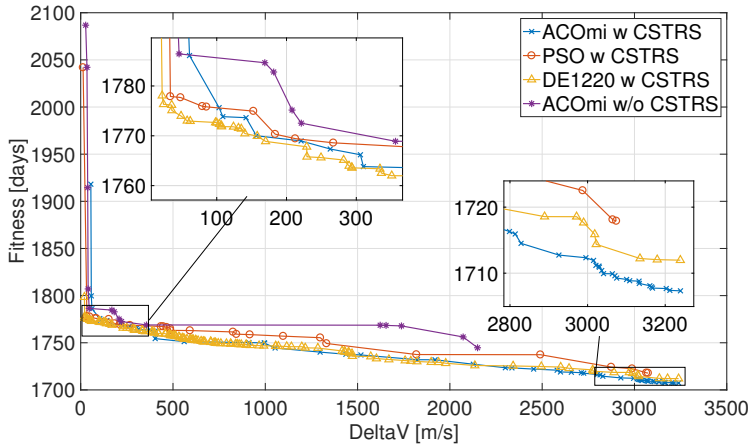


Figure 7.11: Trade-off of different optimizers for the heliocentric phase.

Table 7.10: Three candidate best solutions for the heliocentric phase, and their corresponding fitness values and constraints violations.

Symbol	**Candidate 1**	**Candidate 2**	**Candidate 3**	*Units*
ΔV	12.92	23.67	3239.29	[m/s]
tof	2042	1776	1707	[JD]
Δa	5.3×10^{-6}	1.4×10^{-4}	1.9×10^{-4}	[au]
Δi	8.9×10^{-6}	1.3×10^{-2}	0.08	[deg]
Δe	0.55×10^{-3}	5.5×10^{-4}	0.137	[-]
α_{S1}	-49.57	-35.22	-38.91	[deg]
α_{S2}	-14.98	-31.90	-17.80	[deg]
α_{S3}	37.81	39.73	35.59	[deg]
α_{S4}	57.32	39.36	34.95	[deg]
R_1	0.39	0.36	0.3	[au]
R_2	0.30	0.28	0.26	[au]

Besides the fitness and constraints values and the corresponding variables, in this table, we also show the violations of the final ideal semi-major axis, inclination, and eccentricity. As we have already mentioned in Chapter 5, these three orbital elements are handled through one single constraint (i.e., the ΔV) for accounting the fact that correcting these elements once reached the final orbit has a different cost, depending on the type of orbital element to be adjusted. Nevertheless, it is still useful to understand what are the final orbital elements violations corresponding to a certain value of ΔV.

Before computing the total flight time (also adding the geocentric phase), we decided to run a Monte Carlo simulation around the found best solutions first, to control whether we can improve these results. This will be done in the following section.

7.2.2.1 Local Refinement: Heliocentric Phase

Similarly to what has been done for the geocentric phase, also in this case we run a Monte Carlo simulation around the optimal solutions, by varying each of the variables within a 1% range.

It has to be noted that in this case there are three candidate solutions, and we thus have to perform the runs three times per solution, for a total of nine simulations. Each of these runs will have 20,000 function evaluations, to make sure to sufficiently explore the search space around the best solutions.

In Figure 7.12, we show all the feasible sampled points around the candidate solutions. Furthermore, in Figures 7.13b and 7.13a, we show the histograms that represent the distributions of the ΔV and tof values of the candidate solutions as a function of the runs. First

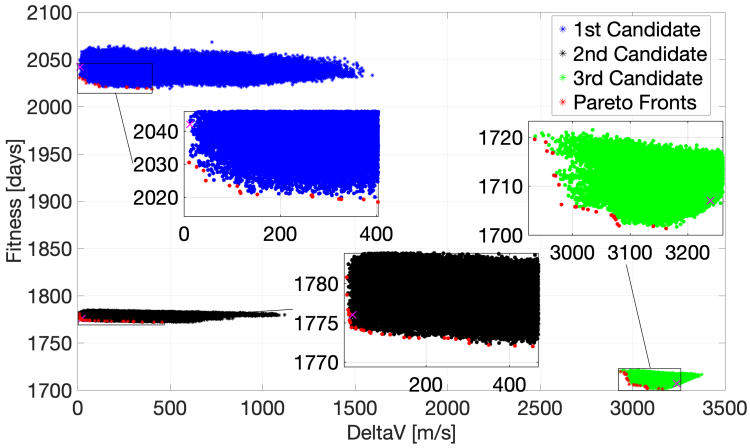
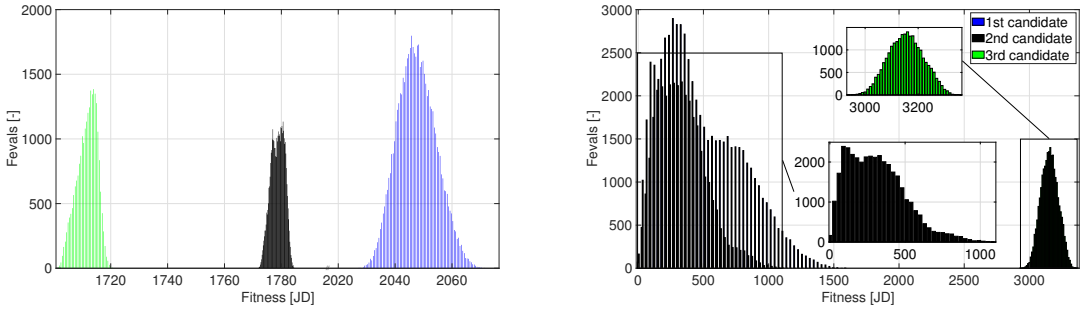


Figure 7.12: MC run around the three candidate solutions of the heliocentric phase. In the figure, also the Pareto-optimal fronts around each candidate solutions have been highlighted.



(a) tof vs number of runs.

(b) ΔV vs number of runs.

Figure 7.13: ΔV and tof distribution as a function of the number of runs for the three candidate solutions of the heliocentric phase.

of all, it is interesting to point out that for the first candidate solution, all the points sampled were feasible. Conversely, for the second candidate solution, 62% of the sampled points were feasible; whereas for the third candidate solution, only 39% of the points were feasible. This also gives us an indication of the sensitivity of each candidate solution (e.g., in the third case, in a real mission scenario, if we miss all the input variables by 1%, we risk to fail the mission with a probability of 61%, which seems to be a risky scenario).

Also, as we can observe from Figure 7.12: from the runs that we executed, we can define a Pareto front around each candidate solution. This permits us to narrow down the choice to only a little number of points. From these simulations, we see that the Candidate 1 can be completely discarded since Candidate 2 reaches the lowest values of ΔV (i.e., 9.389 m/s), while maintaining a way lower time of flight (i.e., around 200 days less): thus causing the Candidate 2 Pareto-optimal front to be dominant w.r.t. that of Candidate 1. Conversely, Candidate 3 (depicted in green in the figure) displays lower time of flights than the other candidates (i.e., around 100 days less than Candidate 2), but it has way higher values of ΔV (in the order of 3 km/s), thus causing the final orbit to be quite far away from the ideal operational one.

All things considered, we have thus decided to select the best solution from the Pareto-optimal front of Candidate 2. In particular, we have decided to choose the solution with the lowest ΔV of 9.389 m/s, and a corresponding time of flight (considering only the heliocentric phase) of 1781 Julian days. The reason is that the Pareto-optimal points of Candidate 2 do not display big differences in terms of flight time (i.e., they all range between 1772 and 1781 days), but they have quite different ΔV. We have thus decided to choose the nearest orbit to

Table 7.11: The best overall solution, coming from the optimization of the geocentric and heliocentric phases separately.

Symbol	**Best Candidate**	*Units*
ΔV	9.39	[m/s]
tof	2009.237	[JD]
Δa	2.88×10^{-5}	[au]
Δi	2.82×10^{-3}	[deg]
Δe	3.0×10^{-4}	[-]
T_{launch}	2023-12-15	yyyy/mm/dd
ω_0	358.737	[deg]
α_{E1}	25.92	[deg]
α_{E2}	7.69	[deg]
α_{E3}	0.08	[deg]
α_{S1}	-35.09	[deg]
α_{S2}	-31.91	[deg]
α_{S3}	39.78	[deg]
α_{S4}	39.60	[deg]
R_1	0.3587	[au]
R_2	0.2847	[au]

the ideal one, sacrificing nine days of flight time. All the parameters of this best solution, as well as its total time of flight (i.e., geocentric plus heliocentric tof summed), is shown in Table 7.11: there, we have also displayed the parameters of the best geocentric orbit used for optimizing the heliocentric phase.

7.2.3. Geocentric and Heliocentric Phase

Having optimized the geocentric and heliocentric phases separately, and having found the best candidate solution, we have decided to also optimize the whole mission as one single problem with 20,000 function evaluations and with the same algorithms used for the heliocentric only phase. The reason why this number of function evaluations was selected was to maintain the same simulation time as the previous two optimizations. Indeed, our objective is to compare this second strategy by maintaining a similar wall-clock time: so that we can estimate the best strategy to tackle the problem based on similar computation times. This builds up a single-objective constrained problem (similar to the heliocentric phase problem), in which we have 11 variables (i.e., α_{E1}, α_{E2}, α_{E3}, ω_0, T_{launch}, α_{S1}, α_{S2}, α_{S3}, α_{S4}, R_1, R_2), one fitness (i.e., tof) and 2 constraints (i.e., the ΔV and the final time constraint). Therefore, we can treat the results similarly to what is done in the heliocentric phase, by plotting the Pareto front in the (ΔV, tof) space of only those solutions that fulfill the final time constraint violation. This plot can be seen in Figure 7.14. As we can see, DE1220 with the meta-algorithm seems to outperform all the other algorithms throughout almost the entire search space. Due to the fact that these algorithms are metaheuristic their performances are really problem dependent and cannot be mathematically forecasted before. Indeed, as can be seen, while in the geocentric mission the ACOmi algorithm outperformed the others, in this case, DE1220 performs better: this is just related to the different nature of the problem and the different fashion in which the two optimization algorithms act. In general, it is not possible to determine a metaheuristic global optimizer that outperforms the others in any kind of problem. Another aspect to be considered are the number of function evaluations: perhaps, increasing this number would allow the other algorithms to all converge to the same or even better Pareto optimal front. However, since we wanted to compare this mission scenario with the aforementioned one, based on a similar wall-clock time (which is highly influenced by the number of function evaluations), we decided to keep the number of function evaluations to 20,000 and compare the two mission scenarios based on nearly the same computation times. From Figure 7.14, we can extract three overall best candidate solutions, which all come from the DE1220 algorithm with meta-algorithm. The three candidate solutions are shown in Table

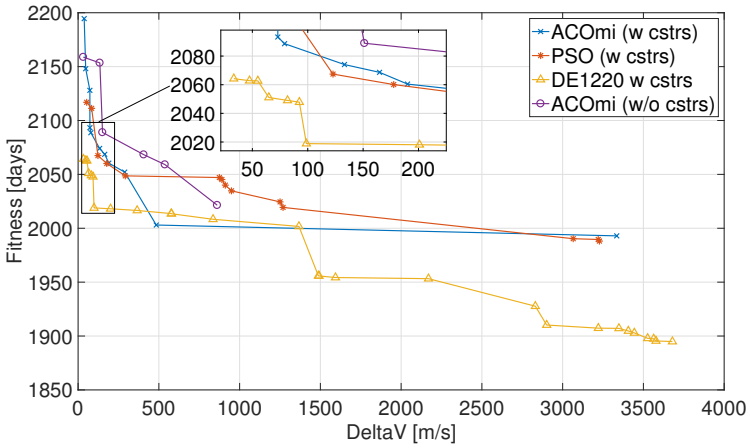


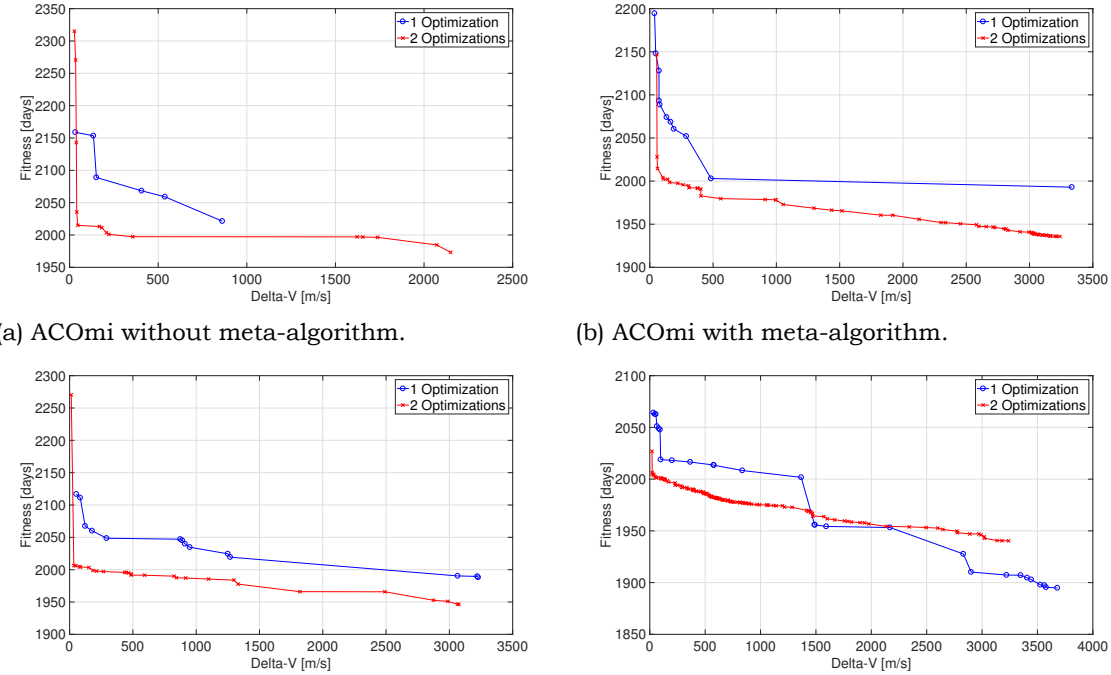
Figure 7.14: Trade-off of different optimizers for the geocentric and heliocentric phases, optimized in one single SO problem.

Table 7.12: Three candidate best solutions for the geocentric and heliocentric phases optimized together, and their corresponding fitness and constraints violations values.

Symbol	**Candidate 1**	**Candidate 2**	**Candidate 3**	*Units*
ΔV	32.41	98.76	3679.58	[m/s]
tof	2159	2019	1895	[JD]
Δa	1.32×10^{-4}	2.15×10^{-4}	1.55×10^{-3}	[au]
Δi	2.4×10^{-3}	8.7×10^{-3}	4.3×10^{-2}	[deg]
Δe	1.3×10^{-3}	3.9×10^{-3}	1.6×10^{-1}	[-]
T_{launch}	2026-10-16	2022-12-21	2027-03-02	yyyy/mm/dd
ω_0	6.20	5.73	6.15	[deg]
α_{E1}	18.95	31.89	27.20	[deg]
α_{E2}	2.69	4.44	21.03	[deg]
α_{E3}	12.20	6.32	12.88	[deg]
α_{S1}	-42.26	-37.99	-36.64	[deg]
α_{S2}	-43.46	-34.16	-34.11	[deg]
α_{S3}	37.66	39.91	34.87	[deg]
α_{S4}	54.12	40.56	35.49	[deg]
R_1	0.521	0.332	0.401	[au]
R_2	0.282	0.286	0.272	[au]

7.12. Together with the fitness values, the constraint values and their respective best variables, also the final orbital elements violations with respect to the ideal values are shown. The launch time is represented as a calendar date, without specifying the hours, minutes and seconds of the day: it is thus useful to also indicate the time in seconds J2000. For the three candidate solutions 1, 2 and 3, these times are: 845433504 s, 724907750.4 s, and 857245248 s

At this point, before proceeding with a Monte Carlo refinement for improving the three candidate solutions, it is interesting to compare the results when the geocentric and heliocentric phases are optimized concurrently and separately. For doing this, we can confront the Pareto fronts shown in Figure 7.11 with those of Figure 7.14. The only aspect to be modified is the time of flight: in the heliocentric only phase we did not add the tof of the best found geocentric phase solution (which corresponded to 228.45 JD): we can now add this time to the fronts and inspect the differences between each single algorithm when the two problems are treated either separately or as a single one. In Figure 7.15, this comparison is shown: in particular, each sub-figure represents the Pareto front of an algorithm in the case in which the problem is treated as two separate optimization problems, or when these two



(a) ACOmi without meta-algorithm.

(b) ACOmi with meta-algorithm.

(c) PSO with meta-algorithm.

(d) DE1220 with meta-algorithm.

Figure 7.15: Pareto fronts comparison between geocentric and heliocentric phase optimizations done separately and concurrently.

problems are optimized together.

As can be seen, all the fronts coming from the two phases optimized separately are always dominant with respect to those coming from their concurrent optimization. The only exception happens for DE1220, where for high ΔV (i.e., ΔV>1300 m/s) the front of the two phases together starts to dominate the other one and reaches very low values of time of flight.

This means that for this number of function evaluations, there are no advantages in treating the problem as one single optimization problem, but it is actually better to split it into two different ones. Hence, this also means that the problem, at least for these numbers of function evaluations, seems to be separable (i.e., the problem can be separated in two different times of flights, which can be optimized independently without worsening the performances). It has also to be noted that by increasing the number of function evaluations, we expect the Pareto fronts of the geocentric and heliocentric phase optimized as one single problem to converge towards those of the two phases optimized separately: indeed, when optimized separately, the geocentric phase has been subjected to nearly 10,000 function evaluations, plus a local MC refinement of 15,000 evaluations. The solution found with these techniques, has then been subjected to other 20,000 function evaluations when optimizing the heliocentric phase. Hence, the domain has been explored more effectively for the two problems treated separately, since the two phases concurrently have been optimized with 20,000 function evaluations. Mainly, the reason for this is that the heliocentric phase is very light in terms of computation time, and we could thus run 20,000 function evaluations without increasing the wall-clock time too much so that the overall wall-clock time for the two separate and joint phases was maintained similar. Perhaps, by strongly increasing the function evaluations, we could also expect an enhancement of the concurrent optimization w.r.t. the separate one. However, this has to be paid in terms of computation time.

As a final remark, we can observe how the solution found with the geocentric and heliocentric phase optimized in one single problem results to be more versatile. In fact, while the three candidate solutions of the two problems treated separately have the same tof for the geocentric phase, equal to 228.45 JD. However, this is not the case anymore for the second strategy, and the three candidate solutions result to have the following tof for the geocentric

phase: 265.10 JD, 232.81 JD, 273.81 JD. In terms of overall tof, the third solution (i.e., the one with a geocentric tof of 273.81 JD) seems to have the lowest overall time of flight. This means that the candidate solution with the highest geocentric flight time is also the one with the lowest overall flight time. Nevertheless, the fact that the final orbit results to be quite far from the ideal one (with a ΔV of 3.68 km/s) makes this solution less appealing.

All things considered, we can summarize the main aspects of the separate and concurrent optimization as follows:

1. For having the same performances in the optimization problem where the two phases are handled concurrently, it is required to use more function evaluations that thus increase the wall-clock time.

2. The joint problem seems to find more versatile solutions, due to the fact that the time of flight of the geocentric phase is free to vary and is not constrained to a single value.

3. By splitting the two phases we can handle the computation time in a better way, due to the fact that the geocentric and heliocentric phases have very different wall-clock times.

4. It is possible that, when optimizing the joint problems, if the function evaluations are increased until very high values (i.e., 40,000, 100,000 or even more), we can then find the best Pareto fronts for all the algorithms (contrary to what happens for 20,000 function evaluations). However, this was not checked, due to the very high wall-clock times (for 20,000 function evaluations, each run was already taking around 4.5 days). Therefore, the two phases optimized separately seems to give better results for the wall-clock times that we analyzed.

7.2.3.1 Local Refinement: Geocentric and Heliocentric Phases

Similarly to what has been done in the heliocentric and geocentric phase alone, also, in this case, an MC local refinement is run within a 1% bound of all the variables, for the three candidate solutions. In this case, due to the quite high simulation times (in the order of 20 seconds per function evaluations), 15,000 function evaluations are performed. Also, each MC run is executed three times, to remove the randomness.

In Figure 7.16, we show all the feasible individuals for all the three candidate solutions, where each candidate is run three times (for a total of nine runs) with different seeds, for ensuring to remove the randomness. Also, the Pareto-optimal solutions shown in red pertain to all the runs of all the candidates: thus constituting the overall Pareto-optimal front of the MC local refinement. Besides, in Figures 7.17b and 7.17a, the histograms that represent the distributions of the ΔV and tof values of the candidate solutions as a function of the runs are shown: the only exception is Candidate 3, which is not displayed in these figures. The reason is that the feasible points were less than 100 out of 15,000 runs: thus making impossible to do statistics with this small amount of data.

As seen in these figures, there are only two main areas that are appealing in terms of Pareto-optimal solutions: one that pertains to Candidate 2 and one to Candidate 3. In particular, the Pareto-optimal individuals that pertain to Candidate 2 seems to be more interesting, as the ΔV is maintained very low (reaching a minimum value of 26.16 m/s) while keeping the time of flights values that range between 1957 JD and 2062 JD. However, the lowest tof's are found for the Pareto-optimal solutions at very high ΔV (i.e., around Candidate 3). In particular, the lowest value of tof found is 1887 JD, but it corresponds to a ΔV of 3636 m/s. Concerning the feasibility, the MC runs have shown that Candidate 1 has 98% of feasible individuals, whereas Candidate 2 has 54% feasible individuals and Candidate 3 only 0.36%. This means that candidate 3 is a very 'unstable' solution, in the sense explained in Section 7.2.2.1. For all these reasons and similarly to what has been done in Section 7.2.2.1, we have decided to choose as best individual the one with the lowest ΔV (that comes from the MC run around Candidate 2). The parameters of this individual are shown in Table 7.13. In this table, the launch date corresponds to an ephemeris time of 728913000 s.

Therefore, we now have all the elements to discuss and compare the best individuals of the separate and concurrent optimizations: this will be done in the following section.

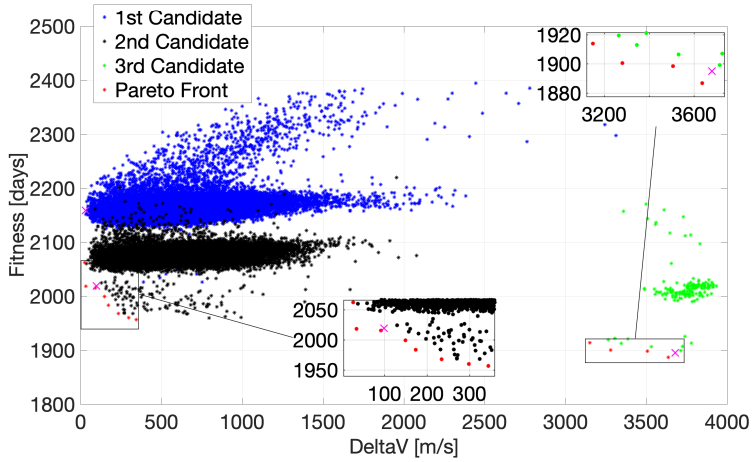
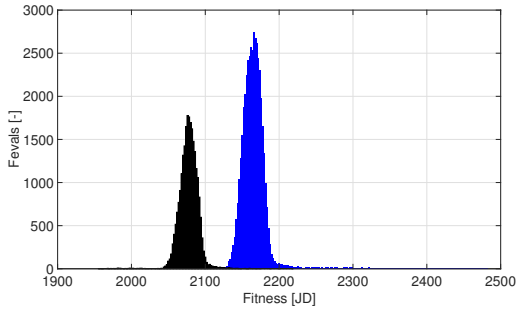
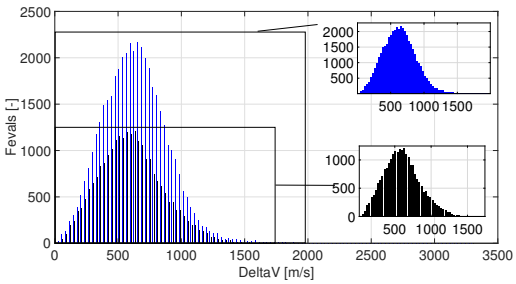


Figure 7.16: MC run around the three candidate solutions of the geocentric and heliocentric phase. In the figure, also the Pareto-optimal fronts around each candidate solutions have been highlighted.



(a) tof vs number of runs.



(b) ΔV vs number of runs.

Figure 7.17: ΔV and tof distribution as a function of the number of runs for two candidate solutions of the geocentric and heliocentric phases.

7.2.4. Optimizations Comparison

As we have already discussed in Sections 7.2.1, 7.2.2 and 7.2.3, the optimizations were performed using two different strategies: in one case, the geocentric phase was optimized separately and then, with the best found solution, the heliocentric phase was optimized. On the other hand, we have also performed a single optimization of both the phases concurrently. Now, we can thus compare the results coming from the two strategies. In particular, in Figure 7.18 we compare the Pareto-optimal shapes of the Monte Carlo runs of both the strategies.

As we can observe, the separate phases are capable of finding more Pareto-optimal points (i.e., 51 versus 12): this was predictable since these results come from two optimizations and several MC simulations (with 20,000 evaluations each). Therefore, we might not find surprising that the separate phases manage to explore the search space in a more accurate fashion. However, the Pareto-optimal front of the separate phases is not always better than the one of the two phases optimized concurrently. Especially for high ΔV, this latter seems to find better Pareto-optimal solutions. This is due to the fact that the tof of the geocentric phase is flexible in concurrent optimization. Indeed, as we have already explained in Section 7.2.3.1 the lowest overall tof is found for high geocentric time of flights: these solutions are however not explored in the separate optimizations, as only the minimum geocentric tof is then optimized in the heliocentric phase.

To summarize, we have a conflicting scenario where we are able to find better solutions for low ΔV with a separate optimization, but the concurrent optimization seems to find more optimal individuals for higher ΔV and lower tof. In our case, we have decided to select the best individual with a low ΔV value, as this means that we are nearer to the ideal operational

Table 7.13: The best overall solution, coming from the optimization of the geocentric and heliocentric phases concurrently.

Symbol	**Best Candidate**	*Units*
ΔV	26.16	[m/s]
tof	2062.40	[JD]
Δa	8.89×10^{-5}	[au]
Δi	3.46×10^{-3}	[deg]
Δe	9.90×10^{-4}	[-]
T_{launch}	2023-02-05	yyyy/mm/dd
ω_0	5.7896	[deg]
α_{E1}	31.78	[deg]
α_{E2}	4.42	[deg]
α_{E3}	6.29	[deg]
α_{S1}	-37.85	[deg]
α_{S2}	-33.92	[deg]
α_{S3}	40.05	[deg]
α_{S4}	40.28	[deg]
R_1	0.3324	[au]
R_2	0.2851	[au]

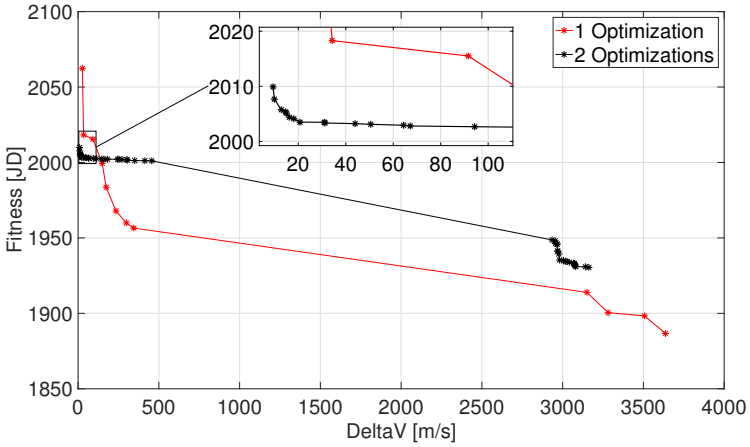


Figure 7.18: Pareto-optimal shapes of the MC runs around the best candidates.

orbit when we reach the Sun. Therefore, the individual shown in Table 7.11 is selected as the best overall solution for the SO solar-sail problem: this individual comes from the geocentric and heliocentric phases optimized separately. Nevertheless, we expect that this individual can also be found in the concurrent optimization if more function evaluations are performed (e.g., increasing the population size or the generation size, or both).

7.3. Multi-Objective

Until now, the mission has been optimized by minimizing the time of flight, while trying to fulfill some constraints. Nevertheless, the sail configuration was kept fixed, so that both the area and the mass of the sail could not vary. We now ask ourselves: what happens if we allow the area and mass of the sail to vary within certain bounds? Can we find other interesting solutions with a smaller area and mass?

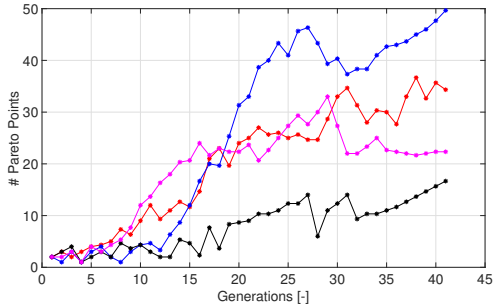
Of course, the area of the sail is a crucial parameter for such a mission since the low-thrust force generated by the sail is proportional to its area, and we thus expect that by varying this parameter the results will strongly change. Until now, the sail area and mass were kept fixed at values of, respectively, 9,800 m^2 and 204 kg. Now, we allow the area of

the sail to vary between 5,000 and 24,000 m^2, which means that the mass varies between 104 and 499.2 kg (remembering that the sail area and mass are related by Equation (2.3)). By adding this new objective in the mission, we thus obtain a multi-objective constrained problem: this makes a whole new set of algorithms available, but also excludes those that we used for single-objective problems. In particular, we will apply four different algorithms, whose working principles and characteristics are discussed in Chapter 5 and in Appendix C: NSGA-II, MOEA/D, MHACO and NSPSO. The first two algorithms are well known and very popular in the scientific community, with a long-standing heritage of tests and benchmarks in various problems. On the other hand, the last two are not widely used and thus required more tweaking and tuning (as it is shown in Sections 7.1.2 and 7.1.3). In particular, MHACO was introduced in this thesis study for the very first time, and thus results to be a brand new MO algorithm.

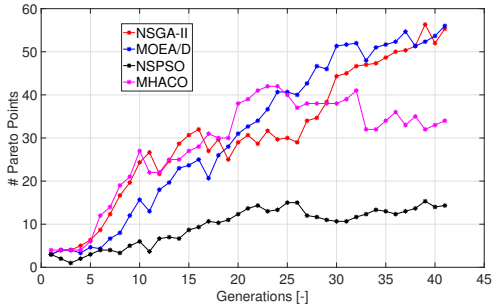
As already mentioned in Section 5.4, for making these algorithms usable for our problem, it is necessary to turn the final orbit and final time constraints into objectives. In fact, these algorithms are not capable of handling constraints, and a common strategy to circumvent this issue is to turn inequality and equality constraints into objectives. Indeed, our purpose is to keep the ΔV constraint as near to zero as possible and to keep the final time constraint strictly to zero, whenever possible: these algorithms, by minimizing these two values, will try to achieve this. Similarly to what has been done in Sections 7.2.2 and 7.2.3, the solutions that violate the final time constraint and other operational constraints (i.e., they crash into celestial bodies or they reach too low altitudes above them) are considered unfeasible and thus not shown in the Pareto fronts. Therefore, in this post-processing phase, the problem results as if it is a 3-objectives unconstrained problem with the peculiarity that certain types of solutions (i.e., those that violate the final time and operational constraints) are excluded.

When comparing the four aforementioned MO algorithms, we had to establish their sets of input parameters. First of all, three different population sizes were tested: 56, 120 and 220. The generation size was kept fixed at 40 and it was not tweaked during the optimization. The reason is that previous studies suggested that this number of generations seems to be a good compromise between finding optimal solutions and maintaining a reasonable computation time (Garot, 2006),(Spaans, 2009). Instead, it was preferred to tweak the population sizes and the algorithms' input parameters. As far as concerns both NSPSO and MHACO, their input parameters were derived from both the V&V phase, thoroughly discussed in Appendix B, and the tuning phase, presented in Sections 7.1.2 and 7.1.3. As a consequence, for the population sizes of 56 and 220, the input parameters chosen for MHACO corresponded to those of Option 9, in Table 7.1, while for a population size of 120, to Option 7. On the other hand, for NSPSO, the same input parameters mentioned in Section 7.1.3 were chosen, with the niche count diversity strategy for 120 individuals and the crowding distance diversity strategy for 56 and 220 individuals. Concerning MOEA/D and NSGA-II, their input parameters are those presented in Section 6.2.2 and they were determined during the V&V phase.

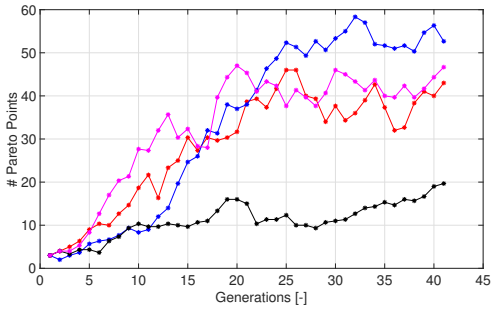
By running these four algorithms on the three aforementioned population sizes over three runs per population set (for removing the randomness), we have found the Pareto front progressions shown in Figure 7.19 (where the number of Pareto optimal points found at each generation are shown). Also, the joint hypervolume values depicted in Figure 7.20 and numerically indicated in Table 7.14 were found. As can be seen, the NSGA-II algorithm seems to outperform the others in terms of the absolute maximum value of the hypervolume. Furthermore, when looking at each population set we observe that MHACO outperforms the other algorithms for 56 individuals but it results to be worse than NSGA-II for 120 and 220 individuals, although it still outperforms MOEA/D and NSPSO for all the population sizes. This trend is not compliant with what we observe in the Pareto front progression plots. Indeed, there we observe that MOEA/D seems to outperform all the other algorithms for all the population sizes, whereas NSGA-II always performs as second best and MHACO as third best. Therefore, as already pointed out in Section 7.1.2, the number of Pareto points as a function of the generations does not seem to be a very trustworthy performance metric, especially if it is not associated with other metrics. In fact, MOEA/D might result in a bigger number of Pareto-optimal individuals due to the fact that it has converged to a local optimal front, whereas the other algorithms are still progressing towards the global best. For this reason,



(a) 56 individuals and 40 generations.



(b) 120 individuals and 40 generations.



(c) 220 individuals and 40 generations.

Figure 7.19: Average number (over three runs) of Pareto points as a function of the generations for NSGA-II, MOEA/D, NSPSO and MHACO algorithms .

Table 7.14: Joint hypervolume values w.r.t. the reference point, for NSGA-II, MOEA/D, NSPSO and MHACO.

Algorithms	pop= 56	pop=120	pop=220
NSGA-II	492228.09×10⁶	502815.24×10⁶	518969.89×10⁶
MOEA/D	483955.84×10⁶	485162.83×10⁶	497872.69×10⁶
NSPSO	455460.43×10⁶	465393.45×10⁶	462066.31×10⁶
MHACO	494756.29×10⁶	501915.54×10⁶	506573.98×10⁶

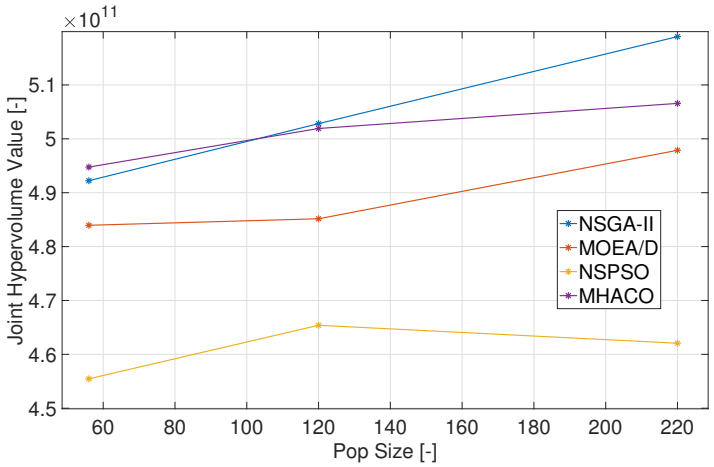


Figure 7.20: Graphical representation of the joint hypervolume values reported in Table 7.2. These values are shown for all the four tested algorithms and as a function of the population sizes.

we should always couple this information with other performance metrics that also account for how good the Pareto fronts of the different algorithms are (such as the hypervolume metric used in this research).

Concerning the hypervolume values, it can be seen that the best overall hypervolume value is found for NSGA-II with 220 individuals. By checking the greatest contributors (among all the population sizes) for all the best runs (in terms of hypervolume values) of the four algorithms, we find the following values:

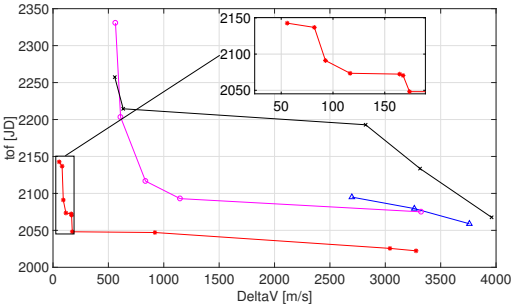
- For NSGA-II, the best run results to be for 220 individuals and has the following greatest contributor: (105.79 kg, 1952.71 JD, 557.11 m/s, 0 s).
- For MOEA/D, the best run happens for 220 individuals and displays the following greatest contributor: (243.07 kg, 1991.35, 1102.15 m/s, 0 s).
- For NSPSO, the best run results to be for 120 individuals and has the following greatest contributor: (104 kg, 2202.63 JD, 114.43 m/s, 0 s).
- For MHACO, the best run results to happen for 220 individuals and has the greatest contributor equal to: (129.35 kg, 1975.56 JD, 1334.96 m/s, 0 s).

However, as already pointed out in Section 7.1, the greatest contributors to the hypervolume are not necessarily the best individuals in the front, but they are the ones that are located in less crowded regions of the fronts. Therefore, we cannot base our choice of the best solutions on the greatest contributors to the hypervolume only. Moreover, having three objectives (the fourth one is the final time constraint, which results to be equal zero for all the found points), we would need to plot a three-dimensional Pareto front, which is not very insightful for the search of the best candidates. For this reason, we have decided to restrict our search to only the individuals with the lowest mass: indeed, we would like to see whether it is possible to fly the mission with the minimum mass, while maintaining a reasonably low time of flight and ΔV. Furthermore, the mass results to be a very critical objective since it heavily influences the cost of the mission (i.e., a higher mass to be launched means a higher launch cost).

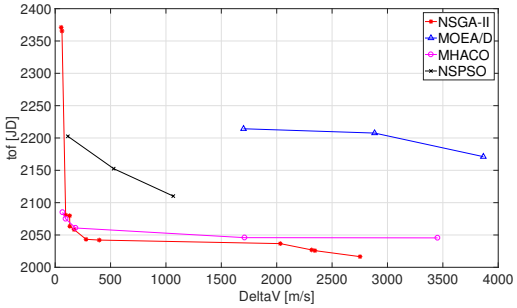
Therefore, we show in Figure 7.21, three different Pareto fronts that only display the individuals with a mass smaller than 105 kg and with the final time constraint fulfilled (i.e., $\Delta T_{final} = 0$ s). These solutions are representative of the Pareto optimal solutions over three different runs with three different seeds for all the algorithms. Also, three population sizes are treated: 56, 120 and 220. Each run within the population set has a different randomly generated initial population, however, the seed is controlled for making sure that the initial populations are the same for the various algorithms so that they all start from the same baseline.

Surprisingly, we discover that the Pareto front of NSGA-II results to dominate that of the other algorithms for a population size of 56, although MHACO displayed a higher joint hypervolume value. This can be explained by remembering that we are only plotting the solutions with a mass smaller than 105 kg, thus hiding several individuals that have a higher mass and are considered in the hypervolume computation. Therefore, MHACO displays better individuals in terms of the time of flight and ΔV for higher mass values. Also, for a population size of 56, MHACO is capable of finding a Pareto optimal front with more diversified individuals (i.e., less crowded). Indeed, by looking at the final front size (of the three runs combined), we observe a size of 28 for MHACO and 55 for NSGA-II: out of 55 individuals, NSGA-II displays only eight individuals with a mass higher than 106 kg (i.e., 85% of the found individuals are below 106 kg), while for MHACO, 15 individuals are above 106 kg (i.e., less than 50% of the found individuals are below 106 kg).

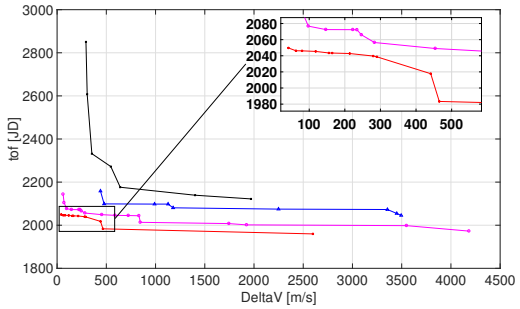
Now, looking at the three found fronts, together with the information coming from the hypervolume computations, we can determine the best candidate solutions. Similarly to what has been done for the SO case, we would like to select three candidate best solutions, which will then be refined with a local Monte Carlo simulation, as shown in Section 7.3.1. We will thus select three types of individuals: one that extremely minimizes the time of flight, one that extremely minimizes the ΔV, and one that represents a trade-off between these two objectives. As we can observe from the Pareto optimal front plots, the two individuals with the lowest tof and ΔV have the following objective functions' values: (104.013 kg, 1959.75 JD, 2598.12 m/s), (104.31 kg, 2049.78 JD, 41.66 m/s, 0 s). While for the individual that minimizes both the time of flight and the ΔV concurrently, the following objective functions



(a) 56 individuals and 40 generations.



(b) 120 individuals and 40 generations.



(c) 220 individuals and 40 generations.

Figure 7.21: Pareto optimal fronts for three different algorithms (i.e., NSGA-II, MOEA/D, NSPSO and MHACO) run over three population sizes, with three runs per population set. Here, only the individuals that fulfill the final time constraint (transformed into objective) and have a mass smaller than 105 kg are benchmarked with respect to the time of flight (tof) versus the final orbit objective (i.e., ΔV).

are found: (104.30 kg, 1983.34 JD, 465.30 m/s, 0 s). All these individuals are found with the NSGA-II optimizer. Furthermore, the variables and objective functions' values corresponding to these three candidate solutions are shown in Table 7.15. The launch dates of the three candidates can be expressed in Julian seconds as 782406432 s for Candidate 1, 753091344 s for Candidate 2 and 752030956.8 s for Candidate 3.

We now have all the elements to refine these three solutions with a Monte Carlo simulation and finally discuss the best overall candidate to be chosen for flying the mission. This is discussed in the following section.

7.3.1. Local Refinement

As it was done for the single-objective case, an MC local refinement run is executed for understanding whether we can improve the three candidate solutions presented in Table 7.15. In particular, each variable is varied within a 1% bound and three runs with three different seeds are executed for each candidate solution so that the randomness is canceled. Moreover, 15,000 function evaluations are performed around each candidate solution. First of all, in terms of solutions feasibility (i.e., those that do not violate the final time constraint), we observe a very high percentage of feasible solutions: 83% for Candidate 1 and 88% for Candidate 2 and 3 (each averaged over three runs).

Since all three candidate solutions vary in a 1% range of the areas shown in Table 7.15, all the masses of the individuals in the MC simulations will be smaller than 106 kg. We can thus plot all the individuals of all the nine runs of the MC simulations in terms of flight times and ΔV. This is done in Figure 7.22, where only the feasible individuals are displayed. Moreover, in Figures 7.23b and 7.23a, the histograms that represent the distributions of the ΔV and tof values of the candidate solutions as a function of the runs are shown. It is important to remember that in all these figures, only the individuals with a mass that is between 104 and 106 kg are shown.

From these figures, we can observe that the MC runs around each candidate produce

Table 7.15: Three candidate best solutions of the MO optimization, and their corresponding fitness values.

Symbol	**Candidate 1**	**Candidate 2**	**Candidate 3**	*Units*
ΔV	41.66	465.30	2598.12	[m/s]
tof	2049.78	1983.34	1959.75	[JD]
m	104.31	104.30	104.013	[kg]
T_{launch}	2024-10-17	2023-11-12	2023-10-31	yyyy/mm/dd
ω_0	348.73	358.27	357.674	[deg]
α_{E1}	22.16	23.19	35.07	[deg]
α_{E2}	10.26	17.62	1.28	[deg]
α_{E3}	0.37	9.70	10.88	[deg]
α_{S1}	-35.22	-36.87	-38.43	[deg]
α_{S2}	-37.00	-30.25	-22.84	[deg]
α_{S3}	40.27	40.71	38.57	[deg]
α_{S4}	38.45	36.11	33.38	[deg]
R_1	0.354	0.307	0.395	[au]
R_2	0.285	0.287	0.264	[au]
A	5015.07	5014.19	5000.64	[m^2]

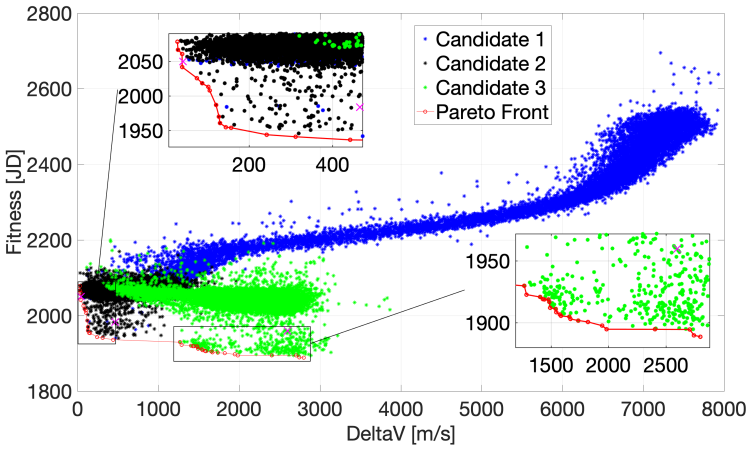
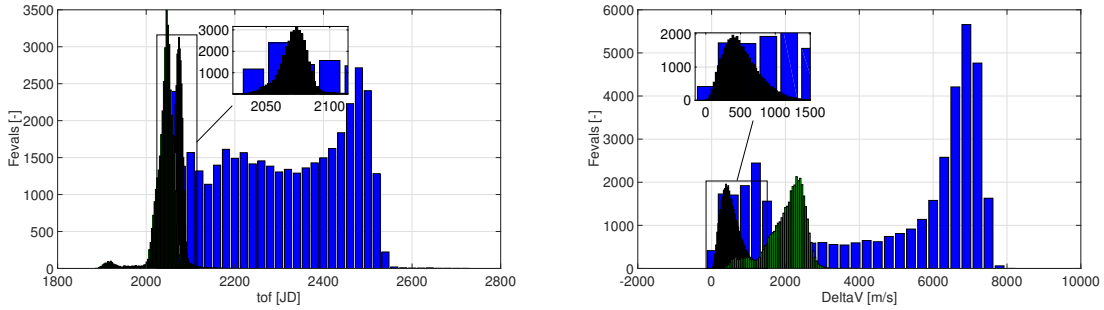


Figure 7.22: All the Monte Carlo runs around the three candidate solutions found in the MO simulation.



(a) tof vs number of runs.

(b) ΔV vs number of runs.

Figure 7.23: ΔV and tof distribution as a function of the number of runs for the three candidate solutions of multi-objective case.

very different results. In particular, the runs around Candidate 1 result to be more spread in terms of ΔV and tof: we hence observe a very wide range of possible solutions when the variables are slightly varied. This also means that the candidate solutions of this type result to be more unstable, as the mission profile varies strongly when the variables are only varied

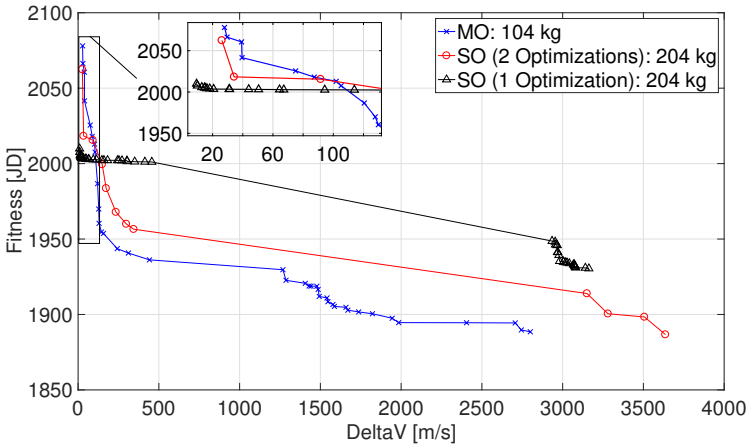


Figure 7.24: Pareto-optimal fronts comparison of two SO optimizations and one MO optimization. All the individuals of the SO cases have a mass of 204 kg, whereas those of the MO case have a mass between 104-106 kg.

by 1%. Therefore, this highlights the difficulty for the optimizer of finding the solutions with a very low ΔV, as they seem to be located in an area of the search space that is crowded of points with a much higher ΔV. Concerning Candidate 2, we observe a very different situation: all the individuals coming from the MC simulations seem to be quite cluttered in a certain area of the graph, thus allowing more stability of the solutions. Finally, Candidate 3 results to be a compromise between an unstable and stable situation, being less spread than Candidate 2 but more cluttered than Candidate 1.

We now have all the elements to compare the best overall candidates of the MC runs with those of the SO case. However, it is first important to recall that for the MO case, we have managed to find solutions with a mass between 104-106 kg, this makes a huge advantage w.r.t. the SO case, as they were all representative of a mass of 204 kg.

In Figure 7.24, we compare the Pareto-optimal fronts for three different situations: two single-objective cases (one where the geocentric and heliocentric phases are optimized together and one where they are optimized separately) and one multi-objective case.

Some interesting conclusions can be drawn from this plot: first of all, the SO case where the two phases are optimized separately manages to achieve the solutions with the lowest ΔV, reaching a minimum value of around 9 m/s, as already shown in Table 7.11. Nevertheless, the MO simulation displays a Pareto front that is always dominant, except for low ΔV. Besides, this front shows more spread solutions in the front: thus granting more freedom to the mission analyst (i.e., a more diverse set of solutions to choose from). Finally, it has to be considered that the MO case has solutions with half of the mass w.r.t. the SO cases.

All things considered, we have thus decided to choose the best overall individual from the MO optimization. In particular, we have selected the individual with the smallest ΔV of 27.89 m/s, with a corresponding flight time of 2078 JD, which is nearly 70 days higher than the one of the best SO solution. The variables and constraint violations corresponding to this best overall solution are displayed in Table 7.16. The launch time expressed in Julian seconds corresponds to 752567328 s. In Section 7.5, we will investigate this solution and its physical characteristics.

7.4. Random Seed Influence

As we have pointed out when describing the optimization of the different missions, the seed of the optimization algorithms was always controlled. In practice, this means that the comparisons between different algorithms (or between the same algorithms with different input parameters) were always made starting from the same initial populations of individuals: so that the performances could be compared starting from the same baseline. If that had not been done, the results would have been biased due to the differences in the initial population. The main reason for this is that we are using metaheuristic global optimization algorithms

Table 7.16: The best overall solution among all the mission profiles studied (i.e., two SO and one MO optimization problems).

Symbol	**Best Candidate**	*Units*
ΔV	27.89	[m/s]
tof	2078	[JD]
Δa	1.58×10^{-4}	[au]
Δi	1.11×10^{-2}	[deg]
Δe	7.97×10^{-4}	[-]
T_{launch}	2023-11-06	yyyy/mm/dd
ω_0	357.58	[deg]
α_{E1}	23.12	[deg]
α_{E2}	17.75	[deg]
α_{E3}	9.76	[deg]
α_{S1}	-37.09	[deg]
α_{S2}	-30.32	[deg]
α_{S3}	40.86	[deg]
α_{S4}	35.80	[deg]
R_1	0.3041	[au]
R_2	0.2893	[au]
A	5062.2	[m^2]
m	105.29	[kg]

for designing the mission. These algorithms are very useful when we need to sample a search space that is too large to be completely explored (e.g. by grid sampling methods or Monte Carlo methods). Controlling the seed, thus allows us to both make the results reproducible and compare the algorithms on the same initial population.

As we could observe in previous chapters, for all the simulated mission scenarios, we have always executed each run three times, with three different seeds. This happened to make sure that the trade-off is as objective as possible and we are not biased due to some random effect. However, are three runs enough for avoiding the results to be biased? How did we come up with this number?

For answering this question, we repeated the SO geocentric mission and the MO geocentric and heliocentric mission over ten runs with ten different controlled seeds. If the found results over ten different seeds consistently differ from those of three seeds, then this would probably mean that ten runs, or maybe more, are needed to remove the randomness. This operation should thus be repeated for higher runs with controlled seeds, until the results start to converge. Since analyzing the seed influence over the entire set of optimization algorithms used in this research and for all the population sizes goes beyond the scope of this study, we limited ourselves to only study the seed influence on two different mission profiles (i.e., the SO geocentric mission and the MO mission), using the SO and MO ant colony optimizers (i.e., ACOmi and MHACO) with a population size of 50 and 56 (for the SO and MO case, respectively) and evolving the algorithms for 40 generations.

In Table 7.17, we show the mean and standard deviations of the fitness values (in Julian days) for the SO case with three and ten runs. As far as concerns the MO case, in Table 7.18, we display the hypervolume mean and standard deviations of three runs, versus that of ten runs. This means that the hypervolume was computed for the Pareto optimal solutions of each run, and their values were then averaged either over three or over ten runs. As we can see, in both cases the results seem to suggest that three runs are already satisfactory for having trustworthy results when comparing the optimizers. Concerning the SO, the difference between ten and three runs are about 0.36% in the mean and 10.63% in the standard deviations. Whereas for the MO case, only 0.085% in the mean and 0.033% in the standard deviation.

For both cases, the differences both in the standard deviations and in the mean values are very low when compared to the differences we found among the different optimization

Table 7.17: Random seed influence for the SO case: 3 vs 10 controlled seeds. The mean (μ) and standard deviation (σ) of the best fitness values found for these two cases are shown.

Runs	μ	σ	*Units*
3	235.77	4.82	[JD]
10	236.61	4.31	[JD]

Table 7.18: Random seed influence for the MO case: 3 vs 10 controlled seeds. The mean (μ) and standard deviation (σ) of the hypervolume values of the Pareto optimal solutions for these two cases are shown.

Runs	μ	σ	*Units*
3	479206750047.6506	11710423955.584814	[-]
10	479618359646.7511	11714274259.124847	[-]

algorithms, or even between the same algorithm with different input parameters, as it can be seen in Sections 7.1, 7.2 and 7.3. Therefore, we have considered three runs enough to remove the randomness, and in this entire research we have optimized the solar sailing polar mission to the Sun by running the SO and MO algorithms three times, using the same three controlled seeds for each batch of runs, so that the algorithms are compared starting from the same initial populations.

Moreover, running each optimization three times instead of ten or more has permitted us to spend nearly three times less computation time, thus allowing us to investigate other aspects (e.g., doing more algorithms tuning or studying the problem with different formulations).

7.5. Optimal Trajectory

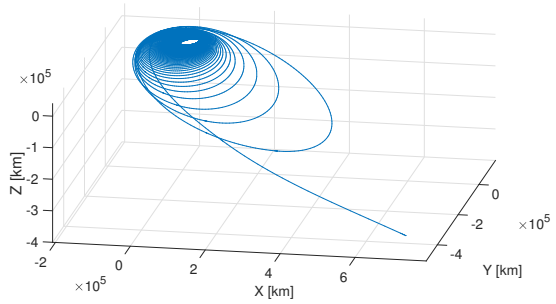
Now that both the SO and MO optimizations have been executed, we can select the best overall solution and show some characteristics of its mission profile.

In particular, due to its smaller mass (i.e., nearly half w.r.t. the SO case) and a tof and a ΔV of only 70 days and 15 m/s higher, we have decided to select the best solution coming from the MO phase.

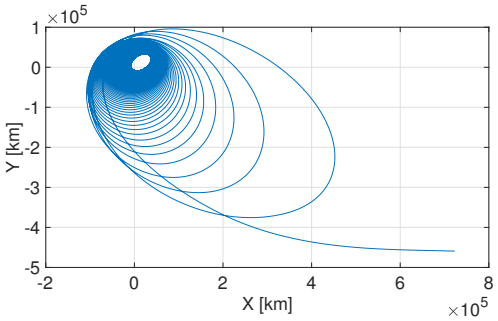
Regarding this best solution, the heliocentric flight time is about 1810 JD, whereas the geocentric time lasts nearly 268 JD. The orbits corresponding to these two phases and their projections are shown in Figures 7.25 and 7.26.

We will now analyze the two phases separately, by studying the characteristics of each phase: in particular, we will mainly focus on the various accelerations exerted on the sailcraft during both phases. All these accelerations will be discussed in terms of acceleration norms, while their real effect manifests itself throughout a three components vector.

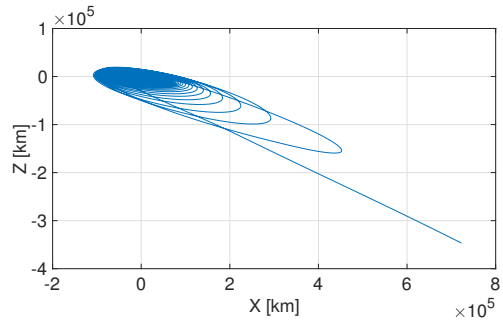
Concerning the geocentric phase, by analyzing the perturbing accelerations to which the sail is subjected in this phase, we found out some interesting aspects. First of all, for this best solution, the atmospheric drag barely affects the sail in terms of acceleration norm: indeed, not only, its maximum peaks at only $1.0{\times}10^{-4}$ m/s^2 but its action on the sail lasts only a few minutes (for the rest of the time it always maintains a value lower than 10^{-7} m/s^2). This aspect is only true for the best individuals, whereas the worst ones always display a stronger and longer influence of the atmosphere. This confirms that the atmosphere is an influential perturbing acceleration for this mission, as it was already demonstrated in Candy (2002). Secondly, the solar-sail acceleration, the Moon acceleration and the Sun's acceleration acting on the sailcraft are analyzed. As we could expect, the Sun's acceleration grows as the sailcraft escapes the Earth (i.e., the time increases), with some wobbling, due to the elliptic shapes of the escape trajectories. Towards the end of the geocentric phase, the heliocentric acceleration outweighs the Earth's one, thus causing the central body to shift from the Earth to the Sun. Furthermore, the solar radiation pressure also shows a predictable behavior: the maximum values of this acceleration stay almost constant for the entire geocentric phase (as the sailcraft basically maintains a fixed distance w.r.t. the Sun in the geocentric phase). Nevertheless, there is a slight decrease in this peak acceleration throughout the flight time: this hints that



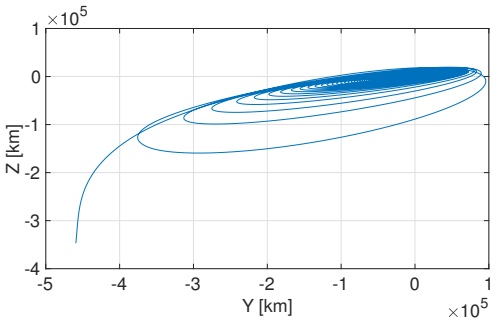
(a) Geocentric orbit.



(b) Projection of the geocentric orbit onto the (X,Y) plane.

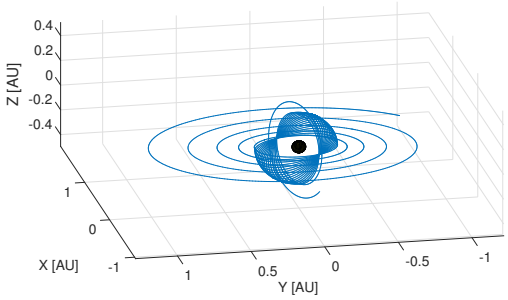


(c) Projection of the geocentric orbit onto the (X,Z) plane.

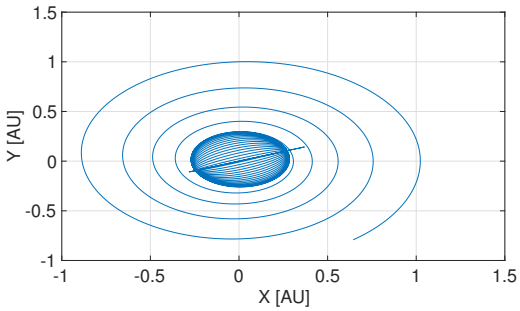


(d) Projection of the geocentric orbit onto the (Y,Z) plane.

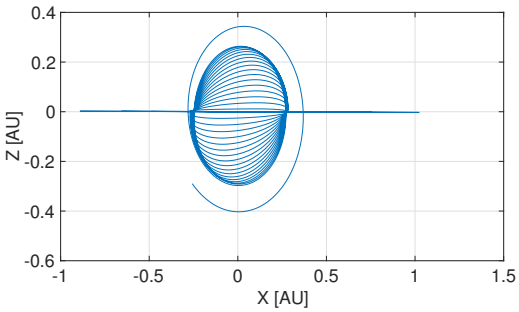
Figure 7.25: Geocentric orbit in the equatorial reference system of the best solution found in Section 7.3.



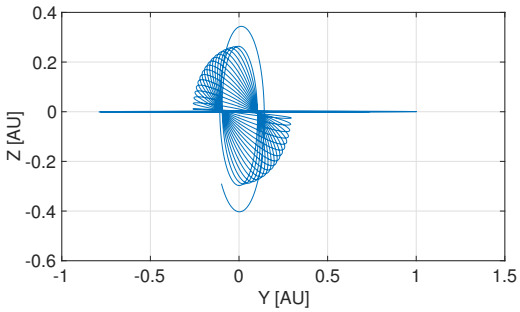
(a) Heliocentric orbit.



(b) Projection of the heliocentric orbit onto the (X,Y) plane.

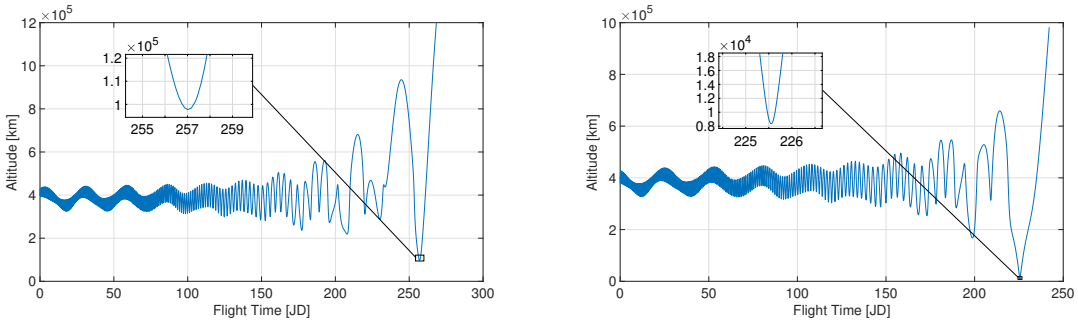


(c) Projection of the heliocentric orbit onto the (X,Z) plane.



(d) Projection of the heliocentric orbit onto the (Y,Z) plane.

Figure 7.26: Heliocentric orbit in the ecliptic reference system of the best found in Section 7.3.



(a) Best overall solution presented in Table 7.16. (b) Candidate 2 solution of Table 7.15.

Figure 7.27: Altitude of the sail w.r.t. the Moon in the geocentric phase, as a function of the flight time in Julian days.

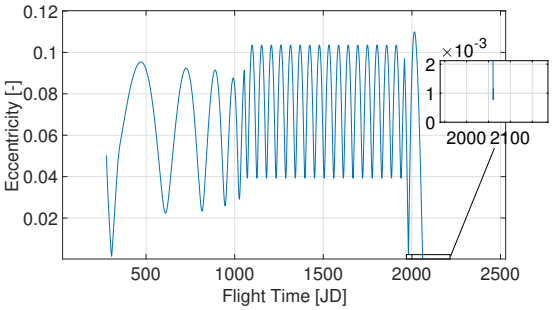
the sailcraft is escaping in the opposite direction w.r.t. the Sun, meaning that the optimal trajectory has a heliocentric phase that starts at a further distance w.r.t. the Sun. We can explain this by observing that the sailcraft would have not been able to be propelled by the solar pressure radiation if it had escaped in the direction of the Sun (indeed, in that case, the sailcraft would have been in the towards the Sun phase, and the solar radiation pressure could have only caused a decelerating effect). Therefore, the optimal trajectories often result in an escape distance higher than the Sun-Earth distance. Moreover, the solar pressure acceleration sometimes goes to zero in the geocentric phase: this happens due to the fact that the sailcraft is either shadowed by the Earth w.r.t. the Sun or is in the "towards the Sun" phase.

Finally, by investigating the Moon's acceleration as a function of the flight time, we notice that the acceleration peaks at a certain point in the trajectory, reaching peak values of around $5{\times}10^{-4}$ m/s^2. To better investigate this, we show the Moon-sail altitude, in Figure 7.27a: as we can observe, the sailcraft never reaches a Moon's altitudes lower than 90,000 km. This means that the optimal trajectory does not display any flyby to the Moon, as the sailcraft does not enter the sphere of influence, which is located at around 65,000 km from the surface.

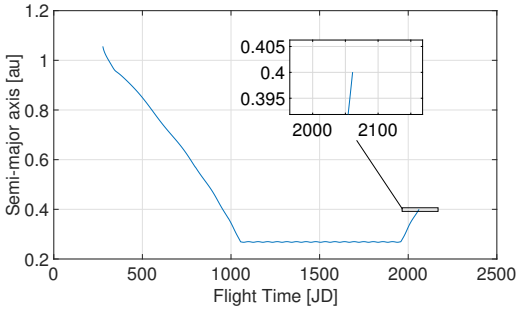
However, this aspect is not always verified: indeed, if the desired time of flight should be maintained to lower values, the sail seems to perform a flyby to the Moon. For instance, in Figure 7.27b, we show the Moon-sail distance plot of the Candidate 2 solution presented in Table 7.15: this solution displays a very low time of flight, even though its ΔV is not among the best ones. As we can observe, for this solution, the sail reaches Moon's altitudes of around 8,000 km, meaning that it enters the sphere of influence of the Moon. As a result, the sail manages to escape the Earth's gravitational attraction in only 239 Julian days. As a consequence, the Moon's gravitational acceleration becomes predominant during a certain phase of the mission (peaking at maximum values of $4.5{\times}10^{-2}$ m/s^2) and the Moon starts to act as a central body. It would thus be interesting to include the Moon's flyby in the optimization for observing whether other interesting sets of solutions could be found if the gravity assist to the Moon is included in the mission profile.

Now that the geocentric orbit has been thoroughly studied, we can analyze the heliocentric phase. This phase starts as soon as the sail reaches the sphere of influence of the Earth. The final objective of this phase is to achieve as precisely as possible a circular orbit of 0.4 au of radius and 90° of inclination around the Sun. In Figure 7.28 we show the orbital elements of the sail. The sail reaches a final eccentricity of $7.968{\times}10^{-4}$, a final inclination of 89.99° and a final semi-major axis of 0.40016 au.

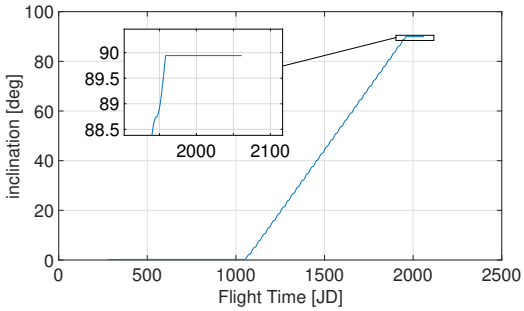
The main accelerations in this phase are only due to the central gravity of the Sun, third-body perturbations from other planets and solar pressure radiation. The two main perturbing bodies in this phase are the Earth, Mercury, and Venus: although the perturbing magnitude is very different (with the Earth's peak perturbing acceleration being one thousand times stronger than the other two planets: 5×10^{-4} vs 5×10^{-7} m/s^2), they act in a different phase of the sail journey, thus making all of them important to include. Indeed, the Earth's acceler-



(a) Eccentricity vs flight time.

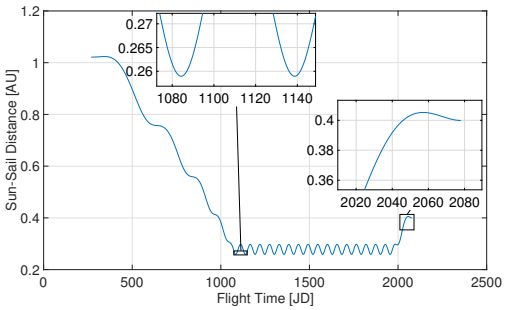


(b) Semi-major axis vs flight time.

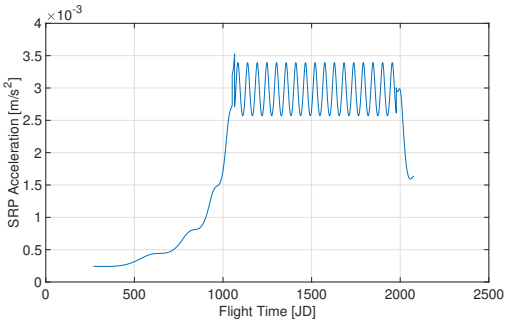


(c) Inclination vs flight time.

Figure 7.28: Semi-major axis, inclination and eccentricity of the sail w.r.t. the Sun in the heliocentric phase of the mission, as a function of the flight time in Julian days.



(a) Sun-sail istance vs flight time.



(b) Solar radiation pressure acceleration vs flight time.

Figure 7.29: Distance of the sail w.r.t. the Sun and solar radiation pressure acceleration acting on the sail, as a function of the flight time in Julian days.

ation only acts in the very first phase, when the sail is still near the Earth, whereas the other ones act in a later stage. Concerning the solar radiation pressure acceleration, its behavior is now very different w.r.t. that of the geocentric phase. Indeed, as shown in Figure 7.29b, we can now see that this acceleration increases as we move towards the Sun, and starts to wobble during the cranking phase. In the end, when spiraling outwards to the final orbit of 0.4 au, this acceleration drops again: basically, it has a behavior that is strictly dependent on the sailcraft distance to the Sun (similarly to what also happens for the Sun's gravitational acceleration). To confirm this, we show the Sun-sail distance in Figure 7.29a.

To summarize, the optimal mission seems to have several key features. First of all, the atmospheric drag seems to play a marginal role, meaning that the best trajectory is the one where this perturbing force is not very prominent. Secondly, the Moon flyby seems an aspect to be investigated further: while it does not play a role for the best mission, for an optimal mission where the time of flight is to be preferred w.r.t. the ΔV it can become a pivotal aspect

to consider. Moreover, concerning the heliocentric phase, it seems that this phase is more regular w.r.t. the geocentric one: there are less unpredictable perturbing forces, and those that are present all have a regular behavior (i.e., predictable from theoretical considerations). As a final remark, by comparing these results with previous studies such as those presented in Candy (2002), Garot (2006) and Spaans (2009), we observe a strong enhancement: while in those cases, the best found solution completed the entire mission in 2064 days, which is very close to our best value of 2078 days. However, in this case, we perform the mission with only 104 kg of mass, whereas 204 kg were required in their mission scenario. This means that we manage to fly the mission with the same final conditions but with almost half of the mass. Furthermore, in this case, we have a wider range of possible solutions to choose from: we can either select a mission with a very low time of flight and a final orbit that is slightly off w.r.t. to the ideal one, or we can maintain a final orbit that is very close to the ideal one but accepting a higher time of flight. This aspect is crucial, as it leaves the mission analyst more freedom in choosing the mission profile. Nonetheless, it is important to highlight that those missions were developed for a different launch window: indeed, they were planning to visit the Sun in the occasion of the 25^{th} solar cycle (i.e., around 2023), whereas we plan to study the Sun in its 26^{th} solar cycle (i.e., around 2033), as we would not be able to reach the Sun in time for the 25^{th} solar cycle. Therefore, although previous missions can be used for a better understanding of the quality of the solutions found in our study, we should always keep in mind that they pertain to a different launch window, and their results are thus not directly comparable with ours.

Chapter 8

Conclusions and Recommendations

This thesis specialized in two main topics: first, in developing a new ant colony optimization algorithm, and second in comparing that global optimizer with other state-of-art optimizers on a solar sailing polar mission. This research thus proposed the introduction of two new ant colony optimizers for both single and multi-objective optimizations. These algorithms have been first tested on a wide set of problems for V&V purposes. Then, as a second step, a solar sailing polar mission was optimized using both a single and multi-objective formulation of the problem. For each of these problems, an ant colony optimizer was developed: this has hence led to the birth of ACOmi (mixed integer ant colony optimizer for SO) and MHACO (multi-objective hypervolume-based ant colony optimizer). These algorithms were developed in the context of a solar sailing mission but they were formulated for any kind of problem. Therefore, they can be applied to any optimization problem, although they are originally developed for space trajectory optimization. The outcome seems to be very promising: with nearly 104 kg of mass and 2078 JD of flight time, we manage to reach the final circular operational orbit of 0.4 au and 90° inclination around the Sun. The sail and its 5 kg payload shall be deployed at the perigee of a GTO orbit on 6 November 2023. After 268 days the sail should be able to escape the Earth's gravitational pull and start orbiting the Sun. Then, after other 1810 days, it should be able to reach a final operational orbit with 0.4 au and 89.99° inclination. The arrival date is foreseen to be the 15th of July 2029.

In this chapter, we will first make a review of the research question and subquestions in Section 8.1. Then, we will discuss the simulation and optimization conclusions in a separate section (i.e., Sections 8.2). Finally, in Section 8.3, we will address the recommendations for further studies.

8.1. Research Questions Overview

As stated in Chapter 1, the main research question of this study was:

Can the time and cost of a solar-sail mission to the Sun be optimized by using a global optimization technique?

As we have discussed in this report, we were able to optimize the solar sailing mission using a global optimization technique. When comparing the data with those found on previous studies, the results seem to be very promising: we manage to fly the mission with only 104 kg of propellant and 2078 Julian days.

Several subquestions were related to the aforementioned research question, in particular, these were divided in two main groups. We hereby present an overview of these subquestions:

1. How well can an ant colony optimization algorithm perform compared to other already implemented optimization strategies in the framework of a solar sailing polar mission?

a. Is the ant colony optimizer able to find trajectories that do not violate the equality and inequality constraints?

b. How does the ant colony optimizer behave in terms of best found solutions when compared to other optimization algorithms?

2. Is it possible to improve the current solutions for a solar sailing mission by optimizing the trajectory with multiple objective functions?

a. How does the physical trajectory vary when the cost and duration of the mission are optimized separately?

b. Can we develop a new ant colony optimization algorithm for multiple-objectives that is competitive with multi-objective state-of-art algorithms?

3. What is the most suitable way for modeling the trajectory of a sailcraft in a solar sailing mission?

a. What is the influence of the perturbing accelerations in the optimization technique?

b. Can the problem be represented in a more simple and effective way, without compromising the quality of the solutions?

In the following section, each of these subquestions will be separately discussed.

8.2. Simulations and Optimizations Conclusions

In this section, we will discuss the outcome of this thesis study related to each research subquestion. This will be done by treating each aspect in a separate paragraph.

Ant Colony Optimizer Performances

As we can observe, various research questions revolve around the question whether an ant colony optimizer is capable of being competitive with other global techniques for space trajectory optimization. In this thesis, both the single and multi-objective ant colony optimizers have been tested on a wide range of problems before being applied for the solar-sail mission. In particular, for SO, the algorithm has been tested on 4 different unconstrained continuous test problem, 24 constrained continuous problems and an integer constrained problem. Moreover, it was also applied on 6 different real-life inspired trajectory optimization problems. In all these cases, the algorithm was benchmarked with other SO global optimizers, which currently represent the state-of-art methods.

Also the multi-objective extension was tested on 21 different problems (belonging to three different well known test-suites).

All these testing phases have allowed us not only to adjust and enhance the algorithms' behavior, but also to develop a thorough understanding of their working principles, so that the different input parameters could have then been tweaked in a more physical and meaningful way.

When applying the ant colony optimizers for the solar sailing mission, we were deeply impressed by the results: the ACO either performed as the best algorithm (e.g. for the geocentric phase) or among the very best ones (e.g. for the heliocentric phase and for the MO mission). This confirms that indeed ACO does behave in a competitive way w.r.t. other optimization algorithms, and that in some cases even outperforms global techniques with a long-standing heritage, such as differential evolution.

Multi-Objective Solar Sailing Mission

Multi-objective optimization has demonstrated to be a very powerful technique for the solar sailing polar mission of our interest: indeed, by including the mass as a separate objective we have managed to find orbits with a time of flight of 2078 JD and a mass of 104 kg, which nearly satisfy completely the ΔV constraints (with a value of around 27 m/s). When compared to SO, we observe a completely different situation: indeed, for SO, the best found solution had a tof of 2062 JD, but a mass of 204 kg, which is nearly twice as the mass of the MO case. This aspect was preferred, although the SO mission reached a smaller ΔV value of around 9 m/s. Furthermore, MO also required substantially fewer function evaluations (we could find very

promising results already with nearly 2296 function evaluations). Thus, the MO formulation of the problem allowed us to find competitive solutions in a shorter time, also allowing a more flexible mission profile, where different sail areas and mass could be evaluated. The very variable and complex configuration of the environment in such a mission (i.e., Moon's flyby, atmospheric drag, etc.) makes the problem cumbersome and counter-intuitive in some cases: as a result, by increasing the solar-sail area did not always result in a better mission profile.

Trajectory Problem Modelling

In this study, the trajectory was modeled in two main ways: by only treating the time of flight as objective (i.e., SO mission) or by also including the mass (i.e., MO mission). As we have already pointed out, this second strategy has turned out to be more powerful and versatile for the mission of our interest. However, another aspect that was also handled was how to deal with the equality and inequality constraints. In particular, in this research, it was decided to turn all the orbital constraints (i.e., final eccentricity, semi-major axis and inclination around the Sun) into a ΔV constraint: so that their violation actually translates into an estimation of how much propellant should be used for correcting the final orbit. This was included as one of the objectives in the MO mission, and was treated as a single inequality constraint in the SO counterpart. As we have observed, this has led a more flexible scenario in which we could also explore very interesting orbits, which have a higher ΔV. In previous studies, this mission was optimized for SO and with a single fitness formulation that included both the orbit elements violations, the cost and the mass. However, this formulation turns out to be very subjective. Also, it does not allow to explore a wide range of solutions, due to the fact that the complexity of the problem is reduced to only one single equation. Our mission formulation, on the other hand, seems to give a more objective and comprehensive picture of the situation. With such a formulation, we were able to find better solutions in the same or lower number of function evaluations.

Another aspect to be noted is that by optimizing the geocentric and heliocentric phases separately, for SO, it was found that better results could be achieved in terms of overall wall-clock time, compared to the case in which these two phases are handled concurrently.

Concerning the perturbations, we verified that it is essential to include the Moon and Sun's third body perturbations in the geocentric phase and that is also fundamental to take into account the solar pressure radiation variations during the geocentric phase. Indeed, we observed that the maximum solar pressure acceleration varies in a noticeable way during this phase, thus shaping the trajectory in a different way. Also, it was verified that both Mercury and Venus act as third-body perturbations of the same strength during the heliocentric phases. Although this also depends on the trajectory flown, it was verified that, in general, both these celestial bodies can perturb the sailcraft with similar acceleration magnitudes. Hence, if one of the two body is included as perturbation, the other one must be included as well.

Finally, we have also studied the influence of five different propagation schemes (i.e., Cowell, modified equinoctial elements and unified state model with quaternions, exponential mapping and Rodrigues parameters) for the SO geocentric and heliocentric mission. We have benchmarked them comparing the different Pareto fronts found and the total wall-clock time required for finding these fronts. It turned out that the modified equinoctial elements propagator represents the best compromise between low wall-clock time and good quality of the Pareto front. Therefore, this propagator has been used over the entire research.

8.3. Recommendations

The recommendation for further studies are divided in three parts: those that concern the optimization part only, those that pertain to the simulation model and those that regard both of them. We will tackle each of these in the following paragraphs.

Optimization Recommendations

Concerning the optimization part, we recommend future researchers to take into account the following aspects:

1. Tweaking and tuning all the input parameters of the algorithm is, in general, time consuming and often subjective. Although we could find a set of input parameters that gave the best result in a group of more, we could not conclude that those sets of parameters are the best overall. Hence, it is recommended to develop a technique for avoiding this burden to the future users of global optimization techniques. This could be done in three ways: one possibility is to develop an artificial intelligence technique that is capable of tweaking the algorithm in the most efficient way, given a set of training data (i.e., neural networks or other machine learning techniques, for instance). The second possibility, is to develop a self-adaptive method for adjusting these parameters automatically, inside the algorithm. As we have seen in this thesis study, this technique has already been implemented in several differential evolution variants, demonstrating to be very effective and handy. The third possibilit is to use hyper-parameters optimization, such as Bayesian optimization, to establish the optimal input set for the optimization algorithm.

2. Although this was not among the primary objectives, in this study, we developed a non-dominated sorting particle swarm optimizer (NSPSO) algorithm. This, however, seemed to be clearly outperformed by other MO optimizers. Nonetheless, we were not able to establish, whether this was due to the lack of a more thorough tweaking and tuning of the many input parameters of the algorithm, or if this only mirrored the weakness of the algorithm itself. A more thorough study seems necessary to solve this. In any case, with the same time allotted for the benchmarking and testing phases, the other MO (including MHACO) algorithms have seemed to be more powerful.

3. Can a local optimization method improve the results? In this thesis study, we only focused on global optimization techniques, although we make use of some optimal steering laws for formulating the optimization problem. It would, however, be interesting to see what are the performances of a local optimizer on this problem, and if it can be coupled with a global one, to improve the overall performances of the mission. For doing this, several local optimizers available in PaGMO could be used.

4. From our study, we have concluded that if a single-objective problem is handled, then it is better to split the geocentric and heliocentric phases, and optimize them separately. It would be interesting to see what happens for very high function evaluations: is the problem still separable in that case?

Trajectory Simulation Recommendations

While analyzing the solar sailing polar problem, we dealt with some interesting and peculiar aspects that have raised several recommendations for future studies. These can be summarized as follows:

1. The flyby to the Moon seems to be a pivotal aspect of the mission when the tof should be minimized: it thus seems to be crucial to include a flyby to the Moon in the mission profile for observing whether a wider set of possible interesting solutions can be found. For future studies, it is recommended to further investigate this, and possibly include some of the flyby parameters (e.g. flyby angle, etc.) to be optimized.

2. The type of final orbit is fundamental for the whole optimization phase. A different final orbit is hence expected to result in different optimization outcomes. It would be interesting to extend the possible final orbits to more than one (e.g. by including circular orbits in a different resonance with the Earth), so that a more flexible mission profile can be designed.

3. When trading-off five different propagators for the SO geocentric and heliocentric mission, it turned out that the modified equinoctial elements represent a good compromise between low wall-clock time and good quality of the Pareto front. This result, however, is problem dependent, and other propagators (such as the modified equinoctial elements with Rodrigues parameters) have also shown interesting results. It would thus be interesting to compare these propagators for all the mission profiles (i.e., for the separate geocentric and heliocentric phases and for the multi-objective case) and establish whether there is a propagator that best suits all these mission scenarios, in terms of quality of the Pareto front and wall-clock time.

Appendix A

Equations of Motions

In this appendix, the equations of motion (i.e., Equation (3.38)) that describe the translational dynamics of the spacecraft will be derived. In particular, this will be done for three different types of state variables introduced in the previous sections: Cartesian elements, modified equinoctial elements, and unified state model. Moreover, three possible representations of the equations of motion using different modifications of the USM will be described (namely, using quaternions, modified Rodrigues parameters and exponential mapping).

A.1. Cowell Propagator

Given a certain Cartesian position vector $\mathbf{r} = (x, y, z)^T$ and velocity vector $\mathbf{v} = (v_x, v_y, v_z)^T$, the state variable vector is defined as:

$$\mathbf{x} = \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \end{pmatrix} \tag{A.1}$$

In the Cowell propagator, this vector is used as state variable, and the time has the role of independent variable. This results in the following equations of motion (set of first-order differential equations):

$$\frac{dx}{dt} = v_x \tag{A.2}$$

$$\frac{dy}{dt} = v_y \tag{A.3}$$

$$\frac{dz}{dt} = v_z \tag{A.4}$$

$$\frac{d^2x}{dt^2} = -\frac{\mu}{\left(\sqrt{x^2 + y^2 + z^2}\right)^3} x + a_{pert,x} \tag{A.5}$$

$$\frac{d^2y}{dt^2} = -\frac{\mu}{\left(\sqrt{x^2 + y^2 + z^2}\right)^3} y + a_{pert,y} \tag{A.6}$$

$$\frac{d^2z}{dt^2} = -\frac{\mu}{\left(\sqrt{x^2 + y^2 + z^2}\right)^3} z + a_{pert,z} \tag{A.7}$$

where $\mathbf{a}_{pert} = (a_{pert,x}, a_{pert,y}, a_{pert,z})^T$ is the vector that represents the sum of perturbing accelerations (including the solar-sail force).

A.2. Modified Equinoctial Elements

As we already pointed out, MEE is a modified version of the Kepler elements, made to avoid mathematical singularities, which limited the use of the Kepler elements. Their first appearance was in Walker (1986), and the relation between MEE and Kepler elements $(a, e, \nu, \omega, \Omega, i)$ is:

$$p = a(1 - e^2) \tag{A.8}$$
$$f = e\cos(\omega + \Omega) \tag{A.9}$$
$$g = e\sin(\omega + \Omega) \tag{A.10}$$
$$h = \tan\frac{i}{2}\cos\Omega \tag{A.11}$$
$$k = \tan\frac{i}{2}\sin\Omega \tag{A.12}$$
$$L = \Omega + \omega + \nu \tag{A.13}$$

For the unperturbed two-body problem, all the elements except for L result to be constant. However, when there are perturbations in the model, we can derive (see Walker (1986) for the derivation) the following differential equations that describe the variations of the MEE due to these forces:

$$\frac{dp}{dt} = \sqrt{\frac{p}{\mu}}\frac{2p}{w}a_N \tag{A.14}$$
$$\frac{df}{dt} = \sqrt{\frac{p}{\mu}}\left(\sin(L)a_S + \left((w+1)\cos L + f\right)\frac{a_N}{w} - \left(h\sin L - k\cos L\right)\frac{f}{w}a_W\right) \tag{A.15}$$
$$\frac{dg}{dt} = \sqrt{\frac{p}{\mu}}\left(-\cos(L)a_S + \left((w+1)\sin L + g\right)\frac{a_N}{w} + (h\sin L - k\cos L)\frac{f}{w}a_W\right) \tag{A.16}$$
$$\frac{dh}{dt} = \sqrt{\frac{p}{\mu}}\frac{s^2}{2w}\sin(L)a_W \tag{A.17}$$
$$\frac{dk}{dt} = \sqrt{\frac{p}{\mu}}\frac{s^2}{2w}\sin(L)a_W \tag{A.18}$$
$$\frac{dL}{dt} = \sqrt{p\mu}\frac{w^2}{p^2} + \frac{1}{w}\sqrt{\frac{p}{\mu}}\left(h\sin L - k\cos L\right)a_W \tag{A.19}$$

where:

$$s^2 = 1 + h^2 + k^2 \tag{A.20}$$
$$w = 1 + f\cos L + g\sin L \tag{A.21}$$
$$r = \frac{p}{w} \tag{A.22}$$

The perturbing acceleration components (a_S, a_N, a_W) are in the same direction of the forces expressed in Figure A.1: hence, a_S is the perturbing acceleration in the direction of the radial outwards, a_W is the component of the acceleration in the direction of the angular momentum vector ($\mathbf{h} = \mathbf{r}\times\mathbf{v}$), and a_N is the acceleration perpendicular to the aforesaid two, in the direction that completes the right-hand orthogonal reference system. It is quite easy to pass from the Cartesian elements to the modified equinoctial elements. Indeed, having transformed the Cartesian elements to Kepler elements, by simply applying the equations listed from Equation (A.8) to Equation (A.13), we can pass from Kepler elements to modified equinoctial elements. However, the opposite (from modified equinoctial to Cartesian) is not as trivial as this case.

After some manipulations (see Walker (1986) for further details), we can describe the transformation between MEE to Cartesian variables as follows:

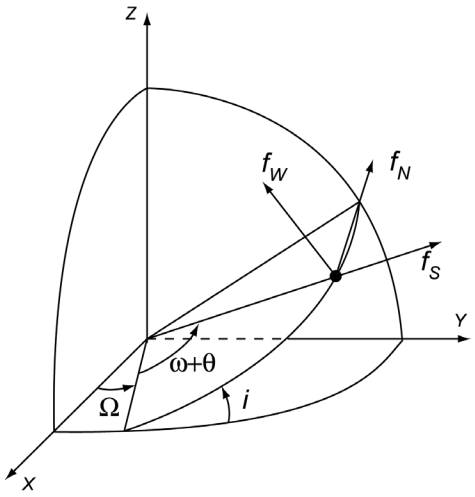


Figure A.1: Geometry of the f_S, f_N and f_W components of the force with respect to the orbit and to the inertial reference frame (X,Y,Z)(Wakker, 2015).

$$x = \frac{r}{s^2}\left(\cos L + \alpha^2 \cos L + 2hk \sin L\right) \tag{A.23}$$

$$y = \frac{r}{s^2}\left(\sin L - \alpha^2 \sin L + 2hk \cos L\right) \tag{A.24}$$

$$z = 2\frac{r}{s^2}\left(h \sin L - k \cos L\right) \tag{A.25}$$

$$v_x = -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g\right) \tag{A.26}$$

$$v_y = -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(-\cos L + \alpha^2 \cos L + 2hk \sin L - f + 2ghk + \alpha^2 f\right) \tag{A.27}$$

$$v_z = \frac{2}{s^2}\sqrt{\frac{\mu}{p}}\left(h \cos L + k \sin L + fh + gk\right) \tag{A.28}$$

where s, w and r are the same as the ones in Equations (A.20), (A.21) and (A.22), respectively; and α can be computed as:

$$\alpha^2 = h^2 - k^2 \tag{A.29}$$

A.3. Unified State Model

In this section, we will discuss the equations of motions derived from three different sets of state variables, all belonging to the unified state model class: those that make use of quaternions, those that take advantage of the modified Rodrigues parameters and those that use the exponential mapping. In doing this, we will make use of the definitions introduced in Section 3.2.

A.3.1. Quaternions

Before introducing the USM equations of motion using quaternions, it is first fundamental to describe certain parameters that will be used.

We recall that the velocity of the spacecraft (**v**) is written as the sum of two components: one along the direction of the velocity at periapsis (**R**) and one along the perpendicular direction to the radius vector, at any instant time (**C**):

$$\mathbf{v} = \mathbf{C} + \mathbf{R} \tag{A.30}$$

It is clear that for rotating from $\mathcal{F}_g$ to $\mathcal{F}_f$, a Euler rotation of an angle i around the line of nodes has to be made. On the other hand, for rotating from $\mathcal{F}_f$ to $\mathcal{F}_e$, another Euler rotation has to be made, around the direction of $\hat{\mathbf{f}}_3$, of an angle λ. This angle is called true longitude, and it is defined as the sum of the right ascension of the ascending node, the argument of perigee and the true anomaly:

$$\lambda = \Omega + \omega + \nu \tag{A.31}$$

We can now describe the orientation of $\mathcal{F}_e$ w.r.t. $\mathcal{F}_g$ in quaternions as follows:

$$\begin{pmatrix} \epsilon_{O1} \\ \epsilon_{O2} \\ \epsilon_{O3} \\ \eta_O \end{pmatrix} = \begin{pmatrix} \sin\frac{i}{2}\cos\left(\frac{\Omega-\omega-\nu}{2}\right) \\ \sin\frac{i}{2}\sin\left(\frac{\Omega-\omega-\nu}{2}\right) \\ \cos\frac{i}{2}\sin\left(\frac{\Omega+\omega+\nu}{2}\right) \\ \cos\frac{i}{2}\cos\left(\frac{\Omega+\omega+\nu}{2}\right) \end{pmatrix} \tag{A.32}$$

In the case of equatorial orbits and circular orbits, Ω and ω are not defined, respectively. However, the USM can still be used and the four parameters of the quaternion that expresses the orientation of the orbit, shown in Equation (A.32), will all be set zero.

Unfortunately, the use of this propagator implies a more complex definition of the equations of motion. To define them, it is first necessary to pass through intermediate steps, which will now be listed in Equations (A.33), (A.34), (A.35) and (A.36). First of all, it is possible to relate the quaternions to the angle λ as:

$$\begin{pmatrix} \sin\lambda \\ \cos\lambda \end{pmatrix} = \frac{1}{\epsilon_{O3}^2 + \eta_O^2} \begin{pmatrix} 2\epsilon_{O3}\eta_O \\ \eta_O^2 - \epsilon_{O3}^2 \end{pmatrix}$$

hence:

$$\lambda = \tan^{-1}\left(\frac{2\epsilon_{O3}\eta_O}{\eta_O^2 - \epsilon_{O3}^2}\right) \tag{A.33}$$

Furthermore, using the hodograph the velocities expressed in the $\mathcal{F}_e$ frame can be found as:

$$\begin{pmatrix} v_{e1} \\ v_{e2} \end{pmatrix} = \begin{pmatrix} 0 \\ C \end{pmatrix} + \begin{bmatrix} \cos\lambda & \sin\lambda \\ -\sin\lambda & \cos\lambda \end{bmatrix} \begin{pmatrix} R_{f1} \\ R_{f2} \end{pmatrix} \tag{A.34}$$

The relation between C and v_{e2} is expressed through the parameter p:

$$p = \frac{C}{v_{e2}} \tag{A.35}$$

Finally, we define a parameter γ as:

$$\gamma = \frac{\epsilon_{O1}\epsilon_{O3} - \epsilon_{O2}\eta_O}{\epsilon_{O3}^2 + \eta_O^2} \tag{A.36}$$

Now, to express the equations of motion, we have to compute the perturbing accelerations $\mathbf{a}_e = (a_{e1}, a_{e2}, a_{e3})^T$ (i.e., all the accelerations acting on the body except for the central gravity field) in the $\mathcal{F}_e$ frame. As derived in Altman (1972) the dynamic equation for the quaternion have the following angular velocities:

$$\omega_1 = \frac{a_{e3}}{v_{e2}} \tag{A.37}$$

$$\omega_2 = 0 \tag{A.38}$$

$$\omega_3 = \frac{C v_{e2}^2}{\mu} \tag{A.39}$$

As one would expect, the angular velocity component along the $\hat{\mathbf{e}}_2$ axis (ω_2) is zero: indeed, there is no velocity component out of the local orbital plane. The parameter C can be easily found remembering that the angular momentum vector can be computed as:

$$\mathbf{h} = \begin{pmatrix} h_x \\ h_y \\ h_z \end{pmatrix} = \mathbf{r} \times \mathbf{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \times \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} yv_z - zv_y \\ zv_x - xv_z \\ xv_y - yv_x \end{pmatrix} \tag{A.40}$$

Now, using the quaternion time derivative (see for instance Wie (2008)) we can compute:

$$\begin{pmatrix} \dot{\epsilon}_{O1} \\ \dot{\epsilon}_{O2} \\ \dot{\epsilon}_{O3} \\ \dot{\eta}_O \end{pmatrix} = \frac{1}{2} \begin{bmatrix} 0 & \omega_3 & 0 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & 0 \\ 0 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & 0 & -\omega_3 & 0 \end{bmatrix} \begin{pmatrix} \epsilon_{O1} \\ \epsilon_{O2} \\ \epsilon_{O3} \\ \eta_O \end{pmatrix} \tag{A.41}$$

Moreover, the time derivative of the hodographic velocity components can be written as (see Chodas (1981) for the entire derivation):

$$\begin{pmatrix} \dot{C} \\ \dot{R}_{f1} \\ \dot{R}_{f2} \end{pmatrix} = \begin{bmatrix} 0 & -p & 0 \\ \cos\lambda & -(1+p)\sin\lambda & -\gamma R_{f2}/v_{e2} \\ \sin\lambda & (1+p)\cos\lambda & \gamma R_{f1}/v_{e2} \end{bmatrix} \begin{pmatrix} a_{e1} \\ a_{e2} \\ a_{e3} \end{pmatrix} \tag{A.42}$$

The two systems of equations in Equation (A.42) and (A.41) constitute the equations of motion for the USM, using the quaternions. However, we still need to express R_{f1} and R_{f2} in the $\mathcal{F}_f$ frame to solve these equations. This can be done by first expressing $\mathbf{R}$ in the $\mathcal{F}_e$ frame, for a true anomaly of 90°:

$$\mathbf{R}_e|_{\nu=90^\circ} = \begin{pmatrix} R \\ 0 \\ 0 \end{pmatrix} \tag{A.43}$$

For $\nu = 90^\circ$, it holds:

$$\lambda|_{\nu=90^\circ} = \Omega + \omega + 90^\circ \tag{A.44}$$

The rotation matrix from $\mathcal{F}_e$ to $\mathcal{F}_f$ can be written as:

$$\mathbf{C}_{f\leftarrow e}|_{\nu=90^\circ} = \begin{bmatrix} \cos((\Omega+\omega)+90^\circ) & -\sin((\Omega+\omega)+90^\circ) & 0 \\ \sin((\Omega+\omega)+90^\circ) & \cos((\Omega+\omega)+90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.45}$$

Hence, we can write $\mathbf{R}$ in the $\mathcal{F}_f$ frame by simply multiplying this matrix for $\mathbf{R}$ expressed in the $\mathcal{F}_e$ frame:

$$\begin{pmatrix} R_{f1} \\ R_{f2} \\ 0 \end{pmatrix} = \begin{bmatrix} \cos((\Omega+\omega)+90^\circ) & -\sin((\Omega+\omega)+90^\circ) & 0 \\ \sin((\Omega+\omega)+90^\circ) & \cos((\Omega+\omega)+90^\circ) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{pmatrix} R \\ 0 \\ 0 \end{pmatrix} \tag{A.46}$$

One final observation is that we usually know the Cartesian perturbing acceleration vector in the inertial frame (i.e., $\mathbf{a}_{pert}$). We thus need to transform this vector from the inertial frame ($\mathcal{F}_g$) to the orbital frame ($\mathcal{F}_e$). This can be done by using the quaternion matrix $\mathbf{C}_{e\leftarrow g}$ defined as:

$$\mathbf{C}_{e\leftarrow g} = \begin{bmatrix} 1-2(\epsilon_{O2}^2+\epsilon_{O3}^2) & 2(\epsilon_{O1}\epsilon_{O2}+\epsilon_{O3}\eta_O) & 2(\epsilon_{O1}\epsilon_{O3}-\epsilon_{O2}\eta_O) \\ 2(\epsilon_{O1}\epsilon_{O2}-\epsilon_{O3}\eta_O) & 1-2(\epsilon_{O3}^2+\epsilon_{O1}^2) & 2(\epsilon_{O2}\epsilon_{O3}+\epsilon_{O1}\eta_O) \\ 2(\epsilon_{O3}\epsilon_{O1}+\epsilon_{O2}\eta_O) & 2(\epsilon_{O3}\epsilon_{O2}-\epsilon_{O1}\eta_O) & 1-2(\epsilon_{O1}^2+\epsilon_{O2}^2) \end{bmatrix} \tag{A.47}$$

Indeed, it holds:

$$\mathbf{a}_e = \mathbf{C}_{e\leftarrow g}\mathbf{f}_{pert,g} \tag{A.48}$$

A.3.2. Modified Rodrigues Parameters

We have already pointed out that by using a different set of parameters for describing the orientation of the orbit, modifications to the USM can be introduced. One of these is made through the use of the modified Rodrigues parameters. Although presenting a singularity in their representation, these parameters benefit from the use of shadow parameters for avoiding this singularity.

To derive the equations of motion for this case, it is necessary to convert quaternion elements into MRP. This can be done (for both the normal and shadow parameters) as follows:

$$\boldsymbol{\sigma} = \frac{\boldsymbol{\epsilon}}{1+\eta} \qquad \forall \eta \neq -1 \tag{A.49}$$

$$\boldsymbol{\sigma^S} = \frac{-\boldsymbol{\epsilon}}{1-\eta} \forall \eta \neq 1 \tag{A.50}$$

Now, by inserting the expression found in Equation (A.32) inside Equations (A.49) and (A.50), we find:

$$\boldsymbol{\sigma}_O = \left(1 + \cos\left(\frac{i}{2}\right)\cos\left(\frac{\Omega+u}{2}\right)\right)^{-1} \begin{pmatrix} \sin\left(\frac{i}{2}\right)\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \end{pmatrix} \tag{A.51}$$

$$\boldsymbol{\sigma}_O = \left(\cos\left(\frac{i}{2}\right)\cos\left(\frac{\Omega+u}{2}-1\right)\right)^{-1} \begin{pmatrix} \sin\left(\frac{i}{2}\right)\cos\left(\frac{\Omega-u}{2}\right) \\ \sin\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \\ \cos\left(\frac{i}{2}\right)\sin\left(\frac{\Omega-u}{2}\right) \end{pmatrix} \tag{A.52}$$

The rest of the equations concerning the shape of the orbit, will be the same: hence, Equation (A.42) will still hold for this case. However, the parameters constituting this equation will be computed differently. In particular, the relations for the cosine and sine of λ will be computed as follows:

$$\begin{pmatrix} \sin\lambda \\ \cos\lambda \end{pmatrix} = \frac{1}{4\sigma_3^2 + (1-\sigma^2)^2} \begin{pmatrix} 4\sigma_3(1-\sigma^2) \\ (1-\sigma^2)^2 - 4\sigma_3^2 \end{pmatrix} \tag{A.53}$$

The differential equations in terms of MRP and SMRP can be derived using the fact that $\omega_2 = 0$ as:

$$\dot{\boldsymbol{\sigma}} = \frac{1}{4}\begin{pmatrix} (1-\sigma^2+2\sigma_1^2)\omega_1 + 2(\sigma_1\sigma_3+\sigma_2)\omega_3 \\ 2(\sigma_2\sigma_1+\sigma_3)\omega_1 + 2(\sigma_2\sigma_3-\sigma_1)\omega_3 \\ 2(\sigma_3\sigma_1-\sigma_2)\omega_1 + (1-\sigma^2+2\sigma_3^2)\omega_3 \end{pmatrix} \tag{A.54}$$

Since both the differential equations and the relations that express the cosine and sine of λ are the same for MRP and SMRP, the aforementioned equations hold for both the sets of parameters.

A.3.3. Exponential Mapping

Also in this case, the differential equations associated with $\dot{C}$, $\dot{R}_{f,1}$ and $\dot{R}_{f,2}$ are the same as the quaternions. Also, the expression of $\boldsymbol{\omega}$ is the same. The time derivative of $\dot{\mathbf{a}}$ (needed for expressing the complete set of equations of motion) can be expressed as shown in Grassia (1998), as follows:

$$\dot{\mathbf{a}} = \frac{1}{2}\left(\Phi\cot\frac{\Phi}{2}\boldsymbol{\omega} - \boldsymbol{\omega}\times\mathbf{a} - \frac{\boldsymbol{\omega}\cdot\mathbf{a}}{\Phi}\left(\cot\frac{\Phi}{2} - \frac{2}{\Phi}\right)\right) \tag{A.55}$$

It is clear that a singularity is present at $\Phi = 0$. In Grassia (1998) a method to avoid this singularity is also proposed, using the Taylor expansion:

$$\dot{\mathbf{a}} \approx \frac{1}{2}\left(\frac{12 - \Phi^2}{6}\boldsymbol{\omega} - \boldsymbol{\omega} \times \mathbf{a} - \boldsymbol{\omega} \cdot \mathbf{a}\left(\frac{60 + \Phi^2}{360}\right)\mathbf{a}\right) \tag{A.56}$$

Appendix B

Verification and Validation

In this appendix, we will discuss the verification and validation of both the simulation model and the optimization procedure. In particular, the appendix will be organized as follows: in Section B.1 we will discuss the unit tests for the simulation model. While in Section B.2, we will investigate the validation of the entire mission with respect to the reference mission introduced in Section 2.3. Finally, in Section B.3, the optimization procedure will be verified and the results of the benchmarks will be presented. This last section is also useful to establish the set of input parameters to be used in this research for the implemented optimization algorithms.

B.1. Simulation Model

Concerning the simulation model, as we have already pointed out in Chapter 6, an existing software was used for representing and formalizing the governing equations of the sail in its journey to the Sun: Tudat. The software part that is already implemented in Tudat is assumed to be free of errors since Tudat includes several unit-tests for verifying and validating each software component. These pre-existing models include all the mathematical methods (i.e, propagators, integrators, etc.) and all environmental models (i.e., cannon-ball solar pressure radiation, gravitational forces and third body perturbations, aerodynamic forces, etc.). The only exception concerns the non-ideal solar-sail force model. Indeed, Tudat is only capable of simulating the cannon ball force model, which assumes the solar-sail force to always be perpendicular to the reflecting surface and parallel to the Sun-sail vector. This is an approximation that is often acceptable for satellites in geocentric orbits, but that would cause a consistent error in the trajectory if applied for a solar sailing polar mission. For this reason, the non-ideal solar-sail force model was introduced in Tudat and several unit-tests have been introduced to verify and validate it.

In particular, the V&V procedure was executed by checking the following aspects:

1. The non-ideal solar-sail force model has to be equal to the cannon ball, when the cone angle is zero and the normal to the sail and the Sun-sail vector are parallel.
2. At several different distances from the Sun, it was checked that the non-ideal solar-sail force model corresponds to the theoretical value when the cone and clock angles are set to zero. This was done at Earth, Venus and Uranus' distances. Also, it was checked for a random distance.
3. The solar-sail model was checked and compared to the Ulysses satellite at 1 AU. This was done for the case in which the cone and clock angles are equal to zero.
4. Finally, the update procedure of the solar-sail force model was checked (when the cone and clock angles are varied) and 17 different unit tests have been executed for checking whether the non-ideal solar-sail force model returned the expected values for the force when the attitude angles are controlled and known. The theoretical values for the force were retrieved from McInnes (1999). The following sets of cone and clock angles have been tested (i.e., (α, δ)): [90°, 0°], [90°, 90°], [90°, 180°], [90°, 360°], [-90°, 0°], [-90°, 90°], [-90°, 180°], [-90°, 360°], [0°, 0°], [0°, 90°], [0°, 180°], [0°, 360°], [45°, 45°], [-45°, -45°], [45°, -45°], [-45°, 45°], [80°, 80°], [-80°, 80°] and [15°, 25°]. These cone and clock angles were chosen in such a way that all the possible singularities were verified (to make sure that the force model was bug free: an example is [0°, 0°] or [90°, 90°]). Besides, some random sets of angles were verified (e.g. [15°, 25°] or [80°, 89°]).

Table B.1: The best overall solution, coming from the optimization of the geocentric and heliocentric phases concurrently.

Symbol	**Values**	*Units*
m	204.0	[kg]
A	9800	[m^2]
α_{attack}	3	[deg]
ϵ_f	0.05	[-]
ϵ_b	0.64	[-]
B_f	0.79	[-]
B_b	0.55	[-]
s	0.88	[-]
α	0.94	[-]

B.2. Integrated System Tests

In this section, we will tackle the V&V of the entire trajectory (including both the geocentric and heliocentric phases). In particular, the reference mission was used for validating our simulation model for the entire journey of the sail. This is also a higher level test to actually verify that all the mathematical and environmental models implemented work well together and produce accurate results (something that is not verifiable from the unit tests only). Also, this V&V procedure has allowed us to understand and discover some key aspects of the mission that would not be probably noticed otherwise.

For reproducing the reference mission, the same environment and mathematical model had to be set. In particular, starting from the same attitude angles, the same initial perigee argument and launch time, and the same cranking and circularization distances, we expect to find the same resulting orbit with the same time of flights (for both the geocentric and heliocentric phases). The variables' values corresponding to the reference mission can be seen in Table 2.1.

Slight modifications had to be performed to adapt our environment and mathematical model to that of the reference mission:

1. The atmosphere model was modified to be the same as in Spaans (2009). Hence, the variation of the density as a function of the altitude was expressed as:

$$\rho(h) = 35\left(243352h^{-7.2305} + 4537152h^{-5.6305\left(1.49-\frac{h}{11000}\right)}\right) \tag{B.1}$$

2. The shadowing of the Earth and other planets on the sail was only considered in the geocentric phase, whereas it was neglected in the heliocentric phase.

3. The propagator used was Cowell.

4. The integrator used was Runke-Kutta Fehlberg with an integration tolerance of 10^{-9} for the Geocentric phase and 10^{-12} for the Heliocentric phase.

5. The same values for the sail area, mass, the angle-of-attack during the atmospheric phase and all the non-ideal solar-sail force constants, were chosen (these can be found in Table B.1).

6. In the geocentric phase, only the Moon and the Sun were considered as third body perturbation, whereas in the heliocentric phase only the Earth, Venus and Mercury's gravitational pull are considered as third body perturbations.

7. The solar radiation pressure value is considered fixed when the sail is in the geocentric phase.

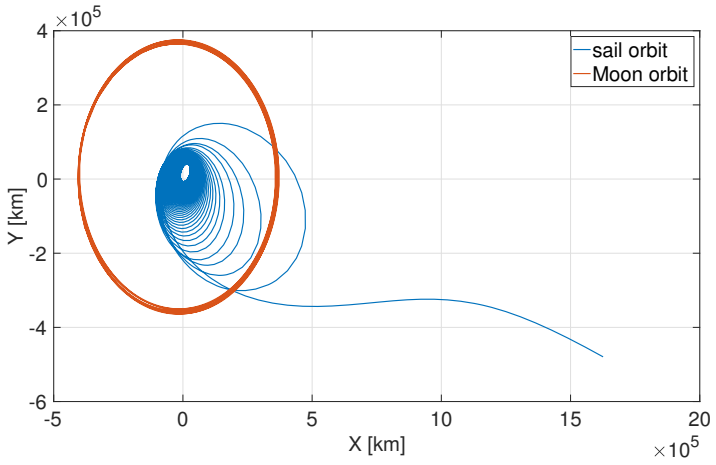


Figure B.1: Geocentric phase validation plot.

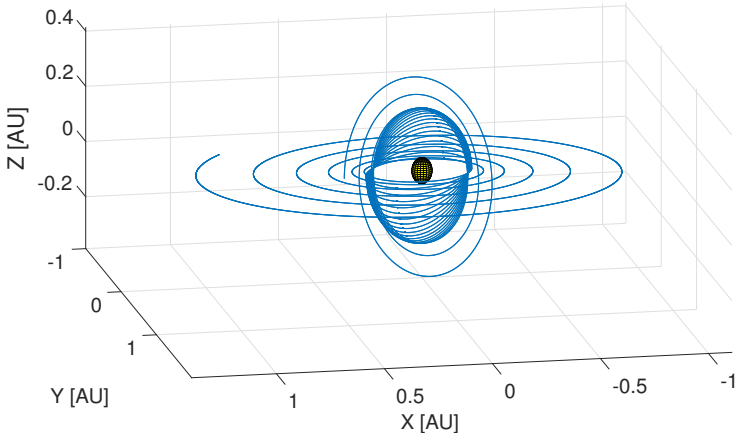


Figure B.2: Heliocentric phase validation plot.

By performing the simulation using these values, we found the results shown in Table B.2 concerning the time of flight, whereas we obtained the orbits shown in Figures B.1 and B.2, for the geocentric and heliocentric phases, respectively. By comparing these with those of the reference mission, which can be found in Spaans (2009), it is clear, also by inspection, that the two orbits are almost identical. We do not indeed expect the two orbits to be exactly identical for several reasons, which pertain both to the geocentric and heliocentric orbits. Concerning the geocentric orbits, the following differences can cause the two orbits to slightly diverge:

1. The initial time is indicated in days in the reference mission (Spaans, 2009). We assumed that this was referred to Modified Julian Days, and thus corresponded to a certain initial time in seconds. However, if a slightly different initial time in seconds was

Table B.2: Validation results for the whole mission.

Symbol	**Values**	**Spaans**	*Units*
tof_{geo}	276.42	277.5	[kg]
tof_{helio}	1792	1787	[kg]
i_{final}	90	90	[deg]
a_{final}	0.4	0.4	[au]
e_{final}	0.007	0.004	[-]

chosen for the reference mission, this could consistently change the orbit, especially due to the varying position of the Moon (whose gravitational pull may be consistently different even within one or two hours differences in the launch time of the sail).

2. Different inclinations of the equatorial plane are considered. Indeed, we decided to use the inclination as it is indicated in the SPICE ephemeris files, whereas the author of the reference mission self-coded an inclination of the equatorial plane as equal to 23.5°, which is slightly off from the real value (i.e., 23.43674°).

3. It is not clear which average radius of the Earth was considered for computing the altitude of the sailcraft (and thus for establishing the initial conditions). We considered an average radius of 6371×10^6 m, but this might be slightly different from the original mission.

Concerning the heliocentric phase, the fact that it was started with slightly different initial conditions w.r.t. those of the geocentric phase (due to the aforementioned differences) may have also caused these orbits to slightly diverge.

Overall, since it is a low-thrust orbit, it is clear that even very small differences in the environmental and mathematical models might cause quite consistent differences in the two trajectories and in the time of flights of the two phases. For these reasons, we considered our results (which are only off by 0.39% and 0.28% for the time of flight, in the geocentric and heliocentric phase, respectively) satisfactory, considering the differences in the simulation models.

B.3. Optimization Procedure

Several different optimizers have been used in this thesis study. Most of them were already coded in the PaGMO software, and they were thus already tested both in terms of performance and usage. However, since three new algorithms have been implemented in this software (and are now available for the entire scientific community), a thorough validation and verification procedure has been conducted for them. In particular, the V&V strategy consists of two different parts: first of all, it is made sure that the algorithm works for the sets of problems for which it is supposed to, and that it throws an error when wrong input parameters (or wrong population, wrong problems, etc.) are passed to the algorithm. Also, it is made sure that the algorithm is covered at 100%, meaning that it is made sure that the tests cover all the lines of the code. These tests are performed using different tools, such as Travis CI [1], AppVeyor [2], Circle CI [3] and Codecov [4]. Travis CI is a hosted continuous integration service released under MIT license and often used to build and test software, which is developed using GitHub. Similarly, also Circle CI and Appveyor are continuous integration services, which are often used for software that is being developed in GitHub. This software is considered the state-of-art test framework for those who develop software through GitHub. All details concerning the working principles of this software can be found on their official websites as discussing all aspects in detail is beyond the scope of this thesis.

When the algorithm is fully tested in its use, it has to be tested in terms of performance. Since these algorithms are all metaheuristic (i.e., they contain a certain degree of randomness for sampling design spaces that are too large to be fully explored), it is not easy to fully assess their performance. Nevertheless, several test suites have been developed throughout the years to establish how an optimization algorithm performs with respect to several different problems, which have different characteristics. In particular, the problems obviously vary between single and multi-objective optimization. For this reason, we treated these two classes of problems separately. In particular, we verified and validated the SO optimizer ACOmi against other popular algorithms, on four continuous unconstrained problems (i.e., Ackley, Griewank, Rosenbrock, Schwefel), on an integer constrained problem (Golomb Ruler) and on 24 different continuous constrained problems (CEC2006 test suite). Finally, we also

[1] `https://travis-ci.org/`, date of access: August 2019.
[2] `https://www.appveyor.com/`, date of access: August 2019.
[3] `https://circleci.com/`, date of access: August 2019.
[4] `https://codecov.io/`, date of access: August 2019.

tested ACOmi on 6 different popular space related problems (i.e., Cassini, Earth-Venus-Earth transfer, Messenger, Rosetta and two different types of Earth-Mars transfer). On the other hand, the MO algorithms have been validated on 21 different MO problems against NSGA-II and MOEA/D algorithms, belonging to three different popular test suites (i.e., ZDT, DTLZ and WFG).

All these tests have been useful to demonstrate that the SO and MO ant colony optimizers, as well as the NSPSO algorithm, have competitive results w.r.t. other well-known algorithms with a strong heritage (e.g., differential evolution, genetic algorithm, evolutionary algorithms, etc.) on a wide set of popular test problems. Also, these tests were essential to establish the set of input parameters to be chosen for the optimization algorithms: this hold for both the newly implemented algorithms and the already known ones (such as particle swarm optimization, differential evolution, etc.). We will now discuss in Sections B.3.1 and B.3.2, all the results of the tested algorithms for both the SO and MO cases.

B.3.1. Single-Objective V&V

After having tested the use of ACOmi, its performances were tested on a set of different problems. In particular, since this algorithm can handle integer and continuous variables, as well as constrained and unconstrained box-bounded problems, it was necessary to benchmark it in all these types of single-objective problems. It has to be noted that this algorithm is the first one in PaGMO capable of implementing a parallelization strategy for computing the fitness in parallel using multiple threads. The improvements deriving from this parallelization scheme are, however, strictly related to the difficulty of the problem. For a very simple test problem like Rosenbrock, using a parallelization scheme based on multi-threads (such as the one implemented for ACOmi) does not bring any computation time advantage. Of course, for applying the parallelization scheme, it is required that the problem is thread-safe [5].

In the following section, we will first treat the verification and validation of ACOmi on several continuous unconstrained problems, in Section B.3.1.1. Then, in Section B.3.1.2 we will treat continuous and integer constrained problems. Finally, in Section B.3.1.3, we will discuss the V&V of the ACOmi optimizer on several real-life inspired space problems.

B.3.1.1 Continuous Unconstrained Problems

When verifying the single-objective ACOmi algorithm, we have decided to test it against three well-known algorithms (i.e., PSO, SADE and SGA), for three popular single-objective unconstrained problems: Ackley, Griewank, Rosenbrock and Schwefel. This was done for two different population sizes (i.e., 40 and 200) and averaging the results over 10 runs. The test results for the case of 200,000 function evaluations are shown graphically in Figure B.3. Besides, in Table B.3 we have shown the numerical results of the algorithm at the last generation step. All these problems are mathematical problems whose minima are known and equal to zero. As it can be seen, the ACOmi algorithm is competitive with almost all the other algorithms for all the problems and population sizes. Furthermore, these results are only useful to show that the algorithm can actually compete with the standard SO algorithms for known test problems. We have thus decided not to spend too much time to tweak and tune ACOmi for achieving even better results.

Ackley

This is a scalable box-constrained continuous SO problem first introduced in Ackley (1987). Its generalized objective function can be formulated (for n dimensions) as follows:

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

[5] A piece of code is said to be thread-safe when the implemented functions can correctly perform simultaneous tasks using multiple threads. This means that these multiple threads must be able to access the same shared data in parallel. In short, for having thread-safety it is required that the code can be executed simultaneously by multiple-threads.

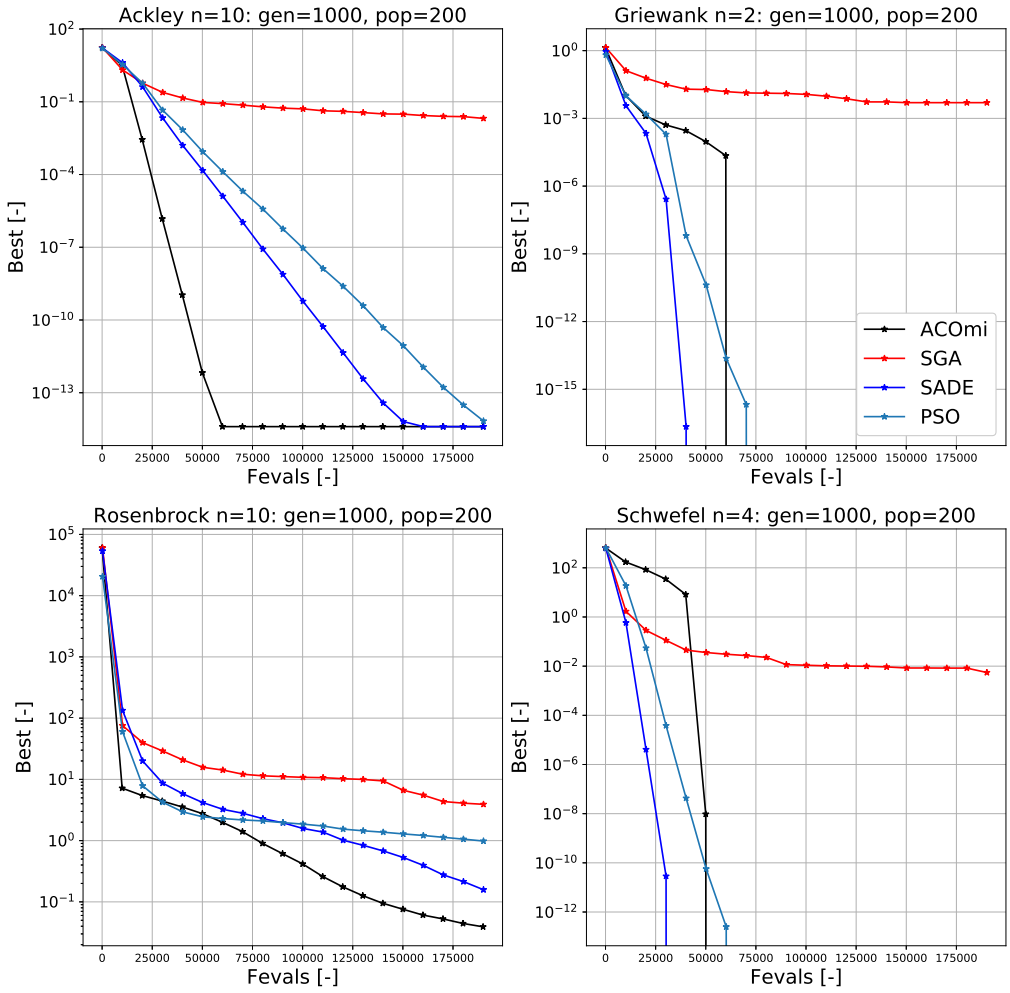


Figure B.3: Graphical representation of test results of ACOmi, SGA, SADE and PSO on several test problems, for 200,000 function evaluations.

where $x_i \in [-15, 30]$. The global minimum is located at $x_i = 0$ and equals zero (i.e., $F(0, ..., 0) = 0$). The graphical representation of this function in two dimensions is shown in Figure B.4.

The Ackley function implemented for our research is ten-dimensional (i.e., $n = 10$).

Griewank

This popular test function for unconstrained optimization was first introduced in Griewank (1981). It represents a box-constrained continuous single-objective problem, that can be generalized (for n dimensions) as follows:

$$F(x_1, ..., x_n) = \sum_{i=1}^{n} x_i^2/4000 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \tag{B.3}$$

where $x_i \in [-600, 600]$ and the global minimum is placed at $x_i = 0$, and equals zero (i.e., $F(0, ..., 0) = 0$). The shape of this function for two dimensions is shown in Figure B.5.

Table B.3: Test results of ACOmi, SGA, SADE and PSO on several test problems, for 40,000 and 200,000 function evaluations: the subscript 40 refers to the former, whereas 200 to the latter.

Problems	**ACOmi**	**SGA**	**SADE**	**PSO**
Ackley_{40}	3.99680×10^{-15}	0.10322	3.99680×10^{-15}	1.89182×10^{-14}
Griewank_{40}	0.00771	0.04782	0.0	0.0
Rosenbrock_{40}	0.75354	6.87421	0.84379	2.08644
Schwefel_{40}	0.0	0.10886	0.0	23.68766
Ackley_{200}	3.99680×10^{-15}	0.02407	3.28626×10^{-15}	6.483702×10^{-15}
Griewank_{200}	0.0	0.00751	0.0	0.0
Rosenbrock_{200}	0.03937	3.97623	0.23784	1.08399
Schwefel_{200}	0.0	0.00667	0.0	0.0

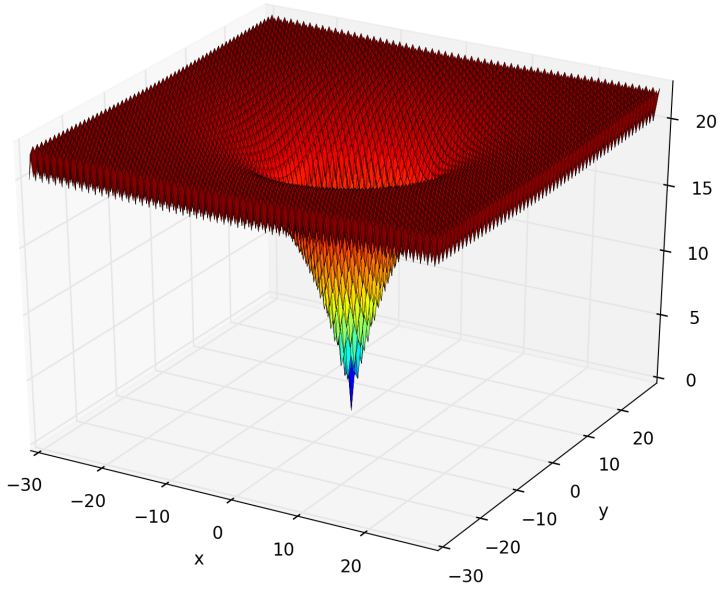


Figure B.4: Two-dimensional Ackley function.

The Griewank function implemented in this thesis study is two-dimensional (i.e., $n = 2$).

Rosenbrock

This is another popular SO unconstrained scalable test problem, first introduced in Rosenbrock (1960). Its generalized objective function is:

$$F(x_1, ..., x_n) = \sum_{i=1}^{n-1} \left[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \right] \tag{B.4}$$

where $x_i \in [-5, 10]$. The global minimum is located at $x_i = 1$ and equals to zero (i.e., $F(1, ..., 1) = 0$).

The version implemented in this thesis study is ten-dimensional (i.e., $n = 10$). In Figure B.6, a two-dimensional representation of the Rosenbrock function is shown.

Schwefel

This is also a scalable box-bounded continuous unconstrained problem. It was first introduced in Laguna and Martí (2005) and its generalized objective function can be formulated as follows:

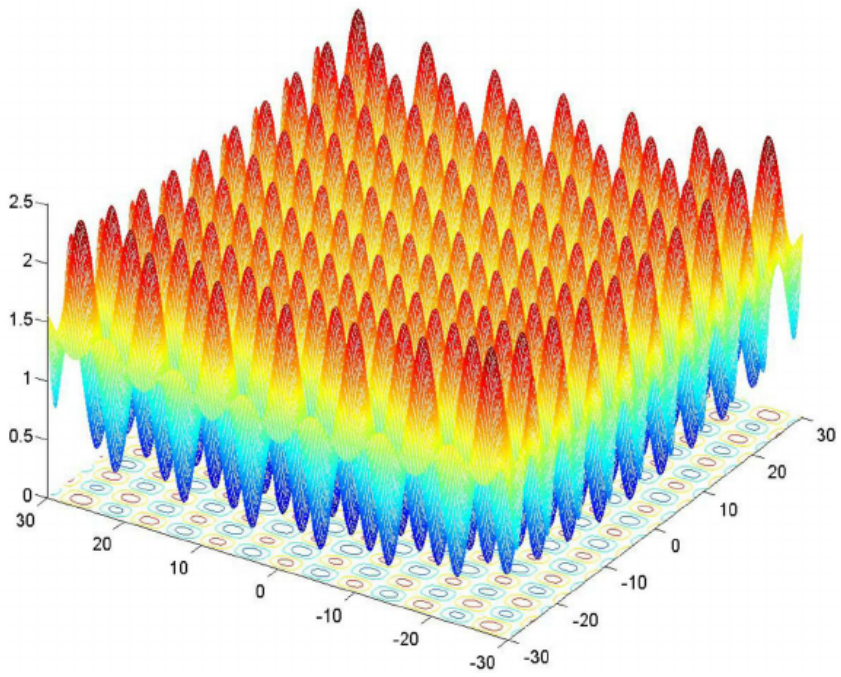


Figure B.5: Two-dimensional Griewank function.

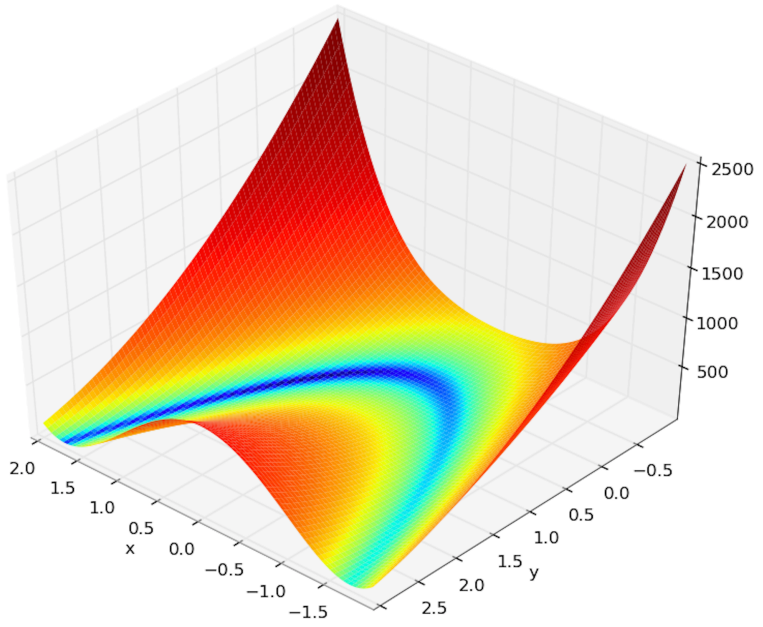


Figure B.6: Two-dimensional Rosenbrock function.

$$F(x_1, \ldots, x_n) = 418.9828872724338n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}) \tag{B.5}$$

where $x_i \in [-500, 500]$. The global minimum is in $x_i = 420.9687$, where the objective function reaches a value of zero (i.e., $F(420.9687, \ldots, 420.9687) = 0$). Its two-dimensional representation can be seen in Figure B.7.

In this research, the four-dimensional Schwefel function was used (i.e., $n = 4$).

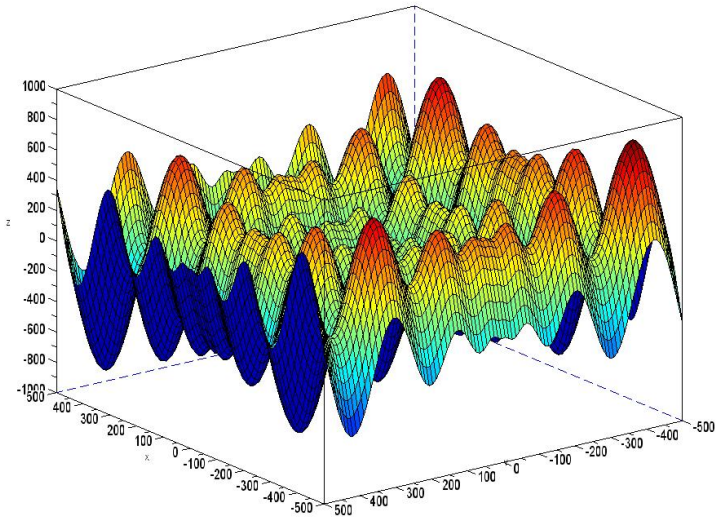


Figure B.7: Two-dimensional Schwefel function.

B.3.1.2 Continuous and Integer Constrained Problems

In this section, we will discuss two different types of constrained problems: integer ones, treated with the Golomb Ruler problem, and continuous ones, handled with 24 different problems introduced in 2006, during the Congress on Evolutionary Computation, in Canada.

Golomb Ruler

This problem was discussed in Babcock (1953) and represents an integer constrained problem. A Golomb ruler with n marks, is a set of n distinct nonnegative integers $(a_1, \ldots, a_n)$ along an imaginary ruler, such that there are not two pairs of marks the same distance apart. The number of marks on the ruler represents its order and the largest distance between two of its marks is its length. If no shorter Golomb ruler of the same order exists, then the Golomb ruler is considered optimal. We thus have a constrained box-bounded integer problem, where we would like to find an optimal Golomb ruler of a given order n. We can also tweak the maximum distance between consecutive ticks (i.e., l_{max}). The decision vector is:

$$\mathbf{x} = [d_1, .., d_{n-1}] \tag{B.6}$$

where d_i indicates the distance between consecutive ticks. The ticks on the ruler are reconstructed as:

$$\begin{aligned} a_0 &= 0 \\ a_i &= \sum_{j=1}^{i} d_i \end{aligned} \tag{B.7}$$

where $i = 1, .., n-1$.

The problem can thus be formulated as follows:

$$\begin{aligned} &\text{Find}: \ 1 \leq d_i \leq l_{max} \ \forall i = 1, \ldots, n-1 \\ &\text{which minimizes:} \ \sum_i d_i \\ &\text{subject to:} \ |a_i - a_j| \neq |a_l - a_m| \ \forall (distinct) i, j, l, m \in [0, n] \end{aligned} \tag{B.8}$$

Table B.4: Golomb Rulers test data, for 40,000 and 200,000 function evaluations.

Golomb Rulers	**ACOmi**	**IHS**	**Known Best**
(n=4, l_{max}=6, fevals=40,000)	6.0	6.0	6
(n=4, l_{max}=6, fevlas=200,000)	6.0	6.0	6
(n=5, l_{max}=11, fevals=40,000)	11.0	11.5	11
(n=5, l_{max}=11, fevals=200,000)	11.0	12.0	11
(n=6, l_{max}=17, fevals=40,000)	17.0	22.8	17
(n=6, l_{max}=17, fevals=200,000)	17.0	23.1	17
(n=7, l_{max}=25, fevals=40,000)	25.4	43.8	25
(n=7, l_{max}=25, fevals=200,000)	25.0	41.7	25

The constraints can easily be transformed into one single equality constraint in the form $c = 0$, where c represents the count of repeated distances.

We chose to minimize four different Golomb ruler with: ($n = 4$, $l_{max} = 6$), ($n = 5$, $l_{max} = 11$), ($n = 6$, $l_{max} = 17$) and ($n = 7$, $l_{max} = 25$). We have chosen these rulers since they have been mathematically proven and their fitness values (and the corresponding optimal decision vectors) are known. We benchmarked two different algorithms for this: improved harmony search (IHS) and ACOmi. The former, is a metaheuristic algorithm said to imitate the improvisation process of musicians. This algorithm, thoroughly described in Mahdavi et al. (2007), was chosen since it can handle integer variables and deal with constraints (whereas the other algorithms like PSO, SADE, etc. cannot deal with integer variables nor handle constraints). The results for 200,000 function evaluations are displayed in Figure B.8. Also, in Table B.4 the numerical results for both 20,000 and 400,000 function evaluations are shown. As we can see, the fitness values are often floats instead of integers: this is just because these numbers are actually an average of ten different runs for each Golomb Ruler. As we can see, ACOmi consistently outperforms IHS and often finds the best fitness for all the tested rulers, thus demonstrating to be competitive also for integer constrained optimization problems.

CEC2006 Test Suite

For benchmarking ACOmi on constrained SO problems with continuous variables, we have decided to test it on 24 constrained SO problems, which were presented during the 2006 IEEE Congress on Evolutionary Computation. During this event, a competition on constrained real-parameter optimization problems was organized. Therefore, several algorithms could be tested on these problems, whose minima are known and whose formulation is mathematically defined.

As we have already discussed, the standard SO algorithms that we have discussed in Chapter 5 are not capable of handling constraints. However, a meta-algorithm exists for allowing SO unconstrained algorithms to deal with constraints as well. It has to be said that ACOmi is capable of handling constraints: it can thus be used either with or without the meta-algorithm. We decided to benchmark three different algorithms: ACOmi, ACOmi with the meta-algorithm and SADE with the meta-algorithm. In our study, we are not very concerned about the performances of the algorithm w.r.t. every other optimization algorithm available, but we just want to make sure that the performances of the algorithm on SO continuous constrained problems are competitive and not clearly outperformed. Besides, we decided to include ACOmi with the meta-algorithm for controlling whether the use of such a meta-algorithm could improve ACOmi itself, for some problems. This information was also helpful for our thesis study itself, where we decided to also include ACOmi with the meta-algorithm. In this thesis, we will not discuss into details the mathematical formulation and the constraint functions of each of the 24 problems, but we will limit ourselves to only show the results. The interested reader can find the definition and properties of each of these problems in Liang et al. (2006). In Figure B.9, B.10, B.11 and B.12, we show a graphical representation of the objective function values against the number of function evaluations

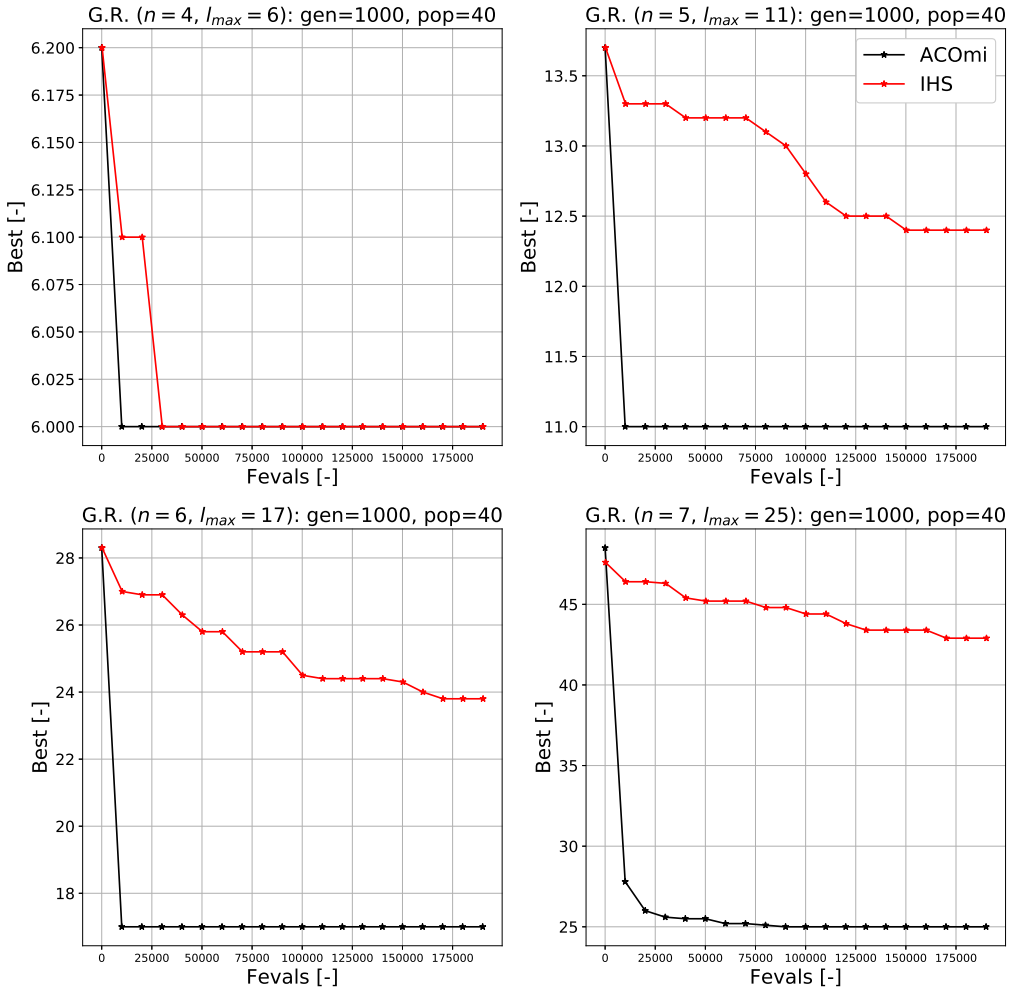


Figure B.8: Golomb Rulers results for 200,000 function evaluations: graphical representation.

for all the 24 test problems.

In particular, this is done on a semi-logarithmic scale, where on the y-axis we place the logarithm of the absolute value of the difference between the found objective value and the real one, at every function evaluation. While on the x-axis we display the number of function evaluations. Each run was averaged over three runs and executed with 500,000 function evaluations. However, as we can observe, some algorithms do not reach that function evaluation value, but they stop before. The reason is twofold: first of all, as it happens for ACOmi without the meta-algorithm in problem 12, it happens that the algorithm reaches the real minimum: this causes the algorithm not to be defined anymore (since it would be the logarithm of zero, which results to be minus infinity). Furthermore, the second reason is related to the implementation of the meta-algorithm: this method is constructed in such a way that if the same individuals happen to be chosen again throughout the iterations, then their function evaluations are not evaluated again but a sort of memory mechanism is implemented for evaluating the fitness values only once for each individual. Hence, it may happen that the function evaluations are not the expected ones, but they are less than the expected values.

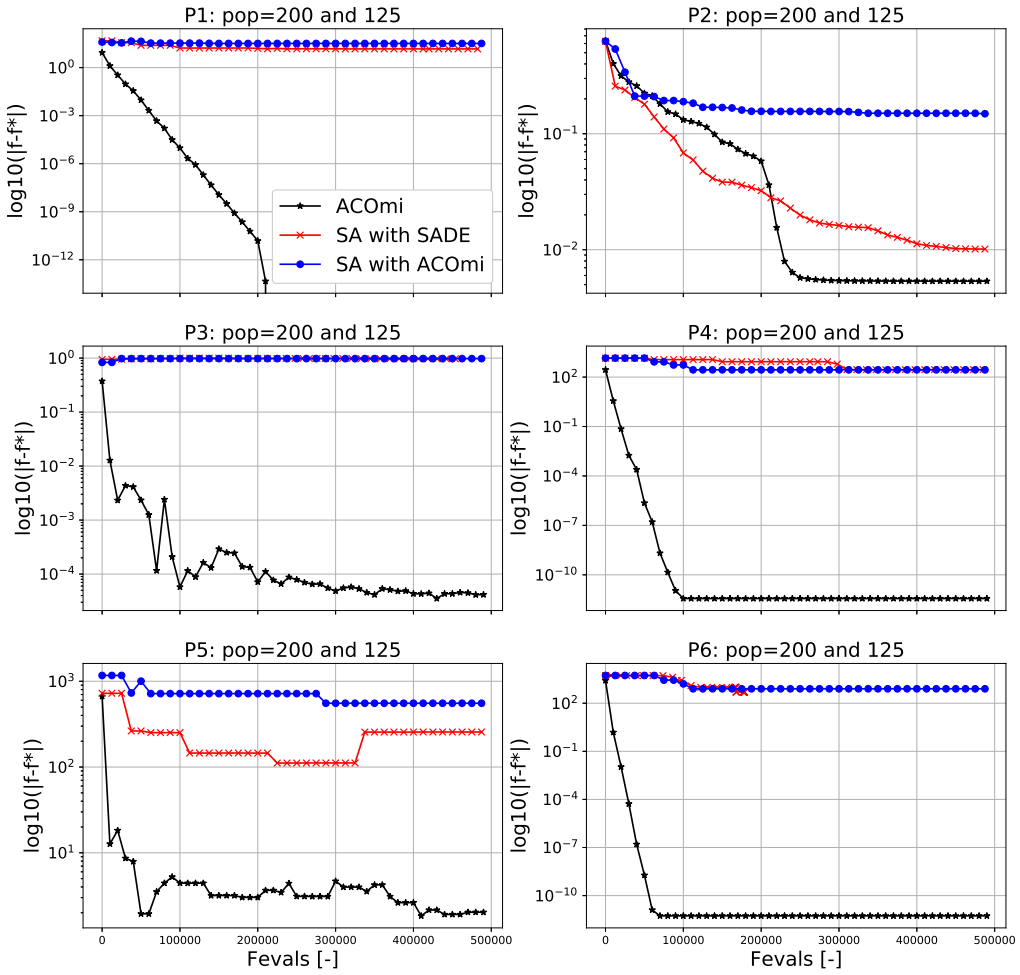


Figure B.9: CEC2006 benchmark: we hereby show the results of problems 1, 2, 3, 4, 5 and 6, in terms of objective function values w.r.t. the real optimum, plotted as a function of the function evaluations.

This behavior, for instance, can be seen in problems 11 and 12 for SADE with the meta-algorithm. An aspect to be noted is that for each of these problems, the oracle parameter was chosen to be equal to the best objective function value for each problem (since this was given in the problems). Also, the other parameters are: $ker = 100$ for the case without the meta-algorithm, and $ker = 125$ for the case with the algorithm, $q = 1$, $acc = 0$, $NGenMark = 7$. Concerning the $threshold$ parameter, it holds $threshold = 1000$ without the meta-algorithm, and $threshold = 80$ with the meta-algorithm. Also, all the algorithms with the meta-algorithm are run for 40 iterations, with a generation size of 100 and a population size of 125. Whereas the ACOmi algorithm alone is run with a generation size of 2500 and a population size of 200 individuals. This is done for having the same nominal number of function evaluations.

As we can observe from the graphical data, ACOmi without meta-algorithm seems to outperform the other algorithms in most of the problems, thus demonstrating to be a very competitive algorithm also for constrained optimization problems.

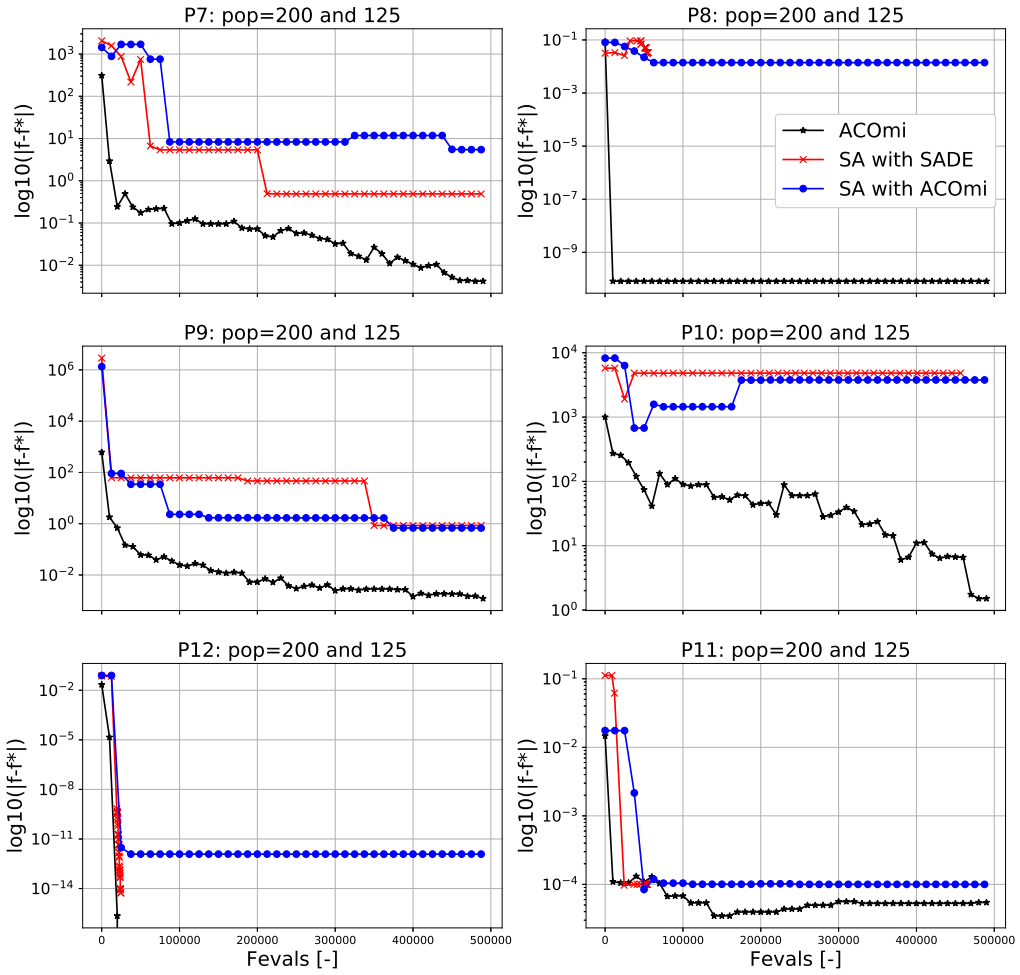


Figure B.10: CEC2006 benchmark: we hereby show the results of problems 7, 8, 9, 10, 11 and 12, in terms of objective function values w.r.t. the real optimum, plotted as a function of the function evaluations.

B.3.1.3 Global Trajectory Optimization Problems

The global trajectory optimization problems are a set of box-bounded space problems often used in literature to test global trajectory optimization algorithms on space applications (Izzo, 2010), (Vinkó et al., 2007b), (Vinkó and Izzo, 2008). These problems take inspiration from real interplanetary trajectories such as Rosetta, Cassini, Messenger, Earth-Mars transfers, etc., and their thorough description can be found in the aforesaid references. In particular, we have decided to test three well-known metaheuristic algorithms (i.e., SGA, PSO and SADE) against our mixed integer ant colony optimization algorithm (i.e., ACOmi). All these tests have been performed by running these algorithms on six different problems:

1. Cassini 2: a Cassini-inspired multiple gravity assist (MGA) problem, with a deep space maneuver (DSM). This problem is thus inspired by the Cassini spacecraft interplanetary journey to Saturn. The final orbit around Saturn is set to a pericenter radius of r_p = 108950 km and an eccentricity of e = 0.98. The spacecraft makes the following planets' sequences: Earth-Venus-Venus-Earth-Jupiter-Saturn, which is the same as the real

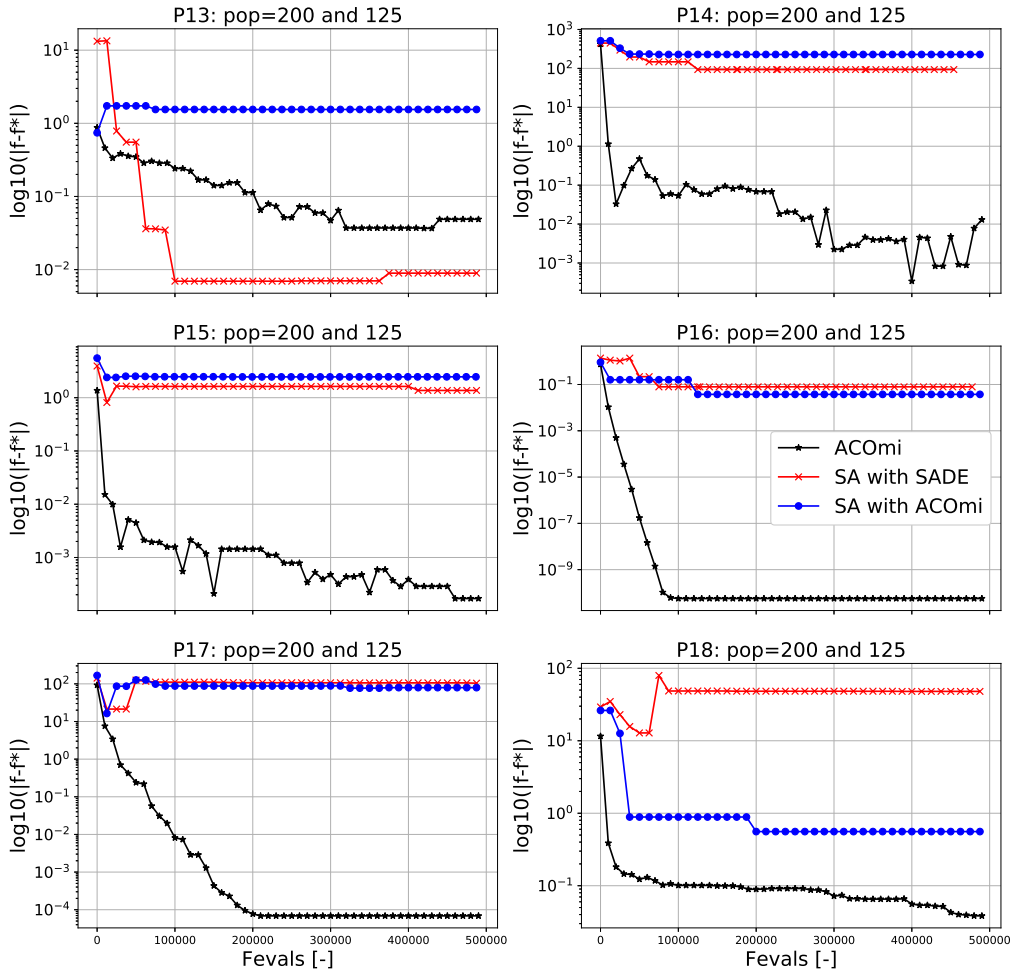


Figure B.11: CEC2006 benchmark: we hereby show the results of problems 13, 14, 15, 16, 17 and 18, in terms of objective function values w.r.t. the real optimum, plotted as a function of the function evaluations.

Cassini mission.

2. E-V-E MGA 1DSM: an Earth-Venus-Earth multiple gravity assists problem, with a deep space maneuver allowed for each leg.
3. Messenger: a Messenger-inspired MGA 1 DSM problem. This mission is a rendezvous to Mercury, modeled with a multiple gravity assist and one deep space maneuver. The fly-by sequence is Earth-Venus-Venus-Mercury-Mercury-Mercury. This represents a particularly difficult and complex mission to design since the many fly-bys and the possible resonances due to the planets' configurations make the global optimization techniques struggle to find the best solution.
4. Rosetta: this is a problem inspired by the real Rosetta mission, which is a rendezvous mission to the comet 67P/Churyumov-Gerasimenko. The selected fly-by sequence is Earth-Earth-Mars-Earth-Earth-Comet. Deep space maneuvers are also included.
5. E-M 5 imp: this is an Earth Mars transfer with 5 impulses.

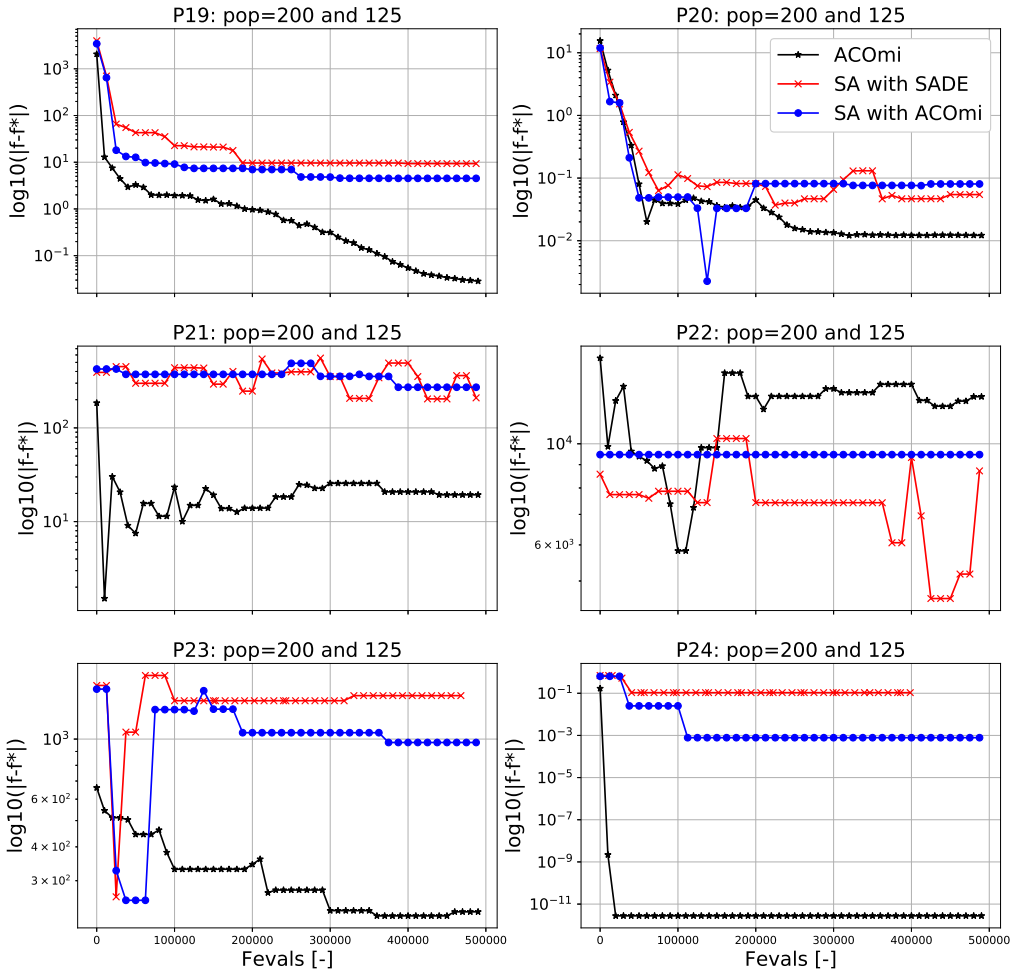


Figure B.12: CEC2006 benchmark: we hereby show the results of problems 19, 20, 21, 22, 23 and 24, in terms of objective function values w.r.t. the real optimum, plotted as a function of the function evaluations.

6. E-M 7 imp: this is the same as the one above, but 7 impulses are used, thus making the problem slightly more difficult.

All these problems have the same objective functions, that is the total ΔV used for the mission. Of course, the lower this value, the better.

These problems have been optimized using two different population sizes: 20 and 200, over 1000 generations (for a total of 20,000 and 200,000 function evaluations, respectively). Also, each optimizer was run ten times with ten different controlled seeds, to make sure to remove the randomness and have trustworthy results. Also, all the optimizers were always started with the same population of individuals. The results corresponding to 200,000 function evaluations are shown graphically in Figure B.13. Also, for both 20,000 and 200,000 function evaluations we display the numerical results in Tables B.5 and B.6.

Overall, we can see that ACOmi has competitive results with the other optimizers, sometimes also outperforming them (e.g., in the case of Rosetta). Thus, having tested this algorithm on both mathematical and space trajectory problems and having demonstrated that its

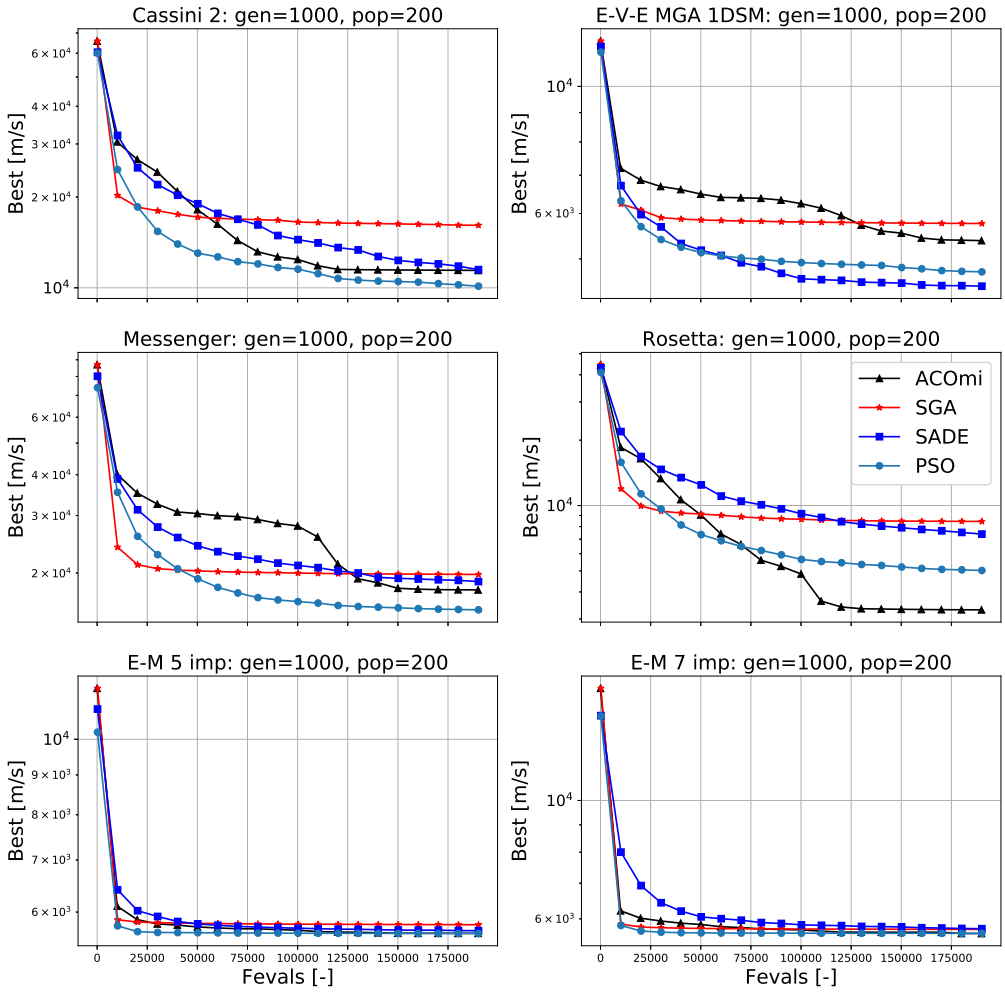


Figure B.13: ACOmi, PSO, SADE and SGA performances on 6 global trajectory optimization problems for 200,000 function evaluations.

performances are competitive with the current state-of-art optimizers, we consider it as verified and validated. Also, these problems have helped us to determine how to tweak and tune the algorithm depending on the type of problems and the population and generation sizes. Furthermore, these test problems have also been used in the design phase of the algorithm, for enhancing its performances and adjust its behavior throughout the evolution process.

B.3.2. Multi-Objective V&V

Concerning MO optimization, two new algorithms have been developed and implemented in PaGMO, in this thesis study. The first one was derived from literature: the nondominated sorting particle swarm optimizer (NSPSO), whereas the second was coded from scratch, starting from the idea of ACOmi and extending it for multi-objective using the hypervolume concept: the multi-objective hypervolume-based ant colony optimizer (MHACO). After having tested the usage of both these algorithms, they have been benchmarked on three different test-suites: the Zitzler, Deb and Thiele (ZDT) test suite (Zitzler et al., 2000), the Deb, Thiele,

Table B.5: Best found values (in terms of ΔV, measured in m/s) of ACOmi, SGA, PSO and SADE on a population size of 20for 6 global trajectory optimization problems.

Problems	**ACOmi**$_{20}$	**SGA**$_{20}$	**SADE**$_{20}$	**PSO**$_{20}$
Cassini 2	15113.7	23294.8	15586.9	15963.4
E-V-E	5638.4	6397.8	5261.0	5425.5
Messenger	21978.2	33376.6	22410.0	18765.1
Rosetta	6376.9	17801.6	10186.1	8127.2
E-M 5	5696.5	5949.7	5775.9	5745.1
E-M 7	5706.2	6555.8	5848.7	5659.9

Table B.6: Best found values (in terms of ΔV, measured in m/s) of ACOmi, SGA, PSO and SADE on a population size of 200 for 6 global trajectory optimization problems.

Problems	**ACOmi**$_{200}$	**SGA**$_{200}$	**SADE**$_{200}$	**PSO**$_{200}$
Cassini 2	9339.9	13525.8	11742.9	9189.1
E-V-E	5236.6	5935.6	4553.2	4652.3
Messenger	19618.5	18621.8	16840.3	15305.7
Rosetta	3258.8	7636.2	7017.6	4129.7
E-M 5	5630.6	5760.2	5684.8	5632.2
E-M 7	5637.4	5759.3	5732.9	5635.9

Laumanns and Zitzler (DTLZ) test suite (Deb et al., 2005) and the Walking Fish Group (WFG) test suite (Huband et al., 2006). The ZDT and DTLZ test suites are commonly used test suites. The first one, only concerns bi-objective problems, whereas the latter has the advantage to include scalable fitness problems, thus also allowing to test the algorithms for many objectives. However, both these test problems have some drawbacks and do not cover certain types of problems. For this reason, the WFG was introduced and used also as a benchmark set (Huband et al., 2006). Of course, for all these test-suites the exact ideal Pareto front is known, and the found Pareto fronts can directly be confronted with that.

The objective of this V&V is to check whether the two implemented algorithms (i.e., MHACO and NSPSO) can be competitive with MOEA/D and NSGA-II (i.e., two widely used and tested optimizers) for MO optimization. The input parameters of these algorithms have been kept as constant as possible, for avoiding a specific tweaking and tuning of the algorithm for each of these problems, which would have been very time consuming and problem specific. In particular, for NSGA-II and MOEA/D, the input values presented in Section 6.2.2 were used, as they demonstrated to have good performances on most of the problems. For MHACO the following input parameters were chosen: $ker = pop_{size}$, $q = 1$, $threshold = 128$, $NGenMark = 27$, $focus = 0$ and the $evalstop$ parameter was not activated, while for NSPSO the niche count diversity mechanism was used for most of the problems and the following input parameters were used for most of the problems: $\omega = 0.4$, $\eta_1 = 2$, $\eta_2 = 2.0$, $V_{MAX} = 0.5$, $LSR = 50$. Whenever either MHACO or NSPSO resulted to be clearly outperformed by the other algorithms in terms of p-distance, hypervolume and number of Pareto-optimal individuals, the input parameters have been tweaked for checking whether a more competitive performance could be achieved.

Of course, by tweaking and tuning the algorithm we can expect that it improves: ideally, depending on the type of problem, a set of input parameters exists for the algorithm to optimize that problem in the most efficient way. However, we do not focus on this aspect, and we thus do not spend too much time enhancing the performances, but we rather verify the competitiveness of these algorithms. We hence only observe whether the results produced in terms of various performance criteria (i.e., p-distance, hypervolume, number of Pareto solutions in the last front, etc.) are similar among the algorithms. For having the possibility to inspect the results, we have decided to only run bi-objective problems.

From the results we got, we conclude that the two algorithms have competitive results w.r.t. both MOEA/D and NSGA-II. Besides, they sometimes outperform them in some problems. We always have to keep in mind that the purpose of this study was not to improve previous algorithms' performance on these test-suites, but rather to figure out whether NSPSO

and MHACO could be competitive with the widespread MO optimizers. Of course, if we would like to improve these results even more tweaking and tuning should be done to find the most efficient input parameters of these algorithms for achieving better results. Also, we have not focused our study on many objectives optimization (i.e., 10, 50, 100 or more objectives), as this is a separate field, which was not directly related to our research.

This benchmark has been done using three population sets (i.e., 32, 64 and 128), evolving each algorithm for 250 generations, and by executing each run ten times for removing the randomness and with a controlled seed, to have the same initial populations. The results of this benchmark are analyzed in terms of hypervolume values (i.e., the higher the better), p-distances values (the lower the better) and number of Pareto-optimal individuals in the final population. Concerning the p-distance and hypervolume values, these results have been divided into two sets: one where we display the results of MHACO against both NSGA-II and MOEA/D, for the three different population sizes and for both the hypervolume and the p-distance, and another one where we do the same comparisons but for NSPSO.

Concerning MHACO, the results coming from this validation phase can be seen in Tables B.7, B.8 and B.9 for the hypervolume values, and in Tables B.10, B.11 and B.12 for the p-distance values. Moreover, the same number of tables was also produced for showing the results of NSPSO against NSGA-II and MOEA/D. These numerical results can be seen in Tables B.13, B.14 and B.15 for the hypervolume values, and in Tables B.16, B.17 and B.18 for the p-distance values. On the other hand, concerning the number of Pareto-optimal individuals, we have grouped all the four algorithms together and we have shown the results in Tables B.19, B.20 and B.21. Each value in these tables is an average over three runs.

From these results, we can notice that NSPSO seems to struggle more on ZDT4, DTLZ1 and DTLZ3, whereas MHACO seems to be outperformed on ZDT4, DTLZ3 and DTLZ6. This aspect may be adjusted with a more refined and thorough tweaking and tuning of the input values. This, however, results to be time consuming and to go beyond the scope of this V&V procedure, where we wanted to demonstrate that in most of these problems the algorithms behavior could be verified in terms of both usage (i.e., they could run without issues) and performances (i.e., they could be competitive with MOEA/D and/or NSGA-II in most of these problems).

For completeness, in the following sections, we will provide the mathematical description of all the employed test-suites.

B.3.2.1 ZDT test suite

This is a wide-spread test suite, which was conceived for bi-objective problems (Zitzler et al., 2000). It consists of 6 problems: the first four and the last one are all continuous variables' problems, whereas the 5th one is an integer problem. As it is often done for comparing algorithms that cannot deal with integer variables (as it happens for MOEA/D), for comparing the algorithms we have thus decided to only use the 5 continuous problems.

All these problems are minimization problems, in which we would like to minimize both the two fitnesses ($f_1(\mathbf{x})$, $f_2(\mathbf{x})$). In particular, each of these test functions is structured in the same manner and it consists of three functions: f_1, g, h:

$$\begin{aligned}&\text{Minimize } T(\mathbf{x}) = (f_1(x_1), f_2(\mathbf{x}))\\&\text{Subject to } f_2(\mathbf{x}) = g(x_2, \dots, x_n)h(f_1(x_1), g(x_2, \dots, x_n))\\&\text{where } \mathbf{x} = (x_1, \dots, x_n)\end{aligned}$$

where n is the variables' dimension. The authors recommend a certain dimension for each of the problems: $n = [30, 30, 30, 10, 11, 10]$. These dimensions will thus be used in our case.

Each of the problems of the test suite can be formulated as follows:

ZDT1

This is a continuous box-constrained bi-objective problem, where:

Table B.7: Hypervolume trade-off (MHACO vs MOEA/D vs NSGA2 on a population size of 32).

Problems	**MOEA/D 32**	**MHACO 32**	**NSGA2 32**
ZDT1	5.60521416	5.56493892	5.71608537
ZDT2	10701.75495905	10607.88202384	10727.701467
ZDT3	5.29511749	5.75015199	6.11519368
ZDT4	581.84487543	581.48477781	677.77787996
ZDT5	/	/	/
ZDT6	4.56833663	3.54344788	5.75409382
DTLZ1	91682.30167486	91682.57068519	91681.63579175
DTLZ2	0.38960076	0.38106638	0.40068458
DTLZ3	578869.31173575	578679.55732527	578832.53370347
DTLZ4	10165.33069082	10116.06537193	10131.94020926
DTLZ5	10199.28558643	10179.82240874	10199.27578719
DTLZ6	10979.2034656	10972.05072546	10978.99285502
DTLZ7	10208.80711175	10246.58853106	10208.81595428
WFG1	10162.91009828	10069.74583633	10078.68031725
WFG2	6.06884694	6.30198718	5.68517331
WFG3	7.12206994	6.46126271	7.13875659
WFG4	7.37976975	6.37102619	7.39635473
WFG5	7.34657575	6.67163127	7.53551429
WFG6	8.94916903	9.14174821	9.33939618
WFG7	10471.58869241	10361.93596388	10462.63451077
WFG8	7.69959253	6.18514162	6.89729072
WFG9	9.78333785	9.40460617	9.57984289

Table B.8: Hypervolume trade-off (MHACO vs MOEA/D vs NSGA2 with a population size of 64).

Problems	**MOEA/D 64**	**MHACO 64**	**NSGA2 64**
ZDT1	5.81072491	5.80454584	5.86159956
ZDT2	10736.54350845	10679.29104344	10754.77433633
ZDT3	5.91961744	6.10988226	6.24685769
ZDT4	606.32764414	534.76262348	722.77253513
ZDT5	/	/	/
ZDT6	5.32037729	4.35762678	6.48063238
DTLZ1	92671.46194399	92671.5274698	92670.70029827
DTLZ2	0.41223905	0.40765706	0.41691899
DTLZ3	578869.31839251	578744.98009298	578867.92288507
DTLZ4	10199.70072334	10188.45150817	10166.30179513
DTLZ5	10200.17944552	10191.58144326	10200.17350132
DTLZ6	10981.27593247	10976.15211454	10981.23305192
DTLZ7	10253.78410107	10251.96564207	10211.56391154
WFG1	10157.26691424	10090.3965477	10128.03079218
WFG2	7.80470365	7.82207872	7.79276901
WFG3	9.20475751	8.95044206	9.20262895
WFG4	8.52791941	7.94013916	8.52993563
WFG5	8.33974337	8.05953149	8.43981338
WFG6	9.55582369	9.69685009	9.91480153
WFG7	10603.92028265	10442.2431013	10525.06650005
WFG8	8.37685827	7.23600575	7.76047001
WFG9	9.68939911	9.63522549	9.56199282

Table B.9: Hypervolume trade-off (population size: 128).

Problems	**MOEA/D 128**	**MHACO 128**	**NSGA2 128**
ZDT1	5.87010927	5.88042026	5.90338596
ZDT2	10761.36515338	10732.0327118	10773.56051095
ZDT3	6.23942464	6.18568337	6.32594527
ZDT4	683.47552524	676.85316254	747.59938023
ZDT5	/	/	/
ZDT6	5.10448042	4.80593587	6.77723054
DTLZ1	99929.03782669	99929.04286897	99928.915215
DTLZ2	0.42182439	0.42280552	0.42457594
DTLZ3	629101.38897944	629029.37151961	629100.6033699
DTLZ4	10200.02790986	10194.51854924	10200.02849473
DTLZ5	10200.54000741	10200.23248696	10200.53602246
DTLZ6	10989.36256178	10986.53363978	10989.31126731
DTLZ7	10213.209114	10255.14298125	10255.48135147
WFG1	10162.87629543	10090.92714595	NaN
WFG2	8.01762875	7.90089083	7.46785127
WFG3	10.39828301	10.30284008	10.41410762
WFG4	8.65030758	8.172753	8.69189383
WFG5	8.54447338	8.3623464	8.59974148
WFG6	9.65725346	9.76008362	9.91668875
WFG7	10624.38388636	10505.69518352	10585.1801076
WFG8	8.99005078	8.21929392	8.46910645
WFG9	10.0340789	9.6748688	9.79291525

Table B.10: P-distance trade-off (MHACO vs MOEA/D vs NSGA2 on a population size of 32 individuals).

Problems	**MOEA/D 32**	**MHACO 32**	**NSGA2 32**
ZDT1	0.16327763	0.12611162	0.0293881
ZDT2	0.07069111	0.2724249	0.01231875
ZDT3	0.98580786	0.42460238	0.01847673
ZDT4	57.09439147	144.0521526	23.4321128
ZDT5	/	/	/
ZDT6	2.54834424	3.84769796	1.17727217
DTLZ1	1.55304507	0.57038582	2.04855862
DTLZ2	3.40883082×10^{-4}	0.00145649	2.20774811×10^{-5}
DTLZ3	0.16703657	16.19169878	5.21160793
DTLZ4	0.00027264	0.00056951	2.06165852×10^{-5}
DTLZ5	3.40883082×10^{-4}	0.00145649	2.20774811×10^{-5}
DTLZ6	0.01181132	1.7947661	0.13342154
DTLZ7	4.17646050×10^{-4}	0.00234079	9.31748365×10^{-5}

Table B.11: P-distance trade-off (MHACO vs MOEA/D vs NSGA2 with a population size of 64).

Problem	**MOEA/D 64**	**MHACO 64**	**NSGA2 64**
ZDT1	0.05413535	0.04372473	0.00471381
ZDT2	0.05046506	0.11091524	0.00281303
ZDT3	0.43830665	0.20257633	0.00465985
ZDT4	54.00833935	246.34576719	5.46353146
ZDT5	/	/	/
ZDT6	1.69297054	2.88108694	0.42782386
DTLZ1	6.20793802×10^{-2}	0.02006755	1.66724282
DTLZ2	1.75237798×10^{-4}	0.00072231	7.35852363×10^{-5}
DTLZ3	0.02845748	10.77884714	0.67055369
DTLZ4	0.00018812	0.00070195	8.06493553×10^{-7}
DTLZ5	1.75237798×10^{-4}	0.00072231	7.35852363×10^{-5}
DTLZ6	0.00253952	1.78315624	0.03807964
DTLZ7	2.70626013×10^{-4}	0.00092652	3.83535612×10^{-8}

Table B.12: P-distance trade-off (MHACO vs MOEA/D vs NSGA2 with a population size of 128).

Problems	**MOEA/D 128**	**MHACO 128**	**NSGA2 128**
ZDT1	0.03877232	0.02380317	0.00124501
ZDT2	0.0341838	0.06162727	0.00073551
ZDT3	0.11202489	0.20323239	0.00137918
ZDT4	35.05038224	91.75185272	2.63902538
ZDT5	/	/	/
ZDT6	2.01597067	2.46467676	0.20803473
DTLZ1	1.20577628×10^{-3}	0.00609794	0.33564028
DTLZ2	9.10006010×10^{-5}	0.00039529	6.00403863×10^{-6}
DTLZ3	0.00076059	9.87855792	0.34197763
DTLZ4	0.00010256	0.00027907	8.64787851×10^{-6}
DTLZ5	9.10006010×10^{-5}	0.00039529	6.00403863×10^{-6}
DTLZ6	0.00139555	1.2149684	0.03593371
DTLZ7	8.69694479×10^{-5}	0.00064627	5.34667770×10^{-6}

Table B.13: Hypervolume trade-off (NSPSO vs MOEA/D vs NSGA2 on a population size of 32).

Problems	**MOEA/D 32**	**NSPSO 32**	**NSGA2 32**
ZDT1	3.4559696	3.65426523	3.65451556
ZDT2	10102.43382865	10066.78311812	10152.06732687
ZDT3	10238.76977339	10279.58897337	10294.48863904
ZDT4	1620103.99232668	1457062.74302579	1647999.33983491
ZDT5	/	/	/
ZDT6	10207.92979539	10361.68249398	10364.49183578
DTLZ1	162166.95071301	161404.0699437	162166.27617563
DTLZ2	120.24966975	120.25247392	120.25868146
DTLZ3	2755639.27932227	2731065.81555699	2755602.11006836
DTLZ4	113.74118677	107.61308381	110.68345
DTLZ5	0.21805835	0.21213144	0.22824271
DTLZ6	3.17228889	3.16658622	2.93257983
DTLZ7	118.26426256	114.19736833	118.23772989
WFG1	151.9621354	153.25765732	142.05304443
WFG2	148.44920629	151.65382838	145.2237346
WFG3	161.21943705	159.79856202	161.10061822
WFG4	161.55637372	157.65677691	161.05953566
WFG5	158.12581162	153.29335851	159.18566119
WFG6	204.18822433	202.61932173	207.44871992
WFG7	159.66746736	161.17866082	158.88543323
WFG8	11.82022289	8.82337507	10.92193094
WFG9	36.77906309	31.08277254	35.66646098

Table B.14: Hypervolume trade-off (NSPSO vs MOEA/D vs NSGA2 with a population size of 64).

Problems	**MOEA/D 64**	**NSPSO 64**	**NSGA2 64**
ZDT1	3.59452737	3.67207367	3.68554132
ZDT2	10186.77092383	10200.90063168	10200.63813395
ZDT3	10247.58491252	10261.74982319	10267.31444963
ZDT4	1637497.71811025	1432633.6028558	1668646.4787854
ZDT5	/	/	/
ZDT6	10290.50432235	10366.77202485	10451.29821172
DTLZ1	163488.93427329	162703.81033471	163488.16438284
DTLZ2	120.2814699	120.2787037	120.28509672
DTLZ3	2755639.29692633	2755639.29692633	2755637.90679656
DTLZ4	112.89361204	106.75573159	109.82948574
DTLZ5	0.22991069	0.22880069	0.23439078
DTLZ6	3.19737837	3.20153301	3.12475714
DTLZ7	122.17117863	114.08727148	117.84822336
WFG1	147.84821167	157.27602915	144.61813845
WFG2	156.29556735	164.26574756	156.1857532
WFG3	163.44787192	163.36748104	163.11060693
WFG4	162.25310911	157.89345214	161.7437617
WFG5	158.90905784	155.29108461	159.51017757
WFG6	191.15602256	167.16308986	196.97391454
WFG7	162.55675403	158.04908735	154.56909363
WFG8	14.17534868	10.66209341	12.5348636
WFG9	36.89508556	29.1764626	35.45347331

Table B.15: Hypervolume trade-off (NSPSO vs MOEA/D vs NSGA2 with a population size of 128).

Problems	**MOEA/D 128**	**NSPSO 128**	**NSGA2 128**
ZDT1	3.6449406	3.69538358	3.7056356
ZDT2	10186.01332428	10200.89010093	10200.91791574
ZDT3	10275.95015023	10278.79550239	10281.09391726
ZDT4	1653653.43798041	1562028.55352806	1681648.1216013
ZDT5	/	/	/
ZDT6	10213.36466078	10480.87780418	10443.50243847
DTLZ1	172757.56305401	172161.86888352	172757.44156658
DTLZ2	120.26497643	120.26412721	120.26706026
DTLZ3	2864635.38062922	2851957.36175022	2864634.6099994
DTLZ4	116.58519013	113.51500536	116.587623
DTLZ5	0.23385066	0.23309494	0.23662311
DTLZ6	3.20612182	3.20842751	3.12506959
DTLZ7	120.85042545	120.84323571	124.88872398
WFG1	148.69818482	160.57401908	NaN
WFG2	159.0969238	164.32238141	152.90882144
WFG3	163.77689916	163.62271568	163.73830881
WFG4	163.09510471	160.31315305	163.29396045
WFG5	159.24076693	154.80489979	159.54917593
WFG6	170.24049236	171.86691343	174.14211336
WFG7	165.66924623	164.09164029	161.63688916
WFG8	14.96802716	11.12221898	13.04781365
WFG9	37.08577863	31.04443055	36.72290612

Table B.16: P-distance trade-off (NSPSO vs MOEA/D vs NSGA2 on a population size of 32 individuals).

Problems	**MOEA/D 32**	**NSPSO 32**	**NSGA2 32**
ZDT1	0.16327763	0.01002195	0.0293881
ZDT2	0.05434864	0.20785872	0.00894705
ZDT3	0.98580786	0.138507	0.01847673
ZDT4	48.8124939	200.28145259	10.95756434
ZDT5	/	/	/
ZDT6	2.54834424	1.75335143	1.17727217
DTLZ1	1.55304507	4.16011842	2.04855862
DTLZ2	3.40883082×10^{-4}	0.00190017	2.20774811×10^{-5}
DTLZ3	0.16703657	164.60083575	5.21160793
DTLZ4	0.00027264	0.01903876	2.06165852×10^{-5}
DTLZ5	2.23272826×10^{-4}	0.0022779	1.31845948×10^{-5}
DTLZ6	0.01050075	0	0.1552787
DTLZ7	9.31748365×10^{-5}	0.00018991	4.17646050×10^{-4}

Table B.17: P-distance trade-off (NSPSO vs MOEA/D vs NSGA2 with a population size of 64).

Problems	**MOEA/D 64**	**NSPSO 64**	**NSGA2 64**
ZDT1	0.03650967	0.00758809	0.00273574
ZDT2	0.02116328	0.00231174	0.00133105
ZDT3	0.31568176	0.01025467	0.00350863
ZDT4	46.44798895	240.71529621	6.63590182
ZDT5	/	/	/
ZDT6	1.03850418	1.31309502	0.26417885
DTLZ1	6.20793802×10^{-2}	5.59874658	1.66724282
DTLZ2	1.15180817×10^{-4}	0.00122837	2.87010912×10^{-6}
DTLZ3	0.02845748	170.29164529	0.67055369
DTLZ4	0.0001362	0.00046277	2.73442967×10^{-7}
DTLZ5	1.00479667×10^{-4}	0.00110522	1.71645715×10^{-5}
DTLZ6	0.00079428	0	0.03578814
DTLZ7	1.93409100×10^{-4}	0.00048034	1.15131103×10^{-8}

Table B.18: P-distance trade-off (NSPSO vs MOEA/D vs NSGA2 with a population size of 128).

Problems	**MOEA/D 128**	**NSPSO 128**	**NSGA2 128**
ZDT1	0.03877232	0.00666478	0.00124501
ZDT2	0.04648895	0.0037544	0.00068299
ZDT3	0.11202489	0.00954386	0.00137918
ZDT4	39.72507701	120.81867099	3.28794181
ZDT5	/	/	/
ZDT6	2.01597067	0.02501404	0.20803473
DTLZ1	1.20577628×10^{-3}	3.0494438	0.33564028
DTLZ2	9.10006010×10^{-5}	0.00105327	6.00403863×10^{-6}
DTLZ3	0.00076059	88.01688417	0.34197763
DTLZ4	0.00010256	0.0005665	8.64787851×10^{-6}
DTLZ5	9.51934365×10^{-5}	0.00093886	1.01001907×10^{-5}
DTLZ6	0.00123816	0	0.05327033
DTLZ7	8.69694479×10^{-5}	0.0004422	5.34667770×10^{-6}

Table B.19: Number of Pareto-optimal individuals in the final population (MOEA/D vs MHACO vs NSPSO vs NSGA2 with a population size of 32).

Problems	**MOEA/D 32**	**MHACO 32**	**NSPSO 32**	**NSGA2 32**
ZDT1	31	32	21	32
ZDT2	30	22	29	32
ZDT3	31	32	11	32
ZDT4	13	23	1	21
ZDT5	/	/	/	/
ZDT6	15	18	22	26
DTLZ1	29	30	16	32
DTLZ2	32	32	11	32
DTLZ3	30	19	2	32
DTLZ4	32	26	14	32
DTLZ5	32	32	26	32
DTLZ6	32	32	32	32
DTLZ7	32	32	24	32
WFG1	31	20	21	32
WFG2	30	32	11	32
WFG3	31	32	31	32
WFG4	32	21	32	32
WFG5	32	32	32	32
WFG6	6	32	5	32
WFG7	32	32	16	32
WFG8	32	32	10	32
WFG9	11	32	7	32

Table B.20: Number of Pareto-optimal individuals in the final population (MOEA/D vs MHACO vs NSPSO vs NSGA2 with a population size of 64).

Problems	**MOEA/D 64**	**MHACO 64**	**NSPSO 64**	**NSGA2 64**
ZDT1	62	64	41	64
ZDT2	60	41	38	64
ZDT3	49	64	12	64
ZDT4	11	17	1	33
ZDT5	/	/	/	/
ZDT6	24	20	57	60
DTLZ1	63	64	38	64
DTLZ2	64	64	18	64
DTLZ3	60	20	1	64
DTLZ4	64	64	49	64
DTLZ5	64	64	54	64
DTLZ6	64	31	64	64
DTLZ7	64	64	45	64
WFG1	29	12	28	64
WFG2	64	64	20	64
WFG3	64	64	60	64
WFG4	64	25	64	64
WFG5	64	64	63	64
WFG6	5	64	20	64
WFG7	64	64	38	64
WFG8	64	64	19	64
WFG9	19	64	7	64

Table B.21: Number of Pareto-optimal individuals in the final population (MOEA/D vs MHACO vs NSPSO vs NSGA2 with a population size of 128).

Problems	**MOEA/D 128**	**MHACO 128**	**NSPSO 128**	**NSGA2 128**
ZDT1	126	128	73	128
ZDT2	110	94	73	128
ZDT3	103	128	28	128
ZDT4	26	19	1	52
ZDT5	/	/	/	/
ZDT6	32	18	127	117
DTLZ1	128	128	68	128
DTLZ2	128	128	26	128
DTLZ3	126	18	21	128
DTLZ4	128	128	100	128
DTLZ5	128	128	105	128
DTLZ6	128	42	128	128
DTLZ7	128	128	75	128
WFG1	28	20	43	0
WFG2	128	128	128	128
WFG3	128	128	115	128
WFG4	128	34	128	128
WFG5	128	128	125	128
WFG6	20	128	34	128
WFG7	128	128	81	128
WFG8	125	127	27	128
WFG9	35	128	28	128

$$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \tag{B.9}$$

$$f_1(\mathbf{x}) = x_1 \tag{B.10}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(x)}\right] \tag{B.11}$$

with $x_i \in [0,1]\ \forall i$.

ZDT2

This is a continuous box-constrained bi-objective problem, where:

$$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \tag{B.12}$$

$$f_1(\mathbf{x}) = x_1 \tag{B.13}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \left(x_1/g(x)\right)^2\right] \tag{B.14}$$

with $x_i \in [0,1]\ \forall i$.

ZDT3

This is a continuous box-constrained bi-objective problem, where:

$$g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \tag{B.15}$$

$$f_1(\mathbf{x}) = x_1 \tag{B.16}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(x)} - x_1/g(\mathbf{x}) \sin(10\pi x_1)\right] \tag{B.17}$$

with $x_i \in [0,1]\ \forall i$.

ZDT4
This is a continuous box-constrained bi-objective problem, where:

$$g(\mathbf{x}) = 91 + \sum_{i=2}^{n}\left[x_i^2 - 10\cos 4\pi x_i\right] \tag{B.18}$$

$$f_1(\mathbf{x}) = x_1 \tag{B.19}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(x)} - x_1/g(\mathbf{x}) \cos(10\pi x_1)\right] \tag{B.20}$$

$$f_1(\mathbf{x}) = x_1 \tag{B.21}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})}\right] \tag{B.22}$$

where $x_1 \in [0,1]$ and $x_i \in [-5,5]\ \forall i = 2,..,10$.

ZDT6
This is a continuous box-constrained bi-objective problem, where:

$$g(\mathbf{x}) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25} \tag{B.23}$$

$$f_1(\mathbf{x}) = 1 - e^{-4x_1} \sin^6(6\pi x_1) \tag{B.24}$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (f_1(\mathbf{x})/g(\mathbf{x}))^2\right] \tag{B.25}$$

with $x_i \in [0,1]\ \forall i$.

B.3.2.2 DTLZ test suite

This is also a wide-spread test suite, which was conceived for multi-objective problems with scalable fitness and variables' dimensions. It consists of 7 problems: all the problems are continuous box-bounded problems with scalable fitness and variables' dimensions. In particular, when the authors constructed this test suite, they had in mind the following key points (Deb et al., 2005):

1. Test problems should be easy to construct.
2. Test problems should be scalable in both their objectives and decision variables' dimensions.
3. The Pareto-optimal front of the problems should be easy to comprehend, and its exact mathematical formulation should be known.
4. The difficulty of the problems should vary so that possible bottlenecks of the algorithms that are being tested can be found.

Each of the problems of the test suite can be formulated as follows (whenever is used, k is always defined as $k = n - M + 1$, with n number of variables and M number of objectives):

DTLZ1

This is a rather simple test problem, with M objectives and a linear Pareto-optimal front. The objective functions have the following mathematical formulation:

$$\begin{aligned} f_1(\mathbf{x}) &= \frac{1}{2} x_1 x_2 \ldots x_{M-1}(1 + g(\mathbf{x}_M)) \\ f_2(\mathbf{x}) &= \frac{1}{2} x_1 x_2 \ldots (1 - x_{M-1})(1 + g(\mathbf{x}_M)) \\ &\vdots \\ f_{M-1}(\mathbf{x}) &= \frac{1}{2} x_1 (1 - x_2)(1 + g(\mathbf{x}_M)) \\ f_M(\mathbf{x}) &= \frac{1}{2}(1 - x_1)(1 + g(\mathbf{x}_M)) \end{aligned} \tag{B.26}$$

where $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$. Also, the functional $g(\mathbf{x}_M)$ is formulated as follows:

$$g(\mathbf{x}_M) = 100\left[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right] \tag{B.27}$$

The Pareto-optimal solution corresponds to $\mathbf{x}_M = [0 \ldots 0]$.

DTLZ2

In this test problem, the objective functions have the following formulation:

$$\begin{aligned} f_1(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \ldots \cos(x_{M-2}\pi/2) \cos(x_{M-1}\pi/2) \\ f_2(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \ldots \cos(x_{M-2}\pi/2) \sin(x_{M-1}\pi/2) \\ f_3(\mathbf{x}_M) &= (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \ldots \cos(x_{M-2}\pi/2) \cos(x_{M-1}\pi/2) \\ &\vdots \\ f_{M-1}(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \sin(x_2\pi/2) \\ f_M(\mathbf{x}) &= (1 + g(\mathbf{x}_M)) \sin(x_1\pi/2) \end{aligned} \tag{B.28}$$

where $0 \leq x_i \leq 1$ for $i = 1, 2, \ldots, n$ and the function $g(\mathbf{x}_M)$ has the following definition:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{B.29}$$

The Pareto-optimal front is located at $\mathbf{x}_M = [0.5..0.5]$.

DTLZ3

In this case, the objective functions are formulated in the same way as Equation (B.28), but a different g function is used:

$$g(\mathbf{x}_M) = 100\left[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right] \tag{B.30}$$

The global Pareto-optimal front is located at $\mathbf{x}_M = [0.5..0.5]$

DTLZ4

This problem is the same as $DTLZ2$ but modified with a meta-variable mapping:

$$\begin{aligned} f_1(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(x_1^{\alpha}\pi/2)\cos(x_2^{\alpha}\pi/2)\dots\cos(x_{M-2}^{\alpha}\pi/2)\cos(x_{M-1}^{\alpha}\pi/2) \\ f_2(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(x_1^{\alpha}\pi/2)\cos(x_2^{\alpha}\pi/2)\dots\cos(x_{M-2}^{\alpha}\pi/2)\sin(x_{M-1}^{\alpha}\pi/2) \\ f_3(\mathbf{x}_M) &= (1+g(\mathbf{x}_M))\cos(x_1^{\alpha}\pi/2)\cos(x_2^{\alpha}\pi/2)\dots\cos(x_{M-2}^{\alpha}\pi/2)\cos(x_{M-1}^{\alpha}\pi/2) \\ &\cdot\cdot \\ &\cdot\cdot \\ &\cdot\cdot \\ f_{M-1}(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(x_1^{\alpha}\pi/2)\sin(x_2^{\alpha}\pi/2) \\ f_M(\mathbf{x}) &= (1+g(\mathbf{x}_M))\sin(x_1^{\alpha}\pi/2) \end{aligned} \tag{B.31}$$

where $0 \le x_i \le 1$ for $i = 1, 2, \dots, n$ and where:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{B.32}$$

The α parameter is set to $\alpha = 100$.

DTLZ5

In this case, a transformed variable (θ_i) is introduced and DTLZ2 is modified as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\dots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2) \\ f_2(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\dots\cos(\theta_{M-2}\pi/2)\sin(\theta_{M-1}\pi/2) \\ f_3(\mathbf{x}_M) &= (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\cos(\theta_2\pi/2)\dots\cos(\theta_{M-2}\pi/2)\cos(\theta_{M-1}\pi/2) \\ &\cdot\cdot \\ &\cdot\cdot \\ &\cdot\cdot \\ f_{M-1}(\mathbf{x}) &= (1+g(\mathbf{x}_M))\cos(\theta_1\pi/2)\sin(\theta_2\pi/2) \\ f_M(\mathbf{x}) &= (1+g(\mathbf{x}_M))\sin(\theta_1\pi/2) \end{aligned} \tag{B.33}$$

where $0 \le x_i \le 1$ for $i = 1, 2, \dots, n$ and where the θ_i variables are constructed as follows:

$$\theta_i = \frac{\pi}{4(1+g(r))}(1+2g(r)x_i) \qquad \text{for} i = 2, 3, \dots, M-1 \tag{B.34}$$

Also, the g function has the following expression:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{B.35}$$

DTLZ6

In this test problem, we use the same objective functions as those of DTLZ5 (expressed in Equation (B.33)). The difference, however, is the g function:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} x_i^{0.1} \tag{B.36}$$

DTLZ7

This problem has a disconnected set of Pareto-optimal regions, and it can be formulated as follows:

$$\begin{aligned} f_1(\mathbf{x}_1) &= x_1 \\ f_2(\mathbf{x}_2) &= x_2 \\ &\cdot\cdot \\ &\cdot\cdot \\ &\cdot\cdot \\ f_{M-1}(\mathbf{x}_{M-1}) &= x_{M-1} \\ f_M(\mathbf{x}) &= (1 + g(\mathbf{x}_M))h(f_1, f_2, \ldots, f_{M-1}, g) \end{aligned} \tag{B.37}$$

where $0 \leq x_i \leq 1$ for $i = 1, 2, \ldots, n$ and where the h function is constructed as follows:

$$h(f_1, f_2, \ldots, f_{M-1}, g) = M - \sum_{i=1}^{M-1}\left[\frac{f_i}{1+g}(1 + \sin(3\pi f_i))\right] \tag{B.38}$$

Also, the g function is defined as follows:

$$g(\mathbf{x}_M) = 1 + \frac{9}{\mathbf{x}_M}\sum_{x_i \in \mathbf{x}_M} x_i \tag{B.39}$$

B.3.2.3 WFG test suite

This test suite was developed after a thorough study on multi-objective test problems that have highlighted a few areas that were not covered by the standard test suites (i.e., ZDT and DTLZ) and that thus required attention (Huband et al., 2006). In their analysis, the authors point out that not only are many test problems poorly constructed, but a particular class of nonseparable multimodal problems is poorly represented. An objective function is multimodal when it has multiple local optima, whereas an objective function with only a single optimum is unimodal. A special class of multimodal objective functions is the deceptive one. For an objective function to be deceptive, as defined by Deb (1999), it is required that it has at least two optima: the real optimum and a deceptive one. The search space of deceptive problems must favor the finding of the deceptive optimum, thus making difficult for the algorithm to find the real one. Furthermore, a nonseparable problem has the characteristic that cannot be optimized by considering each parameter in turn, independently among each other. For multi-objective problems, this implies that if we have a separable problem, then the ideal points can be found by minimizing only one parameter at a time, thus making it easier to find at least some Pareto optimal points.

In particular, in Huband et al. (2006), the limitations of the ZDT and DTLZ test suites are discussed and analyzed. As the authors point out, the ZDT test suite seems to have the following issues:

1. Only the class of bi-objective problems is represented.
2. No fitness landscapes with flat regions are represented.
3. The only deceptive problem is ZDT5, but it is binary encoded.
4. No degenerate Pareto optimal front (i.e., a front that has a lower dimension w.r.t. the objective space of the problem) is present among the problems.
5. The class of nonseparable problems is not represented.

6. All the problems have either extremal or medial parameters (i.e., the optimum is placed either at the edge or at the middle part of the domain). This is considered bad practice because in box-bounded problems these optima are often found by chance, since the algorithm often tends to explore out of the box-bounded domain, and many algorithms have a bounce-back mechanism that pushes the variables either at the middle or at the edges of the domain when this happens.

While DTLZ solves some of these issues, it is still considered by the authors somewhat inadequate to well represent all the problems. In particular, they point out the following drawbacks:

1. None of the problems has flat regions in the fitness landscapes.
2. No deceptive problems are represented.
3. The class of nonseparable problems is not represented.

For overcoming these aspects, the authors thus introduce a new test suite, which consists of 9 different test problems. All the problems have the following format:

$$
\begin{aligned}
\text{Given } \mathbf{z} &= z_1, \ldots, z_k, z_{k+1}, \ldots, z_n \\
\text{Minimize } f_{m=1:M}(\mathbf{x}) &= Dx_M + S_m h_m(x_1, \ldots, x_{M-1}) \\
\text{where } \mathbf{x} &= \{x_1, \ldots, x_M\} = \\
&= \{max(t_M^p, A_1)(t_1^p - 0.5) + 0.5, \ldots, max(t_M^p, A_{M-1})(t_{M-1}^p - 0.5) + 0.5, t_M^p\} \\
\mathbf{t}^p &= \{t_1^p, \ldots, t_M^p\} \leftarrow \mathbf{t}^{p-1} \leftarrow \ldots \leftarrow \mathbf{t}^1 \leftarrow \mathbf{z}_{[0,1]} \\
\mathbf{z}_{[0,1]} &= \{z_{1,[0,1]}, \ldots, z_{n,[0,1]}\} \\
&= \{z_1/z_{1,max}, \ldots, z_n/z_{n,max}\}
\end{aligned}
$$

where M is the number of objectives, $\mathbf{x}$ is a set of M parameters, $\mathbf{z}$ is a set of decision variables of size $n >= M$, D>0 is a distance scaling constant, $A_{1:M-1}$ are degeneracy constants, and they are all within the [0,1] bounds: for each A_i=0, the Pareto optimal front is reduced by one. h_{1_M} are shape functions, $S_{1:M} > 0$ are scaling constants and $\mathbf{t}^{1:p}$ are transition vectors, where "←" is used for indicating that from one transition vector another one is constructed, via transformation functions. All the variables have the domain such that $z_i \in [0, z_{i,max}]$ $\forall$ $i = 1, \ldots, n$, where $z_{i,max} > 0$. In particular, the upper bound is chosen as: $z_{i,max} = 2(i+1)$. It is interesting also to notice how all the x_i will have domain [0,1].

Even though these test problems are more complex than the previous ones, their construction can be easily implemented by following the transformation and shape functions constructions shown in Tables B.22 and B.23. All the problems are shown in Table B.24.

It is important to point out that all the problems are scalable, with no extremal nor medial parameters, and they have dissimilar Pareto optimal trade-off magnitudes and dissimilar parameter domains, as well as known Pareto optimal sets. The properties of each specific problem is shown in Table B.25.

This test suite was not already available in PaGMO and was thus implemented during this thesis work. The suite was thoroughly tested before being applied (by checking that the correct objective functions are returned for known inputs, and by verifying the correct behavior of the problems). Having verified and validated the test suite, it was made available in PaGMO (both in C++ and Python): its description and use can be found in the official website of the software[6].

[6] `https://esa.github.io/pagmo2/docs/cpp/problems/wfg.html`, date of access: August, 2019.

Table B.22: WFG shape functions (Huband et al., 2006), in all the cases $x_i, ..., x_{M-1}$ are within the box-bounds [0, 1].

Linear

$$\begin{aligned} \text{linear}_1(x_1, \ldots, x_{M-1}) &= \prod_{i=1}^{M-1} x_i \\ \text{linear}_{m=2:M-1}(x_1, \ldots, x_{M-1}) &= \left(\prod_{i=1}^{M-m} x_i\right)(1 - x_{M-m+1}) \\ \text{linear}_M(x_1, \ldots, x_{M-1}) &= 1 - x_1 \end{aligned}$$

When $h_{m=1:M} = \text{linear}_m$, the Pareto optimal front is a linear hyperplane, where $\sum_{m=1}^{M} h_m = 1$.

Convex

$$\begin{aligned} \text{convex}_1(x_1, \ldots, x_{M-1}) &= \prod_{i=1}^{M-1}(1 - \cos(x_i \pi/2)) \\ \text{convex}_{m=2:M-1}(x_1, \ldots, x_{M-1}) &= \left(\prod_{i=1}^{M-m}(1 - \cos(x_i \pi/2))\right)(1 - \sin(x_{M-m+1}\pi/2)) \\ \text{convex}_M(x_1, \ldots, x_{M-1}) &= 1 - \sin(x_1 \pi/2) \end{aligned}$$

When $h_{m=1:M} = \text{convex}_m$, the Pareto optimal front is purely convex.

Concave

$$\begin{aligned} \text{concave}_1(x_1, \ldots, x_{M-1}) &= \prod_{i=1}^{M-1} \sin(x_i \pi/2) \\ \text{concave}_{m=2:M-1}(x_1, \ldots, x_{M-1}) &= \left(\prod_{i=1}^{M-m} \sin(x_i \pi/2)\right)\cos(x_{M-m+1}\pi/2) \\ \text{concave}_M(x_1, \ldots, x_{M-1}) &= \cos(x_1 \pi/2) \end{aligned}$$

When $h_{m=1:M} = \text{concave}_m$, the Pareto optimal front is purely concave, and a region of the hyper-sphere of radius one centred at the origin, where $\sum_{m=1}^{M} h_m^2 = 1$.

Mixed convex/concave ($\alpha > 0$, $A \in \{1, 2, \ldots\}$)

$$\text{mixed}_M(x_1, \ldots, x_{M-1}) = \left(1 - x_1 - \frac{\cos(2A\pi x_1 + \pi/2)}{2A\pi}\right)^{\alpha}$$

Causes the Pareto optimal front to contain both convex and concave segments, the number of which is controlled by A. The overall shape is controlled by α: when $\alpha > 1$ or when $\alpha < 1$, the overall shape is convex or concave respectively. When $\alpha = 1$, the overall shape is linear.

Disconnected ($\alpha, \beta > 0$, $A \in \{1, 2, \ldots\}$)

$$\text{disc}_M(x_1, \ldots, x_{M-1}) = 1 - (x_1)^{\alpha} \cos^2(A(x_1)^{\beta}\pi)$$

Causes the Pareto optimal front to have disconnected regions, the number of which is controlled by A. The overall shape is controlled by α (when $\alpha > 1$ or when $\alpha < 1$, the overall shape is concave or convex respectively, and when $\alpha = 1$, the overall shape is linear), and β influences the location of the disconnected regions (larger values push the location of disconnected regions towards larger values of x_1, and vice versa).

Table B.23: WFG transformation functions (Huband et al., 2006), where y_i have always domain [0,1].

Bias: Polynomial ($\alpha > 0,\ \alpha \neq 1$)

$$\text{b\_poly}(y, \alpha) = y^{\alpha}$$

When $\alpha > 1$ or when $\alpha < 1$, y is biased towards zero or towards one respectively.

Bias: Flat Region ($A, B, C \in [0, 1],\ B < C,\ B = 0 \Rightarrow A = 0 \wedge C \neq 1,\ C = 1 \Rightarrow A = 1 \wedge B \neq 0$)

$$\text{b\_flat}(y, A, B, C) = A + \min(0, \lfloor y - B \rfloor)\frac{A(B-y)}{B} - \min(0, \lfloor C - y \rfloor)\frac{(1-A)(y-C)}{1-C}$$

Values of y between B and C (the area of the flat region) are all mapped to the value A.

Bias: Parameter Dependent ($A \in (0, 1),\ 0 < B < C$)

$$\text{b\_param}(y, \mathbf{y}', A, B, C) = y^{B+(C-B)v(u(\mathbf{y}'))}$$

$$v(u(\mathbf{y}')) = A - (1 - 2u(\mathbf{y}'))\left|\lfloor 0.5 - u(\mathbf{y}') \rfloor + A\right|$$

A, B, C, and the secondary parameter vector $\mathbf{y}'$ together determine the degree to which y is biased by being raised to an associated power: values of $u(\mathbf{y}') \in [0, 0.5]$ are mapped linearly onto $[B, B + (C - B)A]$, and values of $u(\mathbf{y}') \in [0.5, 1]$ are mapped linearly onto $[B + (C - B)A, C]$.

Shift: Linear ($A \in (0, 1)$)

$$\text{s\_linear}(y, A) = \frac{|y - A|}{|\lfloor A - y \rfloor + A|}$$

A is the value for which y is mapped to zero.

Shift: Deceptive ($A \in (0, 1),\ 0 < B \ll 1,\ 0 < C \ll 1,\ A - B > 0,\ A + B < 1$)

$$\text{s\_decept}(y, A, B, C) = 1 + (|y - A| - B)\times \left(\frac{\lfloor y - A + B \rfloor (1 - C + \frac{A-B}{B})}{A-B} + \frac{\lfloor A + B - y \rfloor (1 - C + \frac{1-A-B}{B})}{1-A-B} + \frac{1}{B}\right)$$

A is the value at which y is mapped to zero, and the global minimum of the transformation. B is the "aperture" size of the well/basin leading to the global minimum at A, and C is the value of the deceptive minima (there are always two deceptive minima).

Shift: Multi-modal ($A \in \{1, 2, \ldots\},\ B \geq 0,\ (4A + 2)\pi \geq 4B,\ C \in (0, 1)$)

$$\text{s\_multi}(y, A, B, C) = \frac{1 + \cos\left[(4A+2)\pi\left(0.5 - \frac{|y-C|}{2(\lfloor C - y \rfloor + C)}\right)\right] + 4B\left(\frac{|y-C|}{2(\lfloor C-y \rfloor + C)}\right)^2}{B+2}$$

A controls the number of minima, B controls the magnitude of the "hill sizes" of the multi-modality, and C is the value for which y is mapped to zero. When $B = 0$, $2A + 1$ values of y (one at C) are mapped to zero, and when $B \neq 0$, there are $2A$ local minima, and one global minimum at C. Larger values of A and smaller values of B create more difficult problems.

Reduction: Weighted Sum ($|\mathbf{w}| = |\mathbf{y}|,\ w_1, \ldots, w_{|\mathbf{y}|} > 0$)

$$\text{r\_sum}(\mathbf{y}, \mathbf{w}) = \left(\sum_{i=1}^{|\mathbf{y}|} w_i y_i\right) / \sum_{i=1}^{|\mathbf{y}|} w_i$$

By varying the constants of the weight vector $\mathbf{w}$, EAs can be forced to treat parameters differently.

Reduction: Non-separable ($A \in \{1, \ldots, |\mathbf{y}|\},\ |\mathbf{y}| \bmod A = 0$)

$$\text{r\_nonsep}(\mathbf{y}, A) = \frac{\sum_{j=1}^{|\mathbf{y}|}\left(y_j + \sum_{k=0}^{A-2}\left|y_j - y_{1+(j+k) \bmod |\mathbf{y}|}\right|\right)}{\frac{|\mathbf{y}|}{A}\lceil A/2 \rceil (1 + 2A - 2\lceil A/2 \rceil)}$$

A controls the degree of non-separability (noting that $\text{r\_nonsep}(\mathbf{y}, 1) = \text{r\_sum}(\mathbf{y}, \{1, \ldots, 1\})$).

Table B.24: WFG test suite construction (Huband et al., 2006) (in the table, we let $\mathbf{y} = \mathbf{t}^{i-1}$ and for $\mathbf{t}^1$, we let $\mathbf{y} = \mathbf{z}_{[0,1]} = \{z_1/2, \ldots, z_n/(2n)\}$).

Problem	Type	Setting
All	Constants	$S_{m=1:M} = 2m$ $D = 1$ $A_1 = 1$ $A_{2:M-1} = \begin{cases} 0, & \text{for WFG3} \\ 1, & \text{otherwise} \end{cases}$ The settings for $S_{1:M}$ ensures the Pareto optimal fronts have dissimilar tradeoff magnitudes, and the settings for $A_{1:M-1}$ ensures the Pareto optimal fronts are not degenerate, except in the case of WFG3, which has a one dimensional Pareto optimal front.
All	Domains	$z_{i=1:n,\max} = 2i$ The working parameters have domains of dissimilar magnitude.
WFG1	Shape	$h_{m=1:M-1} = \text{convex}_m$ $h_M = \text{mixed}_M$ (with $\alpha = 1$ and $A = 5$)
	$\mathbf{t}^1$	$t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = \text{s\_linear}(y_i, 0.35)$
	$\mathbf{t}^2$	$t^2_{i=1:k} = y_i$ $t^2_{i=k+1:n} = \text{b\_flat}(y_i, 0.8, 0.75, 0.85)$
	$\mathbf{t}^3$	$t^3_{i=1:n} = \text{b\_poly}(y_i, 0.02)$
	$\mathbf{t}^4$	$t^4_{i=1:M-1} = \text{r\_sum}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \{2((i-1)k/(M-1)+1), \ldots, 2ik/(M-1)\})$ $t^4_M = \text{r\_sum}(\{y_{k+1}, \ldots, y_n\}, \{2(k+1), \ldots, 2n\})$
WFG2	Shape	$h_{m=1:M-1} = \text{convex}_m$ $h_M = \text{disc}_M$ (with $\alpha = \beta = 1$ and $A = 5$)
	$\mathbf{t}^1$	As $\mathbf{t}^1$ from WFG1. (Linear shift.)
	$\mathbf{t}^2$	$t^2_{i=1:k} = y_i$ $t^2_{i=k+1:k+l/2} = \text{r\_nonsep}(\{y_{k+2(i-k)-1}, y_{k+2(i-k)}\}, 2)$
	$\mathbf{t}^3$	$t^3_{i=1:M-1} = \text{r\_sum}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \{1, \ldots, 1\})$ $t^3_M = \text{r\_sum}(\{y_{k+1}, \ldots, y_{k+l/2}\}, \{1, \ldots, 1\})$
WFG3	Shape	$h_{m=1:M} = \text{linear}_m$ (degenerate)
	$\mathbf{t}^{1:3}$	As $\mathbf{t}^{1:3}$ from WFG2. (Linear shift, non-separable reduction, and weighted sum reduction.)
WFG4	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	$t^1_{i=1:n} = \text{s\_multi}(y_i, 30, 10, 0.35)$
	$\mathbf{t}^2$	$t^2_{i=1:M-1} = \text{r\_sum}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \{1, \ldots, 1\})$ $t^2_M = \text{r\_sum}(\{y_{k+1}, \ldots, y_n\}, \{1, \ldots, 1\})$
WFG5	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	$t^1_{i=1:n} = \text{s\_decept}(y_i, 0.35, 0.001, 0.05)$
	$\mathbf{t}^2$	As $\mathbf{t}^2$ from WFG4. (Weighted sum reduction.)
WFG6	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	As $\mathbf{t}^1$ from WFG1. (Linear shift.)
	$\mathbf{t}^2$	$t^2_{i=1:M-1} = \text{r\_nonsep}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, k/(M-1))$ $t^2_M = \text{r\_nonsep}(\{y_{k+1}, \ldots, y_n\}, l)$
WFG7	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	$t^1_{i=1:k} = \text{b\_param}(y_i, \text{r\_sum}(\{y_{i+1}, \ldots, y_n\}, \{1, \ldots, 1\}), \frac{0.98}{49.98}, 0.02, 50)$ $t^1_{i=k+1:n} = y_i$
	$\mathbf{t}^2$	As $\mathbf{t}^1$ from WFG1. (Linear shift.)
	$\mathbf{t}^3$	As $\mathbf{t}^2$ from WFG4. (Weighted sum reduction.)
WFG8	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	$t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = \text{b\_param}(y_i, \text{r\_sum}(\{y_1, \ldots, y_{i-1}\}, \{1, \ldots, 1\}), \frac{0.98}{49.98}, 0.02, 50)$
	$\mathbf{t}^2$	As $\mathbf{t}^1$ from WFG1. (Linear shift.)
	$\mathbf{t}^3$	As $\mathbf{t}^2$ from WFG4. (Weighted sum reduction.)
WFG9	Shape	$h_{m=1:M} = \text{concave}_m$
	$\mathbf{t}^1$	$t^1_{i=1:n-1} = \text{b\_param}(y_i, \text{r\_sum}(\{y_{i+1}, \ldots, y_n\}, \{1, \ldots, 1\}), \frac{0.98}{49.98}, 0.02, 50)$ $t^1_n = y_n$
	$\mathbf{t}^2$	$t^2_{i=1:k} = \text{s\_decept}(y_i, 0.35, 0.001, 0.05)$ $t^2_{i=k+1:n} = \text{s\_multi}(y_i, 30, 95, 0.35)$
	$\mathbf{t}^3$	As $\mathbf{t}^2$ from WFG6. (Non-separable reduction.)

Table B.25: WFG test problems properties (Huband et al., 2006).

Problem	Obj.	Separability	Modality	Bias	Geometry
WFG1	$f_{1:M}$	S	U	polynomial,flat	convex, mixed
WFG2	$f_{1:M-1}$	NS	U	–	convex, disconnected
	f_M	NS	M		
WFG3	$f_{1:M}$	NS	U	–	linear, degenerate
WFG4	$f_{1:M}$	S	M	–	concave
WFG5	$f_{1:M}$	S	D	–	concave
WFG6	$f_{1:M}$	NS	U	–	concave
WFG7	$f_{1:M}$	S	U	parameter dependent	concave
WFG8	$f_{1:M}$	NS	U	parameter dependent	concave
WFG9	$f_{1:M}$	NS	M,D	parameter dependent	concave

Appendix C

Global Optimization Methods

In this appendix, we will explain the working principle of each of the single and multi-objective algorithms used in this research. The discussion will thus cover all the global optimizers used, except for the SO and MO ant colony optimizer and the nondominated sorting particle swarm optimizer, which were developed and defined within this thesis study and are thus separately discussed in Section 5. This appendix is organized as follows: first, in Section C.1, the SO methods are discussed and then, in Section C.2, the MO methods are presented.

C.1. Single-Objective Methods

In this section, we will deal with the single-objective global optimizers. In particular, we will treat six different SO optimizers (i.e., DE, SADE, DE1220, SGA, PSO, and ABC) and a meta-algorithm used for turning single-objective unconstrained problems into constrained ones (so that the aforementioned algorithms can be applied for all the problems that pertain to these two classes).

C.1.1. Standard Differential Evolution (DE)

The Differential Evolution (DE) algorithm was first designed in Storn and Price (1997) and represents a direct search method that takes advantage of NP n-dimensional decision vectors (also known as individuals):

$$\mathbf{x}_{i,G},\ i = 1, 2, \ldots, NP \tag{C.1}$$

where G refers to the generation currently being evolved. Indeed, this algorithm is also of the evolutionary kind: it thus evolves a set of individuals (called population) throughout a certain number of user-defined generations.

Similarly to many other algorithms of this kind, the initial population is initialized randomly within the box-bounds in which the decision vectors are defined, using a uniformly random number generator and trying to cover the search space as much as possible.

The algorithm can be described through three simple passages:

1. Mutation: in this process, DE produces a new individual by adding the weighted difference between two other individuals of the same population to a third one. Mathematically, for each individual $x_{i,G},\ i = 1, 2, \ldots, NP$, a mutant vector is generated by using the following formula:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F \cdot \left(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}\right) \tag{C.2}$$

   where r_1, r_2 and $r_3 \in \{1, 2, \ldots, NP\}$ are three randomly generated and mutually different vectors, and $F \in [0, 1]$ is a user-defined parameter called weight coefficient. A two-dimensional example of this process is shown in Figure C.1.

2. Crossover: having mutated the vector, its mutated parameters are then mixed together with those of another chosen vector (called target vector), to produce a trial vector. In particular, this trial vector (i.e., $u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{ni,G+1})$) is constituted, in which:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (randb(j) \leq CR) \text{ or } j = rnbr(i) \\ x_{ji,G} & \text{if } (randb(j) > CR) \text{ and } j \neq rnbr(i) \end{cases} \tag{C.3}$$

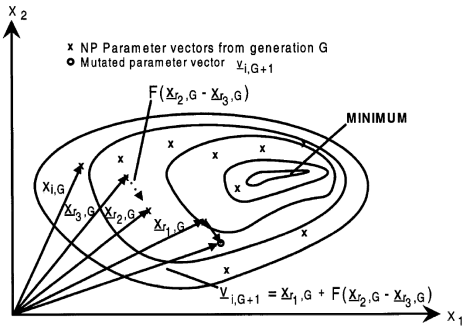


Figure C.1: Graphical representation of the mutation process for the DE algorithm (Storn and Price, 1997)

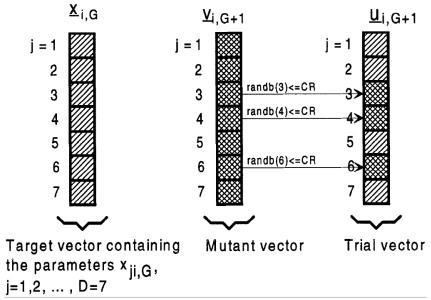


Figure C.2: Graphical representation of the crossover process for the DE algorithm (Storn and Price, 1997)

where $j = 1, 2, \ldots, n$ and CR is a user-defined parameter in the [0, 1] range, called crossover constant. Also, $randb(j)$ is the jth evaluation of a uniformly distributed number in [0, 1] and $rnbr(i)$ is a randomly chosen index ($\in 1, 2, \ldots, n$ which makes sure that $\mathbf{u}_{i,G+1}$ obtains at least one parameter from $\mathbf{v}_{i,G+1}$. An example of the crossover process for a 7-dimensional decision vector is shown in Figure C.2 .

3. Selection: if the trial vector brings an improvement in terms of the cost function (i.e., if it yields a lower cost function value than the target vector), then it substitutes the target vector in the future generation. Hence, it is only checked whether $\mathbf{u}_{i,G+1}$ leads to a smaller cost function w.r.t. $\mathbf{x}_{i,G}$. If this is the case, $\mathbf{x}_{i,G+1}$ is set to $u_{i,G+1}$, otherwise, the old decision vector $x_{i,G}$ is used again in the following generation.

In the above scheme, we have seen some parameters that have some freedom of choice. Besides the F and CR parameters, there are the following aspects to be chosen:

1. A vector to be mutated (i.e., it could be a random vector or the best vector).
2. the number of different vectors to be used.
3. The crossover scheme.

Based on the three aspects, 10 different variants of DE have been developed. All of these variants make use of two possible crossover schemes: either binomial crossover or exponential crossover. A comparison of these two crossover techniques and their detailed implementation is discussed in Zaharie (2007).

To summarize, the DE algorithm, besides the number of individuals and generations, has three tweakable parameters: the two constants F and CR, and the DE variant.

C.1.2. Self-Adaptive Differential Evolution (SADE)

Differential evolution has become a popular algorithm throughout the years. For this reason, many authors have decided to introduce changes to the basic algorithm and produce their

own variants. An interesting one is shown in Brest et al. (2006). In this paper, the authors propose a self-adaptive DE scheme: meaning, a DE algorithm that self-adapts its parameter, thus preventing the user from the burden to tweak them. In particular, they propose the self-adaptation of two of the three DE input parameters mentioned in Section C.1.1: the two constants (i.e., CR and F).

As explained by the authors, better values of these two input parameters lead to better decision vectors' choices that, in turn, allow us to find better fitness values. For this reason, they propose a self-adaptive scheme for F and CR, which are adapted at every G generation, that works as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G}, & \text{otherwise} \end{cases} \tag{C.4}$$

$$CR_{i,G+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_{i,G}, & \text{otherwise} \end{cases} \tag{C.5}$$

where $rand_j$, for $j = 1, 2, 3, 4$, are uniformly distributed random values in the [0, 1] range. τ_1, τ_2, F_l and F_u are user-defined parameters. This might suggest that the user-defined parameters have increased from two to four: this is however wrong since the authors have experimentally set fixed values for all these constants: $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$ and $F_u = 0.9$. In this way, $F \in [0.1, 1.0]$ and $CR \in [0, 1]$. The reason why the authors decide to pass from $F \in [0, 1]$, as defined in the original DE algorithm, to $F \in [0.1, 1]$ is that $F = 0$ corresponds to a new trial vector generated using crossover but no mutation.

Using this proposed scheme, the user has thus only to choose the DE variant: this reduces the tweakable parameters to only one (besides the number of individuals and generations).

C.1.3. Differential Evolution Variant (DE1220)

Similarly to the one explained in Section C.1.2, this is another self-adaptive DE scheme, proposed by the owners of the PaGMO optimization toolbox [1]. This algorithm is identical to the one shown in Section C.1.2, with the innovation that a self-adaptation for the mutation variant is also added. In this way, the users are only left to choose the number of generations and the population size, and they do not have to worry about any other input parameters.

The mutation variant is chosen according to the following scheme:

$$V_i = \begin{cases} random & \text{if } r_i < \tau \\ V_{i-1} & \text{otherwise} \end{cases} \tag{C.6}$$

where i is the currently being evolved generation, τ is set to be equal to 0.1, $random$ selects a random mutation variant (in the range between 1 and 10, since 10 different variants are possible) and r_i is a uniformly distributed random number in the [0, 1] range.

Due to its simplicity, and due to the fact that no input parameters of the algorithm have to be chosen by the user, we have decided to also include this algorithm in our trade-off.

C.1.4. Simple Genetic Algorithm (SGA)

This algorithm was first introduced in Holland et al. (1992), and then thoroughly discussed and analyzed in Oliveto et al. (2007). This algorithm is part of the class of genetic algorithms (GA), which are metaheuristic algorithms that belong to the broad class of evolutionary algorithms and that are inspired by the process of natural selection. This simple genetic algorithm (SGA) makes use of three processes already discussed in Section C.1.1: selection, crossover, and mutation. The algorithm first selects N individuals from the population, it then creates a new population of individuals from the chosen one (using crossover), and it finally mutates this new population to obtain another set of individuals, which will hopefully improve the fitness values. Several different genetic algorithms distinguish each other only in the type of selection, crossover and mutation methods used. In our research, we have decided to employ a rather standard genetic algorithm scheme, which makes use of the following three types:

[1] https://esa.github.io/pagmo2/docs/cpp/algorithms/de1220.html

1. Tournament selection: each offspring (i.e., new individual) is selected based on the minimal fitness of a random group of two individuals.
2. Exponential crossover: this consists in selecting a random point in the parent chromosome and inserting it in the following genes with a certain probability CR (which is set to be 0.9), until the algorithm stops.
3. Polynomial mutation: this is a very popular mutation scheme introduced in Deb and Agrawal (1999).

Besides the aforementioned three procedures, also a reinsertion scheme is used. This scheme is called pure elitism and consists in placing, at every G generation, all the new and old individuals in the same pool and only selecting the best (in terms of minimal fitness values) n individuals to be passed to the next generation (where n is the population size).

Having discussed the simple genetic algorithm variant employed in this study, as for the other optimizers, we are left to choose the population size and generations' number for making the algorithm work for a single-objective unconstrained problem.

C.1.5. Particle Swarm Optimization (PSO)

This evolutionary algorithm was first introduced in Eberhart and Kennedy (1995) and then many modifications and variants have been developed throughout the years. The version that we propose in this study is the standard one and is inspired by the foraging behavior of swarms.

In this algorithm, we have a population of NP individuals to be evolved for G generations. The first population is initialized randomly within the box-bounds of each decision vector's component. For the following generations, each decision vector $\mathbf{x}_i$ is updated using the information of where it achieved the best performance in previous generations (i.e., $\mathbf{x}_i^l$) and using the best decision vector in a certain neighborhood (i.e., $\mathbf{x}^g$). This information is used for defining a velocity vector, which will then be used for updating the decision vector:

$$\mathbf{v}_{i+1} = \omega\left(\mathbf{v}_i + \eta_1 \mathbf{r}_1 \cdot (\mathbf{x}_i - \mathbf{x}_i^l) + \eta_2 \mathbf{r}_2 \cdot (\mathbf{x}_i - \mathbf{x}^g)\right) \tag{C.7}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i \tag{C.8}$$

where $i = 1, 2, \ldots, G$ refers to the current generation being evolved, and ω, η_1, η_2 are three user-defined parameters. The two random vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are uniformly distributed random numbers in the [0, 1] range.

To summarize, in this algorithm we can tweak and tune, besides the number of individuals and generations, three different parameters (i.e., ω, η_1, η_2). Additionally, there is also the possibility to limit the particles' velocity within a certain user-defined maximum value.

C.1.6. Artificial Bee Colony (ABC)

This algorithm was first introduced in Karaboga and Basturk (2007) and is based on the biological mechanism throughout the bees seek their food source. In particular, for describing the algorithm, we first need to classify the bees in three different groups: onlookers, employed bees and scouts. The first group is constituted by the bees that are waiting for making a decision before choosing a food source. The second group is made by the bees that are going to the food source already previously visited. The third group is made by the bees that are performing a random search for food sources.

In this algorithm, the initialized population is split into employed artificial bees and onlookers. The number of employed bees is equal to the number of food sources found. Whenever a food source is finished, the employed bee becomes a scout. What drives the choice is the nectar amount: the better the source of food, the higher the nectar. Hence, after the first population is randomly initialized, a nectar ranking among all the food sources is established and new bees are sent out as scouts for seeking new food sources, using the information retrieved by the employed bees. Thus, we can summarize the algorithms' steps in three passages (after the population is initialized randomly):

1. The employed bees are sent out to the food source and the nectar is measured.
2. The food sources are then selected by the onlookers, once the information retrieved by the employed bees are processed.
3. The scout bees are finally determined and they are sent to explore new possible sources of food.

These iterations are repeated all over for as many generations as requested. The information of previously found sources of food is stored for the purpose of finding new candidate sources of foods after the information is shared in the hive with the onlookers. Of course, the onlookers will prefer the food areas with more nectar. When the nectar amount of a food source increases, the probability of choosing that food source also increases.

In the algorithm, the position of a food source thus represents a feasible solution to the problem, and the nectar level is representative of the fitness of the associated solution.

Provided that the nectar amount of a new bee is higher than the previous one, then the bee forgets that nectar amount and holds in memory the new one. The onlooker bees select a food source based on the information retrieved by the employed bees and according to the probability value associated with it. This is simply computed by doing:

$$p_i = \frac{fit_i}{\sum_{n=1}^{N} fit_n} \tag{C.9}$$

where N is the population size and fit_i is the fitness value of a solution i, as evaluated by the employed bee.

The critical part of the algorithm is the production of new food sources around the found one. This is done, for each component of each individual decision vector, using the following expression:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{C.10}$$

where k ranges from 1 to the number of employed bees, j goes from 1 to the dimension of the search space, i goes from 1 to the number of individuals. The indices k and j are randomly chosen, but k is always kept different from i. ϕ_{ij} represents a random number in the [-1, 1] range.

The more x_{ij} and x_{kj} difference decreases, the more the perturbation on x_{ij} decreases too, thus producing a closer individual from the previous one. This is done for enhancing convergence towards the optimal solution.

Once the new candidate positions v_{ij} are produced, their fitness values are compared with those of x_{ij} and if the new source of food results to possess a higher or equal value of nectar, the old source is replaced with the new one.

If a solution is not updated in terms of nectar for a certain number of generations, which is called limit, then that solution is dropped.

It is clear, from our description, that besides the population and generation sizes, the artificial bee colony algorithm has only one control parameter: the limit parameter value.

C.1.7. Self-Adaptive Constraints Handling Meta-Algorithm

This meta-algorithm represents a constraint handling algorithm that permits the usage of any single-objective unconstrained optimization algorithm on single-objective constrained problems. This is very useful in our case, since most of our optimization problems (i.e., the heliocentric phase and the geocentric and heliocentric phase) have constraints, but all the single-objective algorithms that we would like to apply (except for ACOmi) cannot handle them, and they thus could not be used for them, if this meta-algorithm was not applied.

This strategy was first introduced in Wright and Farmani (2001), however, it was then refined in Farmani and Wright (2003), were a more dynamic algorithm was implemented for refining the previous method. This method basically changes a single-objective constrained problem into an unconstrained one, by turning its constraint functions into penalty functions to be added to the fitness. In particular, a two-stages penalty method is employed. This

approach has two main advantages: first of all, it does not require any tweaking or tuning of parameters by the user. Secondly, it has also been demonstrated to find the global optimal (feasible) solution, starting from a population of unfeasible individuals.

For describing this method, we first have to introduce some definitions. We have a certain minimization problem, where we would like to minimize an objective function: $f(\mathbf{x}) = f(x_1, \ldots, x_n)$, whose value depends on the decision vectors' components. The problem is subjected to q inequality constraints:

$$g_j(\mathbf{x}) \leq 0, \; j = 1, \ldots, q \tag{C.11}$$

and $(m - q + 1)$ equality constraints:

$$h_j(\mathbf{x}) = 0, \; (j = q + 1, \ldots, m) \tag{C.12}$$

The meta-algorithm is constituted in three passages:

1. Each individual receives an infeasibility value. This infeasibility value accounts for both the number of active constraints on each individual and the extent to which each constraint is violated. For doing this, the sum of the norm of the constraint values is computed (only for the violated constraints). This is thus done by first setting the constraints to zero if they result feasible. Also, a small user-defined tolerance δ can be set: this means that solutions with equality constraints' values below that small threshold still result as zero. Mathematically, this means that we can redefine the constraints as follows:

$$c_j(\mathbf{x}) = \begin{cases} \max(0, g_j(\mathbf{x})), & \text{if } 1 \leq j \leq q \\ \max(0, (|h_j(\mathbf{x}| - \delta)), & \text{if } q + 1 \leq j \leq m \end{cases} \tag{C.13}$$

   where m is the number of inequality and equality constraints. We can then compute the solution's infeasiblity (i.e., $i(\mathbf{x})$), as follows:

$$i(\mathbf{x}) = \frac{\sum_{j=1}^{m} \frac{c_j(\mathbf{x})}{c_{max,j}}}{m} \tag{C.14}$$

   where the $c_{max,j}$ value represents the maximum value of the constraint violation in the population being evolved.

2. The worst and best solutions of the search space are identified. In particular, we define three different individuals:

   - $\mathbf{x}_b$ best individual;
   - $\mathbf{x}_w$ worst of the infeasible solutions;
   - $\mathbf{x}_h$ solution with the highest objective function value in the current population.

   In the case that the current population has at least one feasible decision vector, the best individual will thus be the one with the lowest objective function value. However, if this is not the case, then it will be the solution with the lowest infeasibility value (neglecting the objective function values). Concerning the worst individual, two situations may happen: the first is that one or more infeasible solutions in the population have objective function values that are lower than the one of the best individual. In this case, the worst is taken as the individual with the highest infeasibility value, and the objective function value lower than the best solution. The second case is the one in which all the infeasible individuals in the population have objective function values higher than the best individual. In this case, the worst is taken as the one with the highest infeasibility value.

   It must be noted that in both the aforementioned cases, if two or more individuals have the same infeasibility value, then the objective function will drive the choice (i.e., the individual with the highest objective function value among them will be the worst).

3. The infeasible solutions are finally penalized: this is done before converting the objective function value in fitness. For describing this procedure, it is first important to introduce some definitions. We will call $F(\mathbf{x})$ the fitness, $\dot{f}(\mathbf{x})$ the penalized objective function value after the first penalty and $\ddot{f}(\mathbf{x})$ the penalized objective function value after the second penalty. The fitness is just the objective function value after subtracting the second penalized objective function after the second penalty. The infeasible solutions are first penalized using the infeasibility values of the worst and best solutions (i.e., $i(\mathbf{x}_w)$ and $i(\mathbf{x}_b)$, respectively), together with the highest objective function value in the current population (i.e., $f(\mathbf{x}_h)$). The first stage penalty is only applied if an infeasible solution exist with an objective function value lower than the current best solution (i.e., if $\mathbf{x}$ ∃ such that $f(\mathbf{x}) < f(\mathbf{x}_b)$ and $i(\mathbf{x}) > 0$). If such case is verified, the first stage penalty is introduced to make sure that the objective function values of the infeasible solutions is increased so that the worst infeasible solution will turn out to have an objective function value higher or equal that of the best solution. Mathematically, this is achieved by defining the following penalized objective function:

$$\tilde{i}(\mathbf{x}) = \frac{i(\mathbf{x}) - i(\mathbf{x}_b)}{i(\mathbf{x}_w) - i(\mathbf{x}_b)} \tag{C.15}$$

$$\dot{f}(\mathbf{x}) = f(\mathbf{x}) + \tilde{i}(\mathbf{x})\big(f(\mathbf{x}_b) - f(\mathbf{x}_w)\big) \tag{C.16}$$

As it can be noted, if the penalty is not applied, then $\dot{f}(\mathbf{x}) = f(\mathbf{x})$.

The second penalty is useful for increasing the objective function values so that the penalized objective function value of the worst individual is equal to that of the worst objective individual. This is achieved by defining the following penalized objective function:

$$\ddot{f}(\mathbf{x}) = \dot{f}(\mathbf{x}) + \gamma\left|\dot{f}(\mathbf{x})\right|\left(\frac{e^{2\tilde{i}(\mathbf{x})} - 1}{e^2 - 1}\right) \tag{C.17}$$

$$\gamma = \begin{cases} \dfrac{f(\mathbf{x}_h) - f(\mathbf{x}_b)}{f(\mathbf{x}_b)}, & \text{if } (f(\mathbf{x}_w) \leq f(\mathbf{x}_b)) \\ 0.0, & \text{if } (f(\mathbf{x}_w) = f(\mathbf{x}_h)) \\ \dfrac{f(\mathbf{x}_h) - f(\mathbf{x}_w)}{f(\mathbf{x}_w)}, & \text{if } (f(\mathbf{x}_w) > f(\mathbf{x}_b)) \end{cases} \tag{C.18}$$

The γ parameter makes sure that the penalization is fair depending on the objective function values of the worst, best and highest individuals.

This method is thus capable, through a dynamical and self-tuned two-stages penalty, to transform any single-objective constrained problem into an unconstrained one, thus allowing many algorithms to work in a wider range of problems. Also, if an optimization algorithm is able to handle both constrained and unconstrained optimization problems (which is the case for ACOmi, as shown in Section 5.3.1), we can compare the performances of the algorithm alone, with those of the algorithm coupled with this self-adaptive constraints handling technique.

C.2. Multi-Objective Methods

Multi-objective (MO) optimization is a crucial area of mathematics involved in the minimization of problems that have more than one objective function value to be optimized concurrently. As we have already pointed out, in this case, for a nontrivial problem, no single best solution can be found, but a set of multiple optimal solutions exists (called Pareto front). The methodology and implementation of MO algorithms are thus completely different from single-objective ones. In this section, we will discuss the working principle and pseudo-code of two popular multi-objective optimizers: the nondominated sorting genetic algorithm (NSGA-II) and the multi-objective evolutionary algorithm with decomposition (MOEA/D). These are well known and tested MO algorithms.

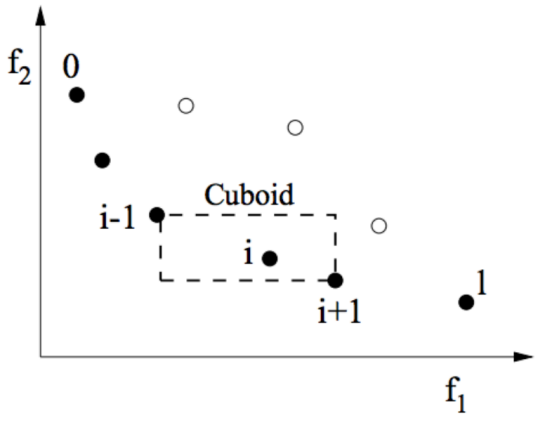


Figure C.3: Crowding distance computation (points marked with the same filled circles belong to the same nondominated front) (Deb et al., 2002).

C.2.1. Nondominated Sorting Genetic Algorithm (NSGA-II)

This algorithm was first introduced in Deb et al. (2002), and is one of the most popular and used MO optimization algorithms. This is a genetic algorithm and thus takes advantage of a certain type of crossover and mutation for then evolving the population to the next generation using the nondominated sorting and crowding distance comparison concepts. These two aspects are thus fundamental for understanding how the algorithm works. The algorithm description can be presented as follows:

1. Nondominated sorting: the objective is to sort the population into different nondomination levels (where the concept of domination was already introduced in Section 5.1.2 and mathematically described in Equation (5.7)). The procedure for doing this is quite simple, whereas its implementation is slightly more sophisticated (for reducing the computational time required for sorting the individuals) and is thoroughly described in the original paper (Deb et al., 2002). The Pareto-optimal front of the current population is first identified, and each solution belonging to this front is assigned a value of zero. Afterwards, these individuals are removed from the population, and a second Pareto-optimal front is found. To this second set of Pareto-optimal individuals, a value of one is assigned (thus referring to the fact that they are worse in terms of Pareto-optimality w.r.t. those who have a rank of zero, but better w.r.t. those with a rank of two). This procedure is repeated until all the individuals are sorted and a rank is assigned to each of them. This procedure will, in general, allow us to store both the number of solutions, which dominate a certain solution p (i.e., the so-called domination count n_p) and the set of solutions that the solution p dominates (i.e., S_p). The domination count of the individuals belonging to the first nondominated front will thus be zero. It is thus clear that the nondomination count can range from 0 to a maximum of $NP - 1$ (where NP is the number of individuals in the current population).

2. Crowding distance comparison: when evaluating the performances of a certain population in a MO framework, besides convergence towards a Pareto-optimal set, it would also be desired to maintain a certain spread of solutions within the fronts. For achieving this, the crowded-comparison operator is introduced in NSGA-II. After having identified only the solutions belonging to a certain nondominated Pareto front, we take one individual i within that front and we compute the average distance of that solution w.r.t. the two nearest solutions belonging to the same front: in this way we get an estimate of the average side length of the cuboid (which is shown with a dashed box in Figure C.3).

   It is thus clear that for storing the crowding distance of each individual, we only need to first divide the population in nondominated fronts, and then sort the individuals within each front according to their objective function values, in ascending order. Also, the boundary solutions (i.e., solutions with smallest and largest objective function values)

are assigned an infinite distance value. All the other solutions are assigned a distance value that is computed through the absolute normalized difference in the objective function values of two adjacent solutions (which is basically identical to the concept of Euclidean distance). In this way, a solution with a smaller crowding distance value will result to be more crowded with respect to the others. We can thus introduce the so-called crowded-comparison operator (i.e., $<_n$) that leads the selection process of NSGA-II, taking into account both the nondomination rank (i_{rank}) and crowding distance (i_{distance}) value of every individual i in the population. In particular, an individual i is preferred than an individual j if the following is verified:

$$i <_n j \text{ if } (i_{\text{rank}} < j_{\text{rank}})$$
$$\text{or } ((i_{\text{rank}} = j_{\text{rank}}) \text{ and } (i_{\text{distance}} > j_{\text{distance}}))$$

This means that between two solutions, we prefer that with a lower domination rank. If the two solutions have the same domination rank, then the solution with the higher crowding distance value (meaning that is located in a less crowded region) is preferred.

To summarize, after the first random population is created, the algorithm proceeds by first computing the nondomination levels and crowding distances of each individual. These values are used for sorting the population according to the crowding-comparison criterion. Then, binary tournament selection, crossover and mutation operators are used for generating the new population (i.e., offspring). At this point, at every generation, both parents and offspring are compared and sorted according to the crowding-distance comparison, and only the best NP individuals (where NP is the population size) are used for generating the future offspring. In this way, elitism is also ensured.

C.2.2. MO Evolutionary Algorithm with Decomposition (MOEA/D)

This popular multi-objective algorithm, of the differential evolution class, was first introduced in Zhang and Li (2007b), and represents one of the most used MO algorithms for space applications. Its performances are often compared to that of NSGA-II, and this algorithm has sometimes demonstrated to outperform its genetic counterpart (Li and Zhang, 2009).

The authors of this DE algorithm have leveraged the decomposition strategy to create a powerful MO optimizer. The strategy basically consists of dividing the MO problem into several sub-problems, which are optimized separately using for each one the information of neighboring sub-problems.

Three different approaches are possible for converting the problem into sub-problems: weighted sum approach, Chebyshev approach, and boundary intersection approach. In our research, we chose to implement the Chebyshev decomposition method, which is extensively described in Zhang and Li (2007a).

The scalar optimization problem, in this decomposition method, is written as:

$$\min_{\mathbf{x}\in\Omega} \mathbf{g}^{te}(\mathbf{x}|\lambda,\mathbf{z}^*) = \max_{1\le i\le m}\{\lambda_i|f_i(\mathbf{x}-z_i^*|)\} \tag{C.19}$$

where $\mathbf{z}^* = (z_1^*,\dots,z_m^*)^T$ is the reference point (i.e., for a minimization problem: $z_i^* = min\{f_i(\mathbf{x}|\mathbf{x}\in\Omega)\}$), for each $i = 1,\dots,m$.

For each Pareto optimal point $\mathbf{x}^*$ a weight factor, λ, exists such that $\mathbf{x}^*$ is the optimal solution of the expression in the Equation (C.19).

Hence, one, by just changing the weight of the weight factors, would be able to obtain different Pareto optimal solutions. This approach does not come without drawbacks: indeed, one major pitfall is that the aggregation function is not smooth for continuous multi-objective optimization problems. However, if the computation of the derivative of the aggregation function is not requested (as it is the case in Zhang and Li (2007a)), then this approach can still be used.

Now that the decomposition approach has been discussed, we can move to the algorithm description.

Let $\lambda^1, \ldots, \lambda^N$ be a set of even spread weight factors, and $\mathbf{z}^*$ be the reference point, then, using Chebyshev approach, we can decompose the problem into N scalar optimization sub-problems, and we can express the objective function of the j^{th} sub-problem as:

$$\mathbf{g}^{te}(\mathbf{x}|\lambda^j, \mathbf{z}^*) = \max_{1 \le i \le m} \{\lambda_i^j | f_i(\mathbf{x}) - z_i^*\} \tag{C.20}$$

where: $\lambda = (\lambda_1^j, \ldots, \lambda_m^j)^T$. The purpose of MOEA/D, is the minimization of all these N objective functions in a single run, simultaneously. One important aspect is that $\mathbf{g}^{te}$ is continuous of λ, which makes the optimal solution of $\mathbf{g}^{te}(\mathbf{x}|\lambda^i, \mathbf{z}^*)$ close to that of $\mathbf{g}^{te}(\mathbf{x}|\lambda^j, \mathbf{z}^*)$, when λ^i and λ^j are close to each other. Hence, all the information about the g^{te} with weight factors close to λ^i, should help the algorithm in the optimization of $g^{te}(\mathbf{x}|\lambda^i, \mathbf{z}^*)$. This is a key characteristic of MOEA/D.

In MOEA/D, a neighborhood of weight factor vector of λ^i is defined as a set of weight factors that are close to it (i.e., $\{\lambda_1, .., \lambda_N\}$). The population is then made of the best solution found so far for each sub-problem, considering that only the sub-problems in the neighborhood are used for optimizing every single sub-problem.

For each generation t, the algorithm (which uses the Chebyshev approach) maintains:

- N points in the population: $\mathbf{x}^1, \ldots, \mathbf{x}^N \in \Omega$ (where $\mathbf{x}^i$ is the current solution to the i^{th} sub-problem).
- $FV^1, \ldots, FV^N$, where FV^i represents the F-value of $\mathbf{x}^i$ (hence: $FV^i = \mathbf{F}(\mathbf{x}^i)$, for $i = 1, \ldots, N$).
- $\mathbf{z} = (z_1, .., z_m)^T$ where z_i is the best value found so far in terms of objective f_i.
- an external population (called EP) used for collecting non-dominated solutions, which are found along the way.
- probability that parent solutions are selected from the neighborhood (δ).

Also, polynomial mutation will be used in the algorithm. In particular, this operation, in this case, generates $\mathbf{y}_{new} = (y_{new,1}, \ldots, y_{new,n})^T$ from $\mathbf{y} = (y_1, \ldots, y_n)^T$ in the following way:

$$\mathbf{y}_{new} = \begin{cases} \mathbf{y}_k + \sigma_k \times (b_k - a_k) \text{ with probability } p_m \\ \mathbf{y}_k \text{ with probability } 1 - p_m \end{cases}$$

with:

$$\sigma_k = \begin{cases} (2 \times rand)^{\frac{1}{\eta + 1}} - 1 & \text{if } rand < 0.5 \\ 1 - (2 - 2 \times rand)^{\frac{1}{\eta + 1}} & \text{otherwise} \end{cases}$$

where $rand$ is a uniform distributed number from [0,1], η is the distribution index and p_m is the mutation rate (two control parameters).

The output of the algorithm will be the external population (EP). The steps followed by the algorithm can be summarized as follows:

1. Set $EP = 0$
2. Compute the Euclidean distances between any two weight vectors, and set up the T closest weight vectors for each of the weight vector considered: $\mathbf{B}(i) = (i_1, \ldots, i_T)^T$ (where $\lambda^{i_1}, .., \lambda^{i_T}$ are the T closest weight vectors to each λ_i considered).
3. Initialize a population $\mathbf{x}^1, \ldots, \mathbf{x}^N$ randomly (or with a specific method to be selected depending on the application), and define the various FV^i.
4. Initialize $\mathbf{z} = (z_1, \ldots, z_m)^T$ (their definition is typically problem related).
5. For $i = 1, \ldots, N$ do steps 6-11.

6. Uniformly generate a number $rand$ between [0,1], and set:

$$P = \begin{cases} \mathbf{B}(i) \text{ if } rand < \delta \\ \{1, \ldots, N\} \text{ otherwise} \end{cases}$$

7. Choose randomly two indexes (k and l) from P and generate a new solution $\mathbf{y}$ from $\mathbf{x}^k$, $\mathbf{x}^l$ and $\mathbf{x}^i$, using genetic operators (i.e., DE operators explained in the previous section).

8. Apply polynomial mutation (i.e., using the DE mutation techniques explained in the previous section) on $\mathbf{y}$, with probability p_m, to produce $\mathbf{y}_{new}$.

9. Update $\mathbf{z}$: for each $j = 1, \ldots, m$, if $z_j < f_j(\mathbf{y}_{new})$ then set $z_j = f_j(\mathbf{y}_{new})$.

10. Update the neighboring solutions by setting $\mathbf{x}^i = \mathbf{y}_{new}$ and $FV^j = \mathbf{F}(\mathbf{y}_{new})$ for each index $j \in \mathbf{B}(i)$, if $g^{te}(\mathbf{y}_{new}|\lambda^j, \mathbf{z}) \leq g^{te}(\mathbf{x}^j|\lambda^j, z)$.

11. Update EP by first erasing from EP all the vectors that are dominated by $\mathbf{F}(\mathbf{y}_{new})$, and then add $\mathbf{F}(\mathbf{y}_{new})$ to EP if no vectors in EP dominate $\mathbf{F}(\mathbf{y}_{new})$.

12. If any stopping criterion is satisfied, then stop the algorithm and return EP; otherwise, go back to step 6.

Bibliography

Doyle, J., and Madjarska, M., "Solar transient events and their importance for coronal heating," *Science progress*, Vol. 87, No. 2, 2004, pp. 101–130.

Mueller, M. R., D., and Gilbert, H., "Solar orbiter," *Solar Physics*, Vol. 285, No. 1-2, 2013, pp. 25–70.

Goldstein, B., Buffington, A., Cummings, A., Fisher, R., Jackson, B., Liewer, P., Mewaldt, R., and Neugebauer, M., "Solar Polar Sail mission: report of a study to put a scientific spacecraft in a circular polar orbit about the sun," *Missions to the Sun II*, Vol. 3442, International Society for Optics and Photonics, 1998, pp. 65–77.

Macdonald, M., Hughes, G., McInnes, C., Lyngvi, A., Falkner, P., and Atzei, A., "Solar polar orbiter: a solar sail technology reference study," *Journal of Spacecraft and Rockets*, Vol. 43, No. 5, 2006, pp. 960–972.

Coverstone, V., and Prussing, J., "Technique for escape from geosynchronous transfer orbit using a solar sail," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 4, 2003, pp. 628–634.

Candy, S., "Evolutionary Optimisation for a Solar Sailing Solar Polar Mission," *TU Delft, MSc Thesis*, 2002.

Garot, D., "Trajectory optimisation of a solar polar sail mission," *TU Delft, MSc Thesis*, 2006.

Spaans, J., "Improving Global Optimization Methods for Low-Thrust Trajectories," *TU Delft, MSc Thesis*, 2009.

Wie, B., "Hovering control of a solar sail gravity tractor spacecraft for asteroid deflection," *Proceedings of the 17th AAS/AIAA Space Flight Mechanics Meeting, AAS*, Vol. 7, 2007, p. 145.

Okada, T., Iwata, T., Matsumoto, J., Chujo, T., Kebukawa, Y., Aoki, J., Kawai, Y., Yokota, S., Saito, Y., Terada, K., et al., "Science and Exploration in the Solar Power Sail OKEANOS Mission to a Jupiter Trojan Asteroid," *Lunar and Planetary Science Conference*, Vol. 49, 2018.

Lyngvi, A., van den Berg, M., and Falkner, P., "Study overview of the interstellar heliopause probe," *ESA Techn. Reference Study, SCI-A/2006/114/IHP, Noordwijk*, 2007.

Dachwald, B., Ohndorf, A., and Wie, B., "Solar sail trajectory optimization for the solar polar imager (SPI) mission," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006a, p. 6177.

Sauer, C., "Solar sail trajectories for solar polar and interstellar probe missions," Vol. 103, 1999.

Wie, B., "Solar sail attitude control and dynamics, part 1," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 4, 2004, pp. 526–535.

Wie, B., "Thrust vector control of Solar sail spacecraft," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 6086.

Wie, B., Thomas, S., Paluszek, M., and Murphy, D., "Propellantless AOCS design for a 160-m, 450-kg sailcraft of the Solar Polar Imager mission," *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, 2005, p. 3928.

McInnes, C. R., *Solar Sailing: Technology, Dynamics and Mission Applications*, Springer, 1999.

Dachwald, B., Ohndorf, A., and Wie, B., "Solar sail trajectory optimization for the solar polar imager (SPI) mission," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006b, p. 6177.

Schlüter, M., "MIDACO software performance on interplanetary trajectory benchmarks," *Advances in Space Research*, Vol. 54, No. 4, 2014, pp. 744–754.

Wie, B., *Space vehicle dynamics and control*, American Institute of Aeronautics and Astronautics, 2008.

Wakker, K. F., "Fundamentals of astrodynamics," 2015.

Walker, M., "A set of modified equinoctial orbit elements," *Celestial Mechanics and Dynamical Astronomy*, Vol. 38, No. 4, 1986, pp. 391–392.

Grassia, F., "Practical parameterization of rotations using the exponential map," *Journal of graphics tools*, Vol. 3, No. 3, 1998, pp. 29–48.

Fu, B., Sperber, E., and Eke, F., "Solar sail technology—a state of the art review," *Progress in Aerospace Sciences*, Vol. 86, 2016, pp. 1–19.

Anderson, J., *Fundamentals of aerodynamics*, Tata McGraw-Hill Education, 2010.

Myatt, D., Becerra, V., Nasuto, S., and Bishop, J., "Advanced global optimisation for mission analysis and design," *Final Report. Ariadna id*, Vol. 3, 2004, p. 4101.

Vinkó, T., Izzo, D., and Bombardelli, C., "Benchmarking different global optimisation techniques for preliminary space trajectory design," *58th international astronautical congress*, International Astronautical Federation Hyderabad, India, 2007a, pp. 24–28.

Schlüter, M., Gerdts, M., and Rückmann, J., "A numerical study of MIDACO on 100 MINLP benchmarks," *Optimization*, Vol. 61, No. 7, 2012, pp. 873–900.

Schlüter, M., Wahib, M., and Munetomo, M., "Numerical optimization of ESA's Messenger space mission benchmark," *European Conference on the Applications of Evolutionary Computation*, Springer, 2017, pp. 725–737.

Parsopoulos, K., and Vrahatis, M., "Particle swarm optimization method for constrained optimization problems," *Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies*, Vol. 76, No. 1, 2002, pp. 214–220.

Yang, J., Chen, Y., Horng, J., and Kao, C., "Applying family competition to evolution strategies for constrained optimization," *International conference on evolutionary programming*, Springer, 1997, pp. 201–211.

Storn, R., and Price, K., "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, Vol. 11, No. 4, 1997, pp. 341–359.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., and Zumer, V., "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE transactions on evolutionary computation*, Vol. 10, No. 6, 2006, pp. 646–657.

Oliveto, P. S., He, J., and Yao, X., "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, Vol. 4, No. 3, 2007, pp. 281–293.

Eberhart, R., and Kennedy, J., "Particle swarm optimization," *Proceedings of the IEEE international conference on neural networks*, Vol. 4, Citeseer, 1995, pp. 1942–1948.

Karaboga, D., and Basturk, B., "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of global optimization*, Vol. 39, No. 3, 2007, pp. 459–471.

Wright, J., and Farmani, R., "Genetic algorithms: A fitness formulation for constrained minimization," *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, Morgan Kaufmann Publishers Inc., 2001, pp. 725–732.

Farmani, R., and Wright, J. A., "Self-adaptive fitness formulation for constrained optimization," *IEEE Transactions on Evolutionary Computation*, Vol. 7, No. 5, 2003, pp. 445–455.

Castellini, F., "Global optimization techniques in space missions design," *Politecnico di Milano, MSc Thesis*, 2008.

Zhang, Q., and Li, H., "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, Vol. 11, No. 6, 2007a, pp. 712–731.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, Vol. 6, No. 2, 2002, pp. 182–197.

Zhang, Q., and Li, H., "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, Vol. 11, No. 6, 2007b, pp. 712–731.

Li, X., "A non-dominated sorting particle swarm optimizer for multiobjective optimization," *Genetic and Evolutionary Computation Conference*, Springer, 2003, pp. 37–48.

Li, X., "Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function," *Genetic and Evolutionary Computation Conference*, Springer, 2004, pp. 117–128.

Fonseca, C. M., Fleming, P. J., et al., "Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization." *Icga*, Vol. 93, Citeseer, 1993, pp. 416–423.

rey Horn, J., Nafpliotis, N., and Goldberg, D. E., "A niched Pareto genetic algorithm for multiobjective optimization," *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, Vol. 1, Citeseer, 1994, pp. 82–87.

Luce, R. D., and Raiffa, H., *Games and decisions: Introduction and critical survey*, Courier Corporation, 1989.

Balling, R., "The maximin fitness function; multi-objective city and regional planning," *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2003, pp. 1–15.

Grosan, C., Oltean, M., and Dumitrescu, D., "Performance metrics for multiobjective optimization evolutionary algorithms," *Proceedings of Conference on Applied and Industrial Mathematics (CAIM), Oradea*, 2003.

Fleischer, M., "The measure of Pareto optima applications to multi-objective metaheuristics," *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2003, pp. 519–533.

Zitzler, E., *Evolutionary algorithms for multiobjective optimization: Methods and applications*, Vol. 63, Citeseer, 1999.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and Da Fonseca Grunert, V., "Performance assessment of multiobjective optimizers: An analysis and review," *TIK-Report*, Vol. 139, 2002.

Nowak, K., Märtens, M., and Izzo, D., "Empirical performance of the approximation of the least hypervolume contributor," *International Conference on Parallel Problem Solving From Nature*, Springer, 2014, pp. 662–671.

Guerreiro, A. P., Fonseca, C. M., and Emmerich, M. T., "A Fast Dimension-Sweep Algorithm for the Hypervolume Indicator in Four Dimensions." *CCCG*, 2012, pp. 77–82.

Beume, N., Fonseca, C. M., López-Ibáñez, M., Paquete, L., and Vahrenhold, J., "On the complexity of computing the hypervolume indicator," *IEEE Transactions on Evolutionary Computation*, Vol. 13, No. 5, 2009, pp. 1075–1082.

While, L., Bradstreet, L., and Barone, L., "A fast way of calculating exact hypervolumes," *IEEE Transactions on Evolutionary Computation*, Vol. 16, No. 1, 2011, pp. 86–95.

Bringmann, K., and Friedrich, T., "Parameterized average-case complexity of the hypervolume indicator," *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, ACM, 2013, pp. 575–582.

Priester, C., Narukawa, K., and Rodemann, T., "A comparison of different algorithms for the calculation of dominated hypervolumes," *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, ACM, 2013, pp. 655–662.

Bringmann, K., and Friedrich, T., "Approximating the least hypervolume contributor: NP-hard in general, but fast in practice," *International Conference on Evolutionary Multi-Criterion Optimization*, Springer, 2009, pp. 6–20.

Märtens, M., and Izzo, D., "The asynchronous island model and NSGA-II: study of a new migration operator and its performance," *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, ACM, 2013, pp. 1173–1180.

Schlüter, M., "Nonlinear mixed integer based optimization technique for space applications," Ph.D. thesis, University of Birmingham, 2012.

Dorigo, M., "Optimization, learning and natural algorithms," *PhD Thesis, Politecnico di Milano*, 1992.

Schlüter, M., Egea, J., and Banga, J., "Extended ant colony optimization for non-convex mixed integer nonlinear programming," *Computers & Operations Research*, Vol. 36, No. 7, 2009, pp. 2217–2229.

Schlüter, M., "Non-linear Mixed-Integer-based Optimization Technique for Space Applications," ESA Conference - International NPI Day, 2010.

Schlüter, M., Erb, S., Gerdts, M., Kemble, S., and Rückmann, J., "MIDACO on MINLP space applications," *Advances in Space Research*, Vol. 51, No. 7, 2013, pp. 1116–1131.

Bernardo Jr, R. M., and Naval Jr, P. C., "Implementation of an ant colony optimization algorithm with constraint handling for continuous and mixed variable domains," *Proceedings of the 10th Philippine Computing Science Congress, PCSC*, Vol. 10, 2010, pp. 95–101.

Izzo, D., "esa/pykep: Bug fixes and more support on Equinoctial Elements," , Feb. 2019. doi:10.5281/zenodo.2575462, URL `https://doi.org/10.5281/zenodo.2575462`.

Biscani, F., and Izzo, D., "esa/pagmo2: pagmo 2.11.1," , Aug. 2019. doi:10.5281/zenodo.3364433, URL `https://doi.org/10.5281/zenodo.3364433`.

Izzo, D., "Global optimization and space pruning for spacecraft trajectory design," *Spacecraft Trajectory Optimization*, Vol. 1, 2010, pp. 178–200.

Izzo, D., "1st ACT global trajectory optimisation competition: Problem description and summary of the results," *Acta Astronautica*, Vol. 61, No. 9, 2007, pp. 731–734.

Vinkó, T., and Izzo, D., "Global optimisation heuristics and test problems for preliminary spacecraft trajectory design," *Eur. Space Agency, Adv. Concepts Team, ACT Tech. Rep., Tech. Rep. GOHTPPSTD*, 2008.

Biscani, F., Izzo, D., and Yam, C., "A global optimisation toolbox for massively parallel engineering optimisation," *arXiv preprint arXiv:1004.3824*, 2010.

Mooij, E., "Orbit-State Model Selection for Solar-Sailing Mission Optimization," *AIAA/AAS Astrodynamics Specialist Conference*, 2012, p. 4588.

Montenbruck, O., and Gill, E., *Satellite orbits: models, methods and applications*, Springer Science & Business Media, 2012.

Kiusalaas, J., *Numerical methods in engineering with Python 3*, Cambridge university press, 2013.

Fehlberg, E., "Zur numerischen Integration von Differentialgleichungen durch Potenzreihen-Ansätze, dargestellt an Hand physikalischer Beispiele," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, Vol. 44, No. 3, 1964, pp. 83–88.

Altman, S., "A unified state model of orbital trajectory and attitude dynamics," *Celestial mechanics*, Vol. 6, No. 4, 1972, pp. 425–446.

Chodas, P., "Application of the extended Kalman filter to several formulations of orbit determination," *NASA STI/Recon Technical Report N*, Vol. 82, 1981.

Ackley, D., "A Connectionist Machine for Genetic Hillclimbing, vol ume SECS28 of," , 1987.

Griewank, A. O., "Generalized descent for global optimization," *Journal of optimization theory and applications*, Vol. 34, No. 1, 1981, pp. 11–39.

Rosenbrock, H., "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, Vol. 3, No. 3, 1960, pp. 175–184.

Laguna, M., and Martí, R., "Experimental testing of advanced scatter search designs for global optimization of multimodal functions," *Journal of Global Optimization*, Vol. 33, No. 2, 2005, pp. 235–255.

Babcock, W. C., "Intermodulation interference in radio systems frequency of occurrence and control by channel selection," *The Bell System Technical Journal*, Vol. 32, No. 1, 1953, pp. 63–73.

Mahdavi, M., Fesanghary, M., and Damangir, E., "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, Vol. 188, No. 2, 2007, pp. 1567–1579.

Liang, J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. C., and Deb, K., "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," *Journal of Applied Mechanics*, Vol. 41, No. 8, 2006, pp. 8–31.

Vinkó, T., Izzo, D., and Bombardelli, C., "Benchmarking different global optimisation techniques for preliminary space trajectory design," *58th international astronautical congress*, International Astronautical Federation Hyderabad, India, 2007b, pp. 24–28.

Zitzler, E., Deb, K., and Thiele, L., "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, Vol. 8, No. 2, 2000, pp. 173–195.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E., "Scalable test problems for evolutionary multiobjective optimization," *Evolutionary multiobjective optimization*, Springer, 2005, pp. 105–145.

Huband, S., Hingston, P., Barone, L., and While, L., "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 5, 2006, pp. 477–506.

Deb, K., “Multi-objective genetic algorithms: Problem difficulties and construction of test problems,” *Evolutionary computation*, Vol. 7, No. 3, 1999, pp. 205–230.

Zaharie, D., “A comparative analysis of crossover variants in differential evolution,” *Proceedings of IMCSIT*, Vol. 2007, 2007, pp. 171–181.

Holland, J. H., et al., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press, 1992.

Deb, K., and Agrawal, S., “A niched-penalty approach for constraint handling in genetic algorithms,” *Artificial Neural Nets and Genetic Algorithms*, Springer, 1999, pp. 235–243.

Li, H., and Zhang, Q., “Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II,” *IEEE transactions on evolutionary computation*, Vol. 13, No. 2, 2009, p. 284.