POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Department of Mechanical and Aerospace Engineering

MASTER'S DEGREE THESIS

Development of a Tool for the Design and Verification of Thermal Control Systems of Small Sats

Supervisors:

Prof. Sabrina Corpino

Prof. Fabrizio Stesina

Candidate:

Davide Cosenza

July 2023

I dedicate this thesis to my dear parents Ermanno and Margherita

Acknowledgement

First and foremost, I would like to thank Professor Sabrina Corpino for allowing me to pursue my passions through this thesis and the Spei Satelles mission, and Daniele Calvi, for being a point of reference and a great source of inspiration for the months of work on the thesis.

Then, I thank all the colleagues and friends of the Spei Satelles project for sharing their enthusiasm with me, especially Francesco Lucia, with whom I have shared every moment of this journey.

My two parents, who put as much effort as I did to enable me to achieve my goals, really deserve special praise.

I also want to thank all my family and friends who always believed in me, especially Tommaso, Alfredo, Paolo, Stefano, Irene, and Francesca who made these years at Politecnico enjoyable and unforgettable through their friendship.

In conclusion, I would like to thank my precious Stefi, who provided me with the strength to achieve this goal and always supported me along the way.

Abstract

In the context of an ever-growing number of small satellite missions, characterized by low costs, short project lifecycles with fast development time, and a multitude of different mission objectives, various design-aiding tools are needed for every phase of the projects. In particular, in the early phases, where wider margins for modification of the design exist, there is the necessity for comprehensive software that combine the aspect of numerical simulations with automated design generation features and efficient subsystem optimization capabilities. One of the crucial aspects of the design of small satellites is the Thermal Control System, whose importance and relevance is increasing even more in the last years thanks to a growing number of nano and microsatellites operating beyond Low Earth Orbit, in challenging interplanetary environments.

This thesis focuses on the improvement and development of a MATLAB-based tool created by the Department of Mechanical and Aerospace Engineering of the Politecnico di Torino, Small Satellites Thermal Tool (S2T2), with special attention to the module of automated optimization of system design. A first state-of-the-art review of the main Multi-Objective Optimization techniques and algorithms is given, with additional details on the "Reference point based archived many objective Simulated Annealing" (RSA) algorithm, which showed promising potential. Then, an improvement of the source code and User Interface of the Design module of S2T2 is presented, starting from the major limitations of the software, and reworking the existing flow, to reach a new, expanded and computationally more efficient Design module.

The work is then tested and validated using different case studies: first, a suite of popular benchmark problems for Multi-Objective Optimization is used to compare the performance of the algorithms implemented, and then an application

of the tool to the SROC mission is used to test the algorithms and the results in a real-world case study. Finally, the insights and outcomes of the previous studies are applied to the Spei Sitellas mission, delivering a new release of S2T2.

Contents

List of Figures

List of Tables

List of Acronyms

AMOSA	Archived Multi-Objective Simulated Annealing
API	Application Programming Interface
ASF	Achievement Scalarizing Function
AVG	Average
BoL	Beginning of Life
CNR	Consiglio Nazionale Delle Ricerche (National Research Council)
C&DH	Command & Data Handling
CoM	Centre of Mass
ComSys	Communication System
ConOps	Concept of Operations
DC	Duty Cycle
DET	Direct Energy Transfer
DTLZ	Deb-Thiele-Laumanns-Zitzler (problems)
ECI	Earth Centred Inertial
EPS	Electrical Power System
FDM	Finite Difference Model
FR4	Flame Retardant Level 4
GD	Generational Distance (metric)
GMM	Geometric Math Model
GMT	Greenwich Mean Time
GODLIKE	Global Optimum Determination by Linking and Interchanging Kindred Evaluators (algorithm)
HL	Hard Limit
HR	Hypervolume Ratio (metric)
HV	Hypervolume
IGD	Inverted Generational Distance (metric)
ID	Identifier

IMU	Inertial Measurement Unit
IR	Infrared
LB	Lower Bound
LEO	Low Earth Orbit
LIDAR	Light Detection And Ranging
LV	Launch Vehicle
MCC	Mission Control Centre
MCRT	Monte Carlo Ray Tracing
MOEA/D	Multi-objective Evolutionary Algorithm based on Decomposition (algorithm)
MOGWO	Multi-Objective Grey Wolf Optimizer (algorithm)
MOO	Multi-Objective Optimization
MOOP	Multi-Objective Optimization Problem
MOPSO	Multi-Objective Particle Swarm Optimization (algorithm)
MPFE	Maximum Pareto Front Error (metric)
MS	Maximum Spacing (metric)
NGSA	Non-dominated Sorting Genetic Algorithm (algorithm)
PCB	Printed Circuit Board
PF	Pareto Front
RSA	Reference point based archived many objective simulated annealing (algorithm)
RVEA	Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization (algorithm)
SL	Soft Limit
SP	Spacing (metric)
S/C	Spacecraft
S2T2	Small Satellite Thermal Tool
SR	Space Rider
SROC	Space Rider Observer Cube (spacecraft/mission)
TCS	Thermal Control System
TD	Thermal Desktop

TMM	Thermal Math Model
TRL	Technology Readiness Level
UB	Upper Bound
UI	User Interface
ZDT	Zitzler-Deb-Thiele (problems)

List of Symbols

f_m	Cost Function/Objective Function
x	Design Vector
$x^{(L)}$	Lower Bounds of the design vector
$x^{(U)}$	Upper Bounds of the design vector
M	Number of optimization Objectives
α_{ext}	Solar absorptivity of external side of the satellite structure faces
ϵ_{ext}	Infrared emissivity of external side of the satellite structure faces
ϵ_{int}	Infrared emissivity of internal side of the satellite structure faces
N_{OPT}	Number of design variables for the optimization of optical properties of the S/C structure
N_H	Number of design variables for the optimization of heaters
N_L	Number of design variables for the optimization of conductance links (thermal straps)
N_{item_L}	Number of items selected for the optimization of conductance links (thermal straps) / Number of conductance links
ID_{start}	Index of the starting node of the conductance links (thermal strap)
ID_{end}	Index of the ending node of the conductance links (thermal strap)
G_L	Thermal conductance of conductance link (thermal straps) [W/K]
k_L	Thermal conductivity of conductance link (thermal straps) [W/m/K]
A_L	Cross-sectional area of conductance link (thermal straps) [m^2]
L_L	Length of conductance link (thermal straps) [m]
V_L	Volume of conductance links (thermal straps) [m^3]
U	Utility value
G_d	Temperature gap between computed and operative temperatures
T^C	Computed temperatures [°K], [°C]
T^O	Operative temperatures [°K], [°C]
P_H	(Total) Heater Power
f	Linear scaling functions

W	Utility value weights
F_p	Spacecraft to Planet View Factor
F_{space}	Spacecraft to Space View Factor
C	Connectivity matrix of the Thermal Finite Difference Model
G_c	Thermal Conductance matrix of the Thermal Finite Difference Model for Conductive Heat Exchange
G_{hc}	Thermal Capacitance matrix of the Thermal Finite Difference Model
G_{irr}	Thermal Conductance matrix of the Thermal Finite Difference Model for Radiative Heat Exchange
N_N	Total number of nodes of the Finite Difference Model
$N_{F/N}$	Number of surfaces that a node of the Finite Difference Model is contained in
Q_s	Heat radiated from the Sun [W/m^2]
Q_{ir}	Infrared radiation of the central body/Earth [W/m^2]
Q_a	Albedo heating from the central body/Earth [W/m^2]
I_s	Solar Irradiance [W/m^2]
θ_j	Angle between element normal of a node and Sun vector
F_{ecl}	Eclipse factor
β_j	Angle between element normal of a node and the position vector in Earth Centred Inertial reference frame
a	Albedo coefficient
$F_{j,p}$	View Factor between the surface elements of a node and the central body/planet/Earth
I_p	Infrared radiation of the central body/planet/Earth [W/m^2]
T_{orbit}	Orbit Period [s]
Δt	Time step of the simulation [s]
N_{item}	Number of Items of the Thermal Model
T_{SS}	Steady-State Temperature
C_T	Cost regarding Temperatures in the Cost Function
C_H	Cost of Heaters in the Cost Function
C_L	Cost of Conductance Links in the Cost Function
σ	Stefan-Boltzmann constant
V_{f_G}	View Factor matrix of the nodes of the Thermal Model

σ_U	Standard Deviation of Utility value
x_p	Perturbed Design Vector
c_p	Specific Heat [J/kg/K]
m	Mass

Chapter 1

Introduction

1.1 Context of the Study

Small Satellites are playing a central role in the space economy over the last few years, with the participation of many different types of developers and operators.

The exact definition of a Small Satellite can vary, but generally, they have a mass ranging from a few kilograms to a few hundred kilograms [1], [2]. They are characterized by their compact size, which allows for lower launch costs and the ability to be deployed in large constellations. Small Satellites, or Small Sats, are capable of fulfilling many diverse objectives, ranging from commercial use such as communication services to Earth observation, scientific research in general, technology demonstration, and education.

Small satellites have gained popularity in recent years due to advancements in miniaturization, electronics, and launch technology. They provide a cost-effective means of accessing space, enabling universities, research institutions, and commercial entities to undertake space missions that were once exclusive to larger and more expensive satellites. The deployment of small satellite constellations has revolutionized Earth observation, global connectivity, and remote sensing applications, allowing for increased coverage, faster revisit times, and improved data collection capabilities.

This trend is supported by recent data, looking at Figure 1 it is evident how a very high share of the current spacecraft upstream sector is devoted to the design. launch and operation of Small Sats, with few commercial constellations giving the highest contribution to these numbers [2].

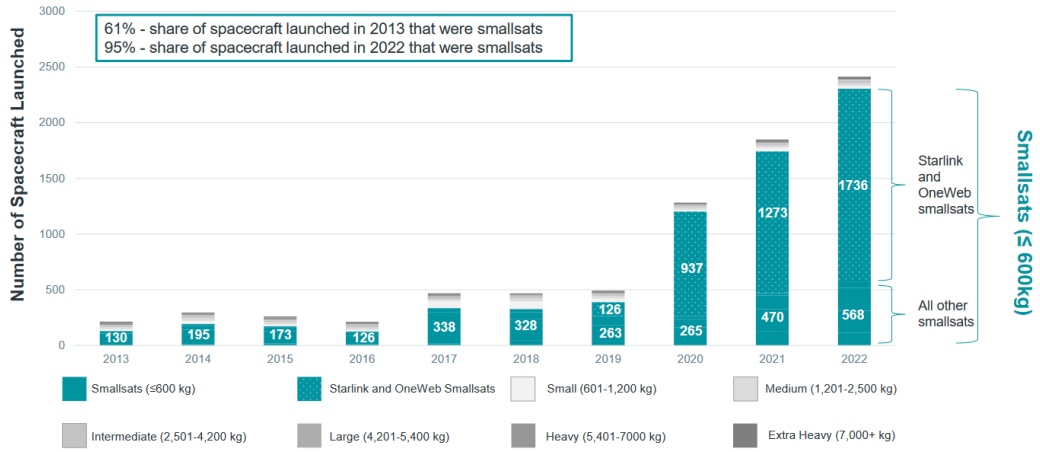


Figure 1: SmallSats in Context, Spacecraft Launched 2013 – 2022, by Mass Class [2].

Figure 2 analyses in more detail the distribution of the total SmallSats by Mass Class. The Starlink and OneWeb constellations make up the majority of satellites, with spacecraft in the range of a few hundred kilograms of mass. The trend of the last years saw a dramatic increase in the numbers thanks to those large constellations, but even smaller mass classes are growing, as visible in Figure 3.

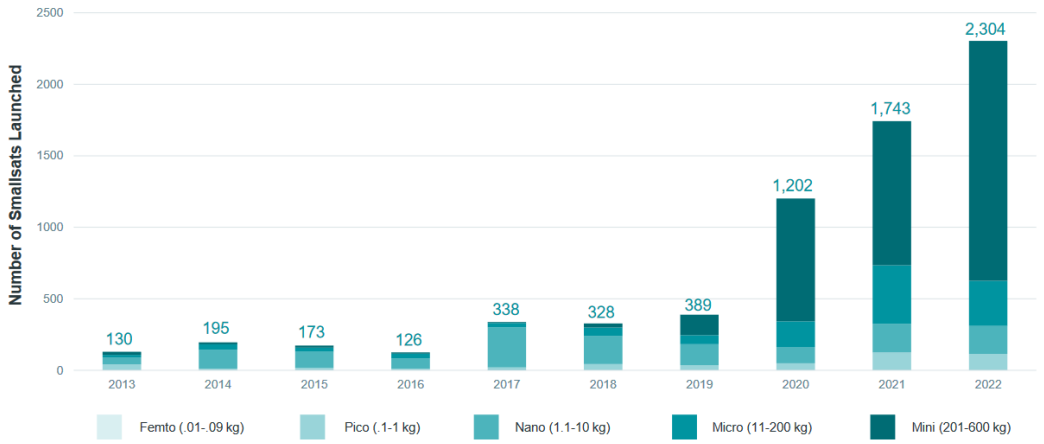


Figure 2: SmallSats in Context, Smallsats 2013 – 2022, by Mass Class [2].

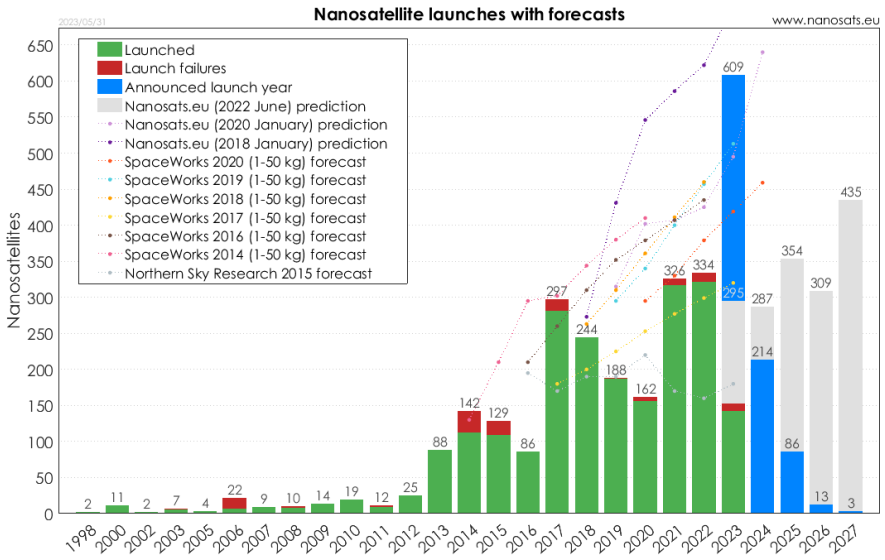


Figure 3: History and predictions of nanosatellite launches [3].

Given these numbers, there is a need for tools to aid all phases of the life cycle of a space mission. One of the most important aspects is the design and verification of the Thermal Control System, which is a key element that needs to work in conjunction with many others to guarantee the success of a space mission. Thermal Control System is the framework in which this thesis collocates.

1.2 Thermal Control Systems of Small Sats

The Thermal Control System (TCS) is responsible for implementing temperature control, it is present on board all satellites and spacecraft in general and it is necessary to keep the on-board equipment, especially the electronics, within certain temperature ranges so that they can operate efficiently and reliably. In addition, it controls temperatures to prevent structural stresses due to the different coefficients of thermal expansion of materials.

Implementing thermal control of a system means making an energy balance, i.e. seeking a balance between the heat absorbed from the outside, and possibly that generated by the system itself, and that emitted to the outside.

The first step is to define the thermal requirements and any existing constraints for the design, which derive from the thermal requirements of the

various components installed on board and the characteristics of the satellite, as well as the mission in which the spacecraft operates.

We then move on to determine the thermal environment, i.e. calculate all the heat flows that the satellite will receive from the external environment. The next step is the definition of the thermal system architecture and all the design choices to be implemented in the system design. All the previous steps are then iterated to converge and define the optimal design for the thermal control system.

The thermal engineer deals with three macro areas:

- Analysis: downstream of the knowledge of all the components of the system and the space environment in which it operates, calculate the temperatures that the subsystems and the structure reach and compare them with the limits imposed by requirements and constraints.
- Design: design of the thermal control system, looking for solutions that allow the components to operate within their operative ranges.
- Verification: in the initial stages of the project, this is carried out through analysis and simulations, while the subsequent stages are carried out experimentally to qualify and accept the system for launch. These phases are typically costly and require special facilities to be carried out, such as vacuum chambers and thermal cycling chambers.

The thermal control system must be able to manage the system's temperatures during all phases of the mission, which cover not only all on-orbit operations but also all pre-launch phases and even the launch phase itself.

The design phase of the TCS of Small Sats poses some unique challenges that stem from several intrinsic properties, summarized in Table 1 [4]. The typical solutions can either be passive or active. By this definition, passive thermal control solutions maintain component temperatures without using powered equipment. Passive systems are typically associated with low cost, volume, weight, and risk, and are advantageous to spacecraft with limited, mass, volume, and power, like SmallSats and especially CubeSats. Active thermal control methods, on the other hand, rely on input power for operation and have been shown to be more effective in maintaining tighter temperature control for components with stricter temperature requirements or higher heat loads. Current state-of-the-art active thermal technologies for passive and active systems are shown in Table 2 and Table 3 [4].

Table 1: SmallSat Thermal Control Challenges [4].

SmallSat Property	Challenge
Low thermal mass	The spacecraft is more reactive to changing thermal environments
Limited external surface area	There is less real estate to be allocated to solar cells, designated radiator area, and/or viewports required for science instruments
Limited volume	There is less space for electronic components, science instruments, and thermal control hardware. Components can be more thermally coupled, and it can be harder to isolate different thermal zones.
Limited power	There is less power available for powered thermal control technology.
Power Density	There is a big challenge to dissipate power as electronics are stacked close to each other, sometimes with no direct path to radiators.
MLI Edge Effects	MLI can "short" along the edges resulting in degraded performance, not specific to SmallSats; more of a general spacecraft issue.

Table 2: State-of-the-art of Passive Thermal Systems [4].

Manufacturer	Products	TRL in LEO Environment
AZ Technology, MAP, Astral Technology Unlimited, Inc., Dunmore Aerospace, AkzoNobel Aerospace Coatings, Parker-Lord, Medtherm	Paint and Coatings	7-9
Sheldahl, Dunmore, Aerospace Fabrication & Materials, 3M	Tapes	7-9
Sheldahl, Dunmore, Aerospace Fabrication & Materials	MLI Materials	7-9
NASA GSFC, Aerothreads, Aerospace Fabrication & Materials	MLI Blanket Fabrication	7-9
Space Dynamics Laboratory, Thermal Management Technologies, Boyd Corp., Technology Applications, Inc., Thermotive Technology, Redwire Space	Thermal Straps	7-9
Bergquist, Parker Chomerics, Aerospace Fabrication & Materials, AIM Products LLC, Intermark USA, Indium Corporation, Dow Corning, NeoGraf, Laird Technologies, Avantor (NuSil)	Thermal Interface Materials and Conductive Gaskets	7-9
Sierra Lobo, Aerospace Fabrication and Materials	Sun Shield	4-7
NASA Goddard Space Flight Center (GSFC)	Thermal Louvers	7-9
Aerospace Fabrication and Materials, Thermal Management Technologies, Redwire Space	Deployable Radiators	5-6
Aavid Thermacore, Inc., Advanced Cooling Technology, Inc., Redwire Space	Heat Pipes	7-9
Thermal Management Technologies, Active Space Technologies, Advanced Cooling Technology, Inc., Redwire Space	Phase Change Materials/ Thermal Storage Units	7-9
Starsys, Redwire Space	Thermal switches	7-9
Thermal Management Technologies	Multifunctional Thermal Structures	4-5

Table 3: State-of-the-art of Active Thermal Systems [4].

Manufacturer	Products	TRL in LEO Environment
Minco Products, Inc., Birk Manufacturing, All FlexFlexible Circuits, LLC., Fralock, Tayco Engineering,Inc., Omega	Electrical Heaters	7-9
Ricor-USA, Inc., Creare, Sunpower Inc., Northrop Grumman, NASA Jet Propulsion Lab, and Lockheed Martin Space Systems Company	Cryocoolers	5-6
Marlow, TE Technology Inc., Laird	Thermoelectric Coolers	7-9
Lockheed Martin	Fluid Loops	4-5
NASA Small Spacecraft Technology program	Active Thermal Architecture	4-6

To gain an edge in the early phases of development of a space mission it is usually necessary to evaluate many different design solutions, weighting costs and performances, before converging to the final design. The automated generation and optimization of the design of thermal control systems is a powerful approach in this context, as it can reduce the time needed by the user to iterate over many different simulations, leading to a larger and better set of trade-off solutions. Moreover, at the start of the design of a space mission, fewer constraints are imposed on the thermal engineer, thus spanning a wide range of design possibilities, while keeping in mind the different contrasting objectives to be optimized, is not only possible but advised.

These necessities were the main drivers for the work of this thesis, and they lead to the definition of the objectives presented in the next section.

1.3 Scope and Objectives

Given these premises, the scope of the work of this thesis was to deliver a simulation tool designed to help the early design phases of Small Sats missions. This tool should be capable of performing steady state and transient thermal simulations using a Finite Difference method for solving the heat equations. It should be capable of handling thermal models with a number of nodes in the hundreds for each phase of the thermal analysis and performing calculations efficiently and reliably.

On top of that, the most important objective for this work, was the realization, in the framework of such a tool, of a multi-objective optimizer for TCS design, complete with many different options and with state-of-the-art performances. This optimization module should be able to evaluate tens of thousands of different

design choices and select the best-performing ones through different metrics of cost and performance. The whole optimization process needs to be fast and robust, but also versatile, and adaptable to many different applications.

The testing and validation aspect when developing simulation software is as important as the development. The performances of this tool need to be evaluated using test problems and compared with commercial software before being used in real-world applications.

1.4 Thesis Structure

The work done in this thesis is organized as follows:

Chapter 1 contains an introduction to the thesis, with some context, some basic knowledge regarding Small Sats and their thermal control system and the objectives of the work presented in this document.

Chapter 2 provides details on the theory behind multi-objective optimization, with some more in-depth explanation of one promising algorithm, called RSA. Other multi-objective optimization algorithms are also presented, as well as the main performance metrics used by the scientific community to measure the results of the optimizers.

Chapter 3 introduces the MATLAB® tool around which the entire work was centred: Small Satellite Thermal Toolkit (S2T2), a software developed at Politecnico di Torino. An in-depth review of the state of development of the software before the work of this thesis is given, then the improvements and modifications made to the source code of the S2T2 application are explained.

Chapter 4 presents the validations performed on S2T2 to assure that the desired level of speed and accuracy of the software was reached, and the applications to two real-world space missions: SROC and Spei Satelles. For the validations of the multi-objective optimization algorithms the two popular benchmarking suites composed of the ZDT and DTLZ problems were used, while to validate the temperature results of the thermal simulations, comparisons with the commercial software C&R Thermal Desktop® are presented.

Chapter 5, finally, reports some conclusive considerations about the work performed and lists in detail the future developments of the work of this thesis, focusing on the follow-up development work for S2T2.

Chapter 2

Multi-Objective Optimization

To understand how Multi-Objective Optimization (MOO) can be used effectively in the context of the thermal design of nanosatellites and microsatellites an overview of the main relevant points of the theory of multi-objective optimization is given in this chapter, as well as a presentation of some of the most popular MOO algorithms.

An in-depth section is dedicated to one in particular: a multi-objective simulated annealing algorithm called RSA, because of the potential it showed in the context of this study. Finally, a selection of some of the most used performance metrics to compare the results of MOO algorithms is presented.

2.1 Overview of Multi-Objective Optimization Theory

MOO techniques are used in dealing with problems which are characterized by the presence of two or more aspects to be optimized simultaneously. The parameters to be minimized (or maximized) are typically called *objectives*, and they are often conflicting, meaning that improving one objective may lead to a degradation in another objective. This is especially common in the engineering field, in which every solution of a design problem presents a certain degree of effectiveness, but also comes with its peculiar associated costs.

A MOO Problem (MOOP) is usually formalized in the following form [5]:

$$\begin{cases} \text{Minimize/Maximize:} & f_m(x), & m = 1,2,\dots,M; \\ \text{subject to:} & g_j(x) \geq 0, & j = 1,2,\dots,J; \\ & h_k(x) = 0, & k = 1,2,\dots,K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1,2,\dots,n; \end{cases} \quad (1)$$

Where $f_m(x)$ are the M objective function corresponding to the M objectives to be optimized. In this thesis the *objective functions*, sometimes also called *fitness functions*, will always be considered as functions to be minimized, thus they can also be interpreted as *cost functions*, where the lower their value is, for each objective, the better. The input of the objective functions is the *design vector* x, composed of n decision variables x_i. Every design vector that satisfies the constraint g_j and h_k and whose decision variables lie inside the range specified by the lower bounds $x_i^{(L)}$ and upper bounds $x_i^{(U)}$ is considered a *feasible solution* of the MOOP. If the lower bound and upper bound vectors are specified the problem is defined as *bounded*, and if the constraints are specified the problem is *constrained*, while in the other cases, the problem is called *unbounded* and *unconstrained*, respectively. In this study, the attention will be focused on bounded and unconstrained problems because in the applications examined the design variables represent physical parameters which have a limited range of variability concerning the complete real value domain, however, they are considered continuous and independent from one another, i.e. the variation of one does not limit the range of variability of the other ones, and thus no additional constraints are required.

The domain in which the decision variables can be found is called *decision space*, and it is a n-dimensional space delimited by the upper and lower bounds. By calculating the objective function of a design vector, a vector of M objective values is obtained, and it can be represented as a point in the M-dimensional *objective space*. A graphical representation of these domains is shown in Figure 4. In this thesis, the names “individuals”, “design points”, “solutions”, or sometimes simply “points” will be used interchangeably to indicate points of the objective space.

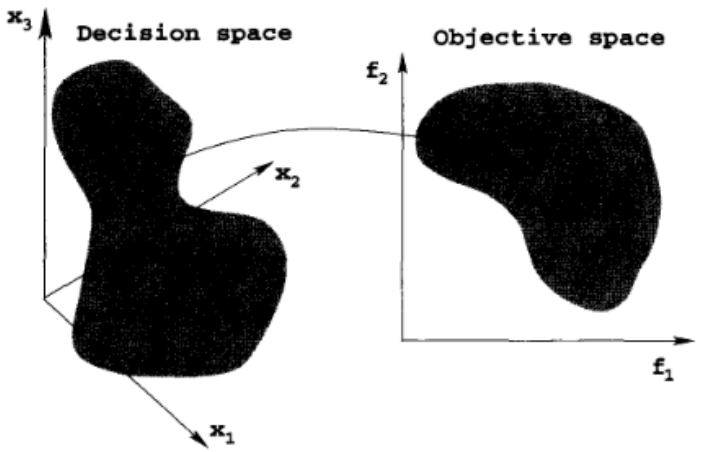


Figure 4: Example of 3-dimensional decision space with its corresponding 2-dimensional objective space [5].

In the context of MOO optimization, the ranking of the solution follows an approach different from single objective optimization since every solution has multiple costs. The ranking process, needed to establish the set of optimal solutions, can be performed using the definition of *Pareto dominance* (or simply *dominance* for short): one solution is said to dominate another if it is at least as good as the other solution in all objectives and strictly better in at least one objective. Formally, considering two solutions x_A and x_B, solution x_A is said to dominate solution x_B if:

- For all the $m = 1, \dots, M$ objectives the objective value $f_m(x_A)$ is less than or equal to the objective value of $f_m(x_B)$. In symbols:

$$f_m(x_A) \leq f_m(x_B) \quad \forall m = 1, \dots, M \tag{2}$$

- There is at least one objective for which the objective value $f_m(x_A)$ is strictly better than the objective value $f_m(x_B)$. In symbols:

$$\exists m \in 1, \dots, M \mid f_m(x_A) < f_m(x_B) \tag{3}$$

If both equation (2) and (3) are satisfied, it can be said that solution x_A dominates solution x_B. In Figure 5 the area marked as "Dominates" is where solutions dominated by x_A lie. If instead neither x_A dominated x_B nor x_B dominates x_A, the two solutions are not directly comparable, they belong to the

same *front,* and neither is better than the other. The first front that contains the set of solutions not dominated by any other point is called the *optimal Pareto front*, and the solution belonging to the Pareto front are called *non-dominated solutions*; see Figure 5 [6].

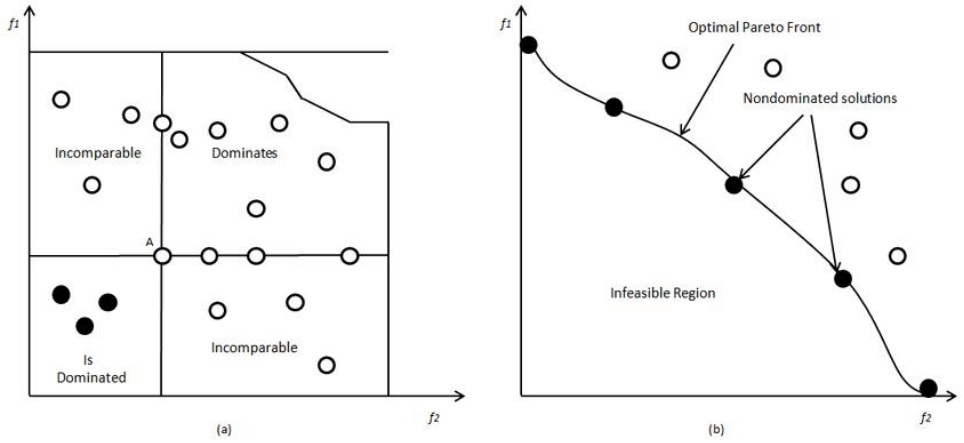


Figure 5: Dominance regions with respect to solution A (on the left) and representation of the optimal Pareto front (on the right)

In a MOOP there are typically two main goals: the first and most obvious one is to find a set of solutions as close as possible to the true Pareto front of the problem (which in general is not known in real-world problems), but also to find a set of non-dominated solution as diverse as possible, spanning the entire objective space, so that a wider range of possibilities is available for the final choice of optimal solution.

This critical last step is usually performed by the user, which can select the most suitable point from all the ones on the Pareto front, making appropriate trade-off considerations. When this process needs to be automated a *decision-maker algorithm* is required. These algorithms employ different criteria to select the best solutions among a set of equivalently optimal ones. One of the simplest and most common techniques in this sense is the *weighted sum approach*, where for every objective a weight is assigned, and the final objective value is the weighted average of the single costs. It requires user input in the form of the weight vector, but in these cases, the weights are often decided before the optimization by an "expert user" that already knows the problem quite well and has the competence needed to come up with meaningful weight values. Other methods include ϵ- constraint and goal programming [5].

2.2 Multi-Objective Optimization Algorithms

In the scientific literature exist many different algorithms to solve MOOPs, each one with its unique strengths and characteristics. Many classifications are possible; here some basic concepts that apply to large numbers of these algorithms are presented.

MOOPs are typically nonlinear nonconvex problems, meaning that it is not possible (or not known how) to solve them in polynomial time. Furthermore, the presence of multiple objectives encourages an evolutionary approach to solving these kinds of problems, which is the basis for many of the most effective MOO algorithms [5].

A key aspect of evolutionary multi-objective algorithms is the balance between exploration and exploitation. In the first iterations of optimization, it is desirable to explore a wide range of possibilities for the design variables, covering large portions of the design space and maintaining good diversity. Once the design space is more known it is important to gradually focus more of the optimization efforts on the most promising solutions, the one that dominates the other, exploiting as much as possible the good candidates to generate new individuals before arriving at the final optimal solutions.

Many evolutionary algorithms are inspired by the natural evolutionary process of living organisms. Often the design vector is imagined as the DNA of a living individual and the process of generating new design points can be inspired by the biological phenomena of mutation, mating, and crossover. When this applies the algorithms can be classified as genetic algorithms [5].

Some MOO algorithms that are designed to handle large numbers of objectives use a reference vector approach (also known as reference points or reference lines approach): the objective space is filled uniformly (or with some pre-determined distribution) by unit vectors starting from the origin and spreading in all directions of the positive quadrant. These vectors represent the desired final distribution of optimal trade-off solutions. During the optimization, each individual of the population is assigned to one reference vector, based on closeness, and when the size of the population needs to be reduced, the first individuals which are discarded are taken from the most populated design vectors, to achieve a good balance in the spread of solutions. This approach was found to be particularly effective when more than 2 objectives are present and, in general,

it has the advantage of helping to maintain an equilibrium between convergence and diversity [7].

To improve the robustness of the optimized solution and to exploit the specific strengths of different algorithms, sometimes a successful approach can be the hybridization of multiple optimizers. Hybrid algorithms combine parts from different other MOO algorithms to balance out their qualities, or in some other cases switch from one algorithm to another, while keeping the same population or creating distinct groups for every different technique applied.

On some architecture an approach that may significantly increase performances, without altering the convergence speed and the number of objective function evaluations required to arrive at the optimal solution is the parallelization or vectorization of the computation. When the computing machine that performs the optimization has parallel computing capabilities is useful to use algorithms that allow simultaneous objective function evaluations or that perform computation in a vectorized way, exploiting the strengths of the specific programming language used. If done correctly the approach may drastically reduce computational time without altering any other quality of the final results.

In the next sub-sections, a collection of the most popular and most used MOO algorithms is presented, along with a brief description for each entry. The selection of the algorithms was based on a review of the state of the art of scientific literature in this field [5] [8] [9].

2.2.1 NGSA-II (Non-dominated Sorting Genetic Algorithm II)

The NGSA-II (Non-dominated Sorting Genetic Algorithm II) is a popular evolutionary algorithm that combines the concepts of non-dominated sorting and genetic operators to efficiently search for Pareto-optimal solutions.

In NGSA-II, a population of candidate solutions is evolved through successive generations. The algorithm employs a fast non-dominated sorting technique to classify individuals into different Pareto fronts based on their dominance relationships. After the sorting process, the crowding distance is calculated for individuals on each front to maintain diversity. The crowding distance measures the density of solutions around each design point, promoting the exploration of different regions in the objective space.

NGSA-II then uses different selection, crossover, and mutation operators to create new offspring solutions from the parent population. These genetic operators are applied to the individuals in the non-dominated fronts, giving preference to solutions with higher crowding distances to maintain diversity. The offspring population is then combined with the parent population to form a combined population. This combined population undergoes non-dominated sorting and crowding distance calculation again to identify the next generation's non-dominated fronts.

In MATLAB environments the function *gamultiobj* of the Global Optimization Toolbox natively implement a modification of NGSA-II.

2.2.2 NGSA-III (Non-dominated Sorting Genetic Algorithm III)

The NGSA-III (Non-dominated Sorting Genetic Algorithm III) is an advanced evolutionary algorithm specifically designed for solving many-objective optimization problems, where traditional algorithms may struggle due to the increased dimensionality. It extends the non-dominated sorting and genetic operators of NGSA-II to handle a large number of objectives more efficiently.

NGSA-III employs an evolution process which starts with the sorting of the population using a fast non-dominated sorting technique which, just like NGSA-II, classifies the individuals into different non-dominated fronts based on their dominance relationships. Then, environmental selection occurs: a niche count operator is used to select individuals from different fronts based on their crowding distances. This operator aims to maintain a balance between convergence and diversity by favouring solutions with lower density. NGSA-III also introduces the concept of reference points to guide the search towards different regions of the Pareto front. The algorithm adaptively adjusts the positions of reference points based on the population distribution.

2.2.3 MOPSO (Multi-Objective Particle Swarm Optimization)

The MOPSO (Multi-Objective Particle Swarm Optimization) algorithm is a population-based optimization technique that utilizes the concept of swarm intelligence to solve multi-objective optimization problems. It is an extension of the traditional Particle Swarm Optimization (PSO) algorithm adapted for handling multiple objectives simultaneously.

In MOPSO, a population of particles moves through the search space, with each particle representing a potential solution. Each particle is defined by its position (which corresponds to a potential solution in the objective space) and velocity, which are updated based on the particle's historical best position and the global best position found by the entire swarm. The main modification in MOPSO compared to the traditional PSO lies in the selection of the global best position. Instead of selecting a single best solution, MOPSO maintains a set of non-dominated solutions known as the Pareto archive or repository. The velocity update equation in MOPSO incorporates both the cognitive and social components, thus balancing exploitation and exploration. The cognitive component guides particles towards their historical best positions, while the social component orients them towards the global best positions in the Pareto archive.

2.2.4 MOGWO (Multi-Objective Grey Wolf Optimizer)

The MOGWO (Multi-Objective Grey Wolf Optimizer) takes inspiration from the natural phenomenon of the hunting behaviour of grey wolves. It is a multi-objective variant of the Grey Wolf Optimizer.

MOGWO ranks the population based on dominance relationships and calculates crowding distance to try and maintain a diverse set of non-dominated solutions. The algorithm leverages the grey wolf's hunting behaviour and incorporates different update equations for each category of solutions (alpha, beta, delta, and omega), mimicking the hierarchy structure of a pack of wolves to maintain diversity.

2.2.5 GODLIKE (Global Optimum Determination by Linking and Interchanging Kindred Evaluators)

GODLIKE is a hybrid multi-objective optimization algorithm that combines several population-based global optimization schemes.

GODLIKE utilizes a genetic algorithm, differential evolution, particle swarm optimization, and adaptive simulated annealing algorithms simultaneously. These algorithms run concurrently, allowing for parallel exploration of the search space. Furthermore, members from each population are occasionally interchanged to prevent premature convergence to local optima. The primary goal of GODLIKE is to enhance the robustness of optimization processes rather than focusing solely on

efficiency, with limited needs to fine-tune the algorithm for every optimization problem.

2.2.6 *Paretosearch*

Paretosearch is a direct search method specifically designed for solving multi-objective optimization problems. It iteratively improves a set of candidate solutions based on their Pareto dominance relationships. Paretosearch utilizes random sampling and local search operators to explore the objective space efficiently. Like NGSA-II this algorithm can be executed natively in the MATLAB environment, calling the function *paretosearch*.

2.2.7 MOEA/D (Multi-objective Evolutionary Algorithm based on Decomposition)

MOEA/D addresses multi-objective problems by decomposing them into a set of scalar sub-problems. It aims to optimize each sub-problem simultaneously, striking a balance between convergence and diversity. The initialization phase is followed by a loop that involves reproduction, variation (with differential evolution and neighbourhood search) and population update steps. The key aspect of MOEA/D is the decomposition update, which adjusts the weight vectors associated with each sub-problem.

2.2.8 RVEA (Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization)

RVEA is designed specifically for many-objective optimization problems. It utilizes reference vectors to guide the search towards the Pareto front, ensuring a diverse and representative set of solutions. After the initialization of the population, it generates reference vectors that cover the objective space uniformly. In the optimization loop, RVEA involves mating selection, variation, environmental selection, and reference vector update steps. The reference vector update adapts the vectors to the evolving population, facilitating the exploration of different regions of the Pareto front.

2.2.9 RSA (Reference point based archived many objective simulated annealing)

RSA is a many-objective version of the simulated annealing algorithms from a single objective. It is a variant of the older AMOSA (Archived Multi-Objective Simulated Annealing) algorithm, with performance improvements and some adaptations to be more suitable to solve problems with a high number of objectives.

The key point of the algorithm is the utilization of a variable called temperature, which determines the probability of accepting uphill solutions after generating a new point. The temperature starts from a high value (typically 100) and slowly cools down to values approaching zeros, following a predetermined law. When the temperature is higher, at the start, the algorithm accepts more unfavourable solutions, focusing on the exploration of the design space, whereas in the final iterations, a low number of uphill solutions are kept, to prefer the exploitation of the already available individuals.

The main additions brought by RSA over other simulated annealing algorithms are the introduction of reference vector and clustering of solutions, to help maintain diversity and explore more uniformly the objective space; the usage of an archive to store the solutions; the approach of archive-to-archive transition instead of point-to-point transition, which means that the quality of the entire archive is evaluated when deciding to accept or reject a new point; an ensemble of mutation techniques to generate new design points.

2.3 Implementation Details of RSA

In this section, some details on the implementation of RSA in a MATLAB environment are given, with an explanation of the main step of the algorithm with the help of flow charts. The algorithm was developed by R. Sengupta S. Saha and further details can be found in their publication [10].

RSA uses an archive, also called a repository in similar contexts, to keep track of the current population of individuals. It can be considered as the mating pool of the genetically inspired algorithms: the new solutions are always generated by perturbing one individual of the archive and are progressively added to the archive. During the optimization two limits are set on the archive: a soft limit (SL) and a hard limit (HL), the archive size is always between these values. Specifically, if the addition of a new solution brings the archive size beyond the SL a process of clustering is applied to reduce the size back to the HL. Archives without limit are also possible in MOO, but the computational cost increases as the number of cost function evaluations grows and are therefore less common.

The first step of RSA is the initialization of the archive: a set of random design vectors are generated, using the upper and lower bound specified. The initial set of random design points has a suggested size of 2-3 times SL, thus, before starting the optimization loop the number of solutions must be brought down to HL. To perform this operation a clustering algorithm, exploiting the reference lines is used. Reference lines are equally spaced lines in the positive quadrant of the objective space that guide the optimization towards better and more uniformly spread solutions; in Figure 9 reference lines are visible in black, for a $M = 2$ case.

Once the archive is initialized the temperature variable is set to 100, the starting temperature, and the optimization loop starts. The flow chart of the process is visible in Figure 7. For every temperature a fixed number of iterations (*iter*, in Figure 7) are performed. Every iteration a random point of the archive is selected and mutated to generate a new solution (called Perturbed Point in the flow chart). The new solution is evaluated using the cost function and then is compared to the rest of the archive: the numbers l and k are computed as the number of points of the archive dominated by the new solution and, the number of points of the archive that dominates the new solution, respectively. l and k are then used to compute the acceptance probability of the new point, which also depends on the temperature variable, as explained before. This step shows how

the archive-to-archive transition approach instead of the point-to-point approach is implemented: a new solution is not considered intrinsically good or bad, instead, by computing l and k the quality new archive as a whole is evaluated. Note that the new archive is the same as before, but with the addition of the new perturbed solution. The acceptance probability p is computed with the following formula, although some other equation may be used:

$$p = e^{\frac{l-k}{\|A\| \cdot T}} \tag{4}$$

Where T is the temperature variable and $\|A\|$ indicates the archive size (*arch_size* in the flow chart). From this equation, it is evident that a new solution has a higher probability of being selected if many points are dominated by it and few points dominate it, while the probability decreases if the archive is already very populated or if the temperature is still high. A typical plot of the rejection probability of a new point is shown in Figure 6, on the y-axis, plotted in function of a value proportional to the number of function evaluations. At the start, most of the new points generated are accepted (exploration phase), but once the archive grows and an increasing number of good solutions are found it become less frequent that a new solution is accepted (exploitation phase), with the final rejection probability flattening between 80% and 90%.

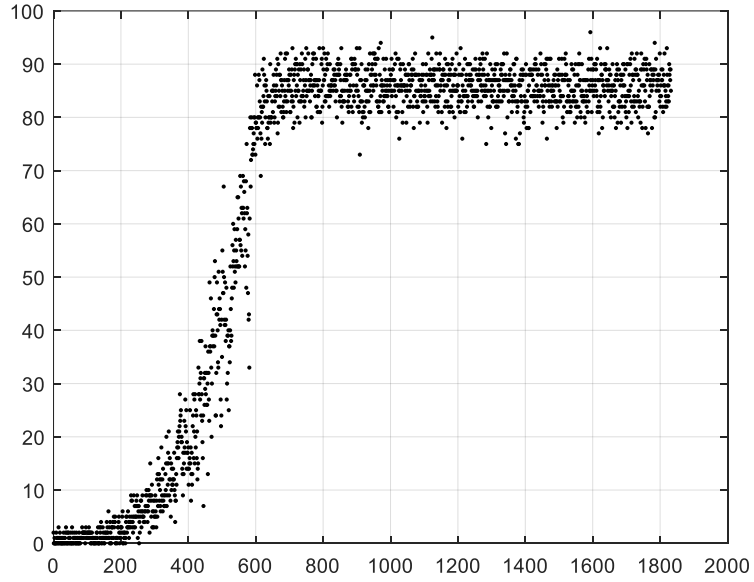


Figure 6: Rejection probability of a new solution generated by RSA versus the total number of function evaluations divided by *iter* (can be thought of as the number of "generations").

The update of the archive follows the logic expressed in the yellow rectangle of Figure 7.

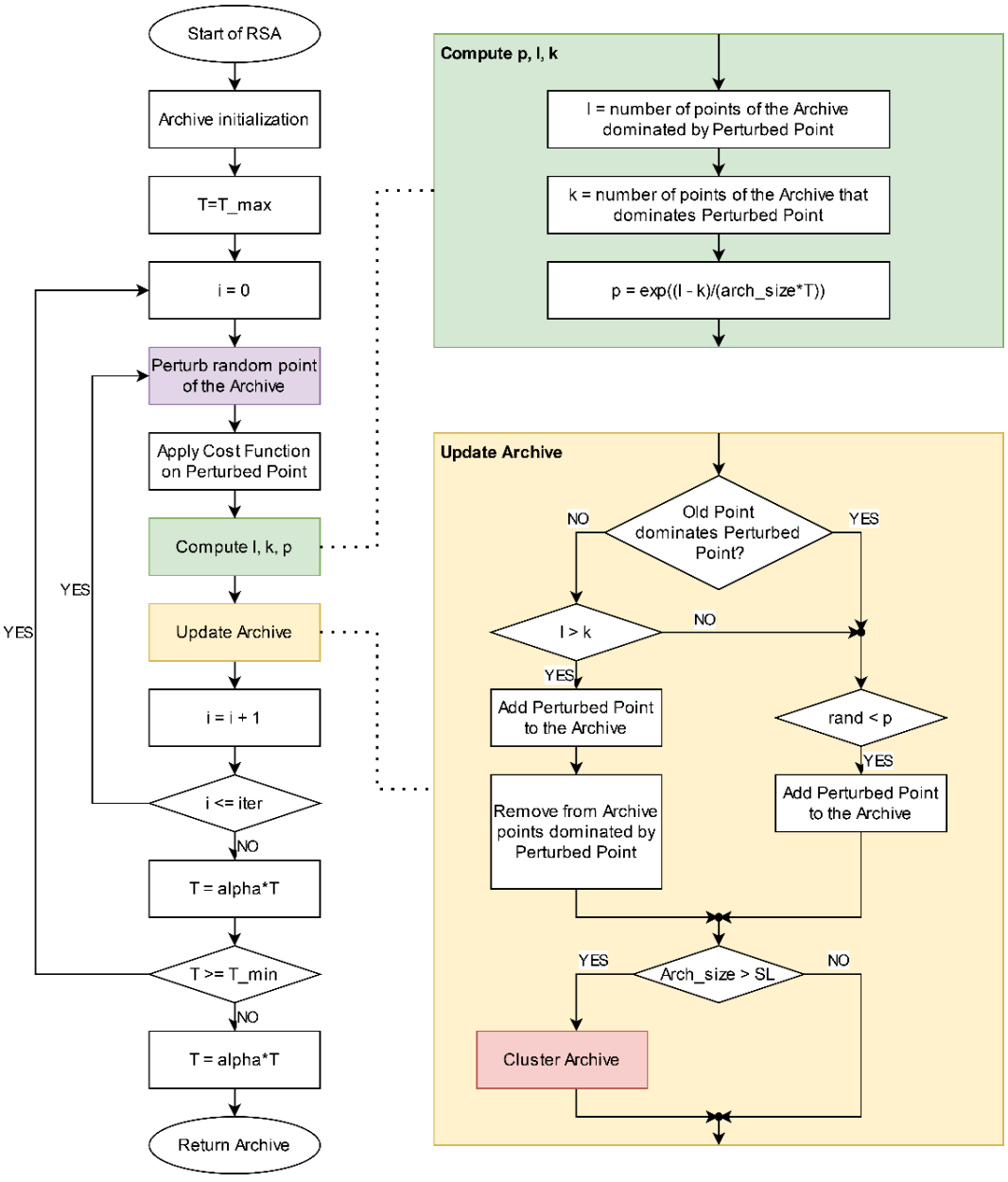


Figure 7: Flow chart of the RSA algorithm for solving MOOPs.

If the old point before perturbation dominates the new point, then the new point is added to the archive with probability p; this is done to allow uphill solutions to be accepted, to avoid trapping the algorithm inside local optima. Otherwise, the new point is again accepted with probability p if $k \leq l$. In the

other case, if $l < k$ and the old point does not dominate the new solution, then the new perturbed point is added to the archive, and every one of the l points which are dominated by it is removed. In any case, if the size of the archive exceeds SL, clustering is applied.

After the iterations of the internal loop are completed, the temperature is updated according to equation (5) and the cycle repeats. The stop condition may be on the temperature, the number of cost function evaluations or some other convergence criteria.

$$T_{new} = T_{old} \cdot \alpha \tag{5}$$

Where α usually is between 0.8 and 0.99 according to scientific literature [11] [10] [9].

In Figure 8 a more detailed presentation of the clustering step is given, in the form of a high-level flow chart. The first step is to normalize the costs of the points of the archive: the normalization scales every cost between 0 and 1, this way the reference lines can be used to select the solution to keep for the next iterations. To compute the normalized archive a M-dimensional hyperplane is constructed, to do so its intercepts with the axes are set as the extreme points of the archive. The extreme points can be found in two different ways: by default, they are computed by minimizing the Achievement Scalarizing Function (ASF), but in case this procedure returns duplicate points or negative values the extreme points are simply computed by finding the maximum value for each cost in the objective space [10] [12]. After the normalization, each point of the archive is assigned to one reference line, the closest to the point, and the number of solutions allocated to each line is saved, as well as the distances between the line and its points.

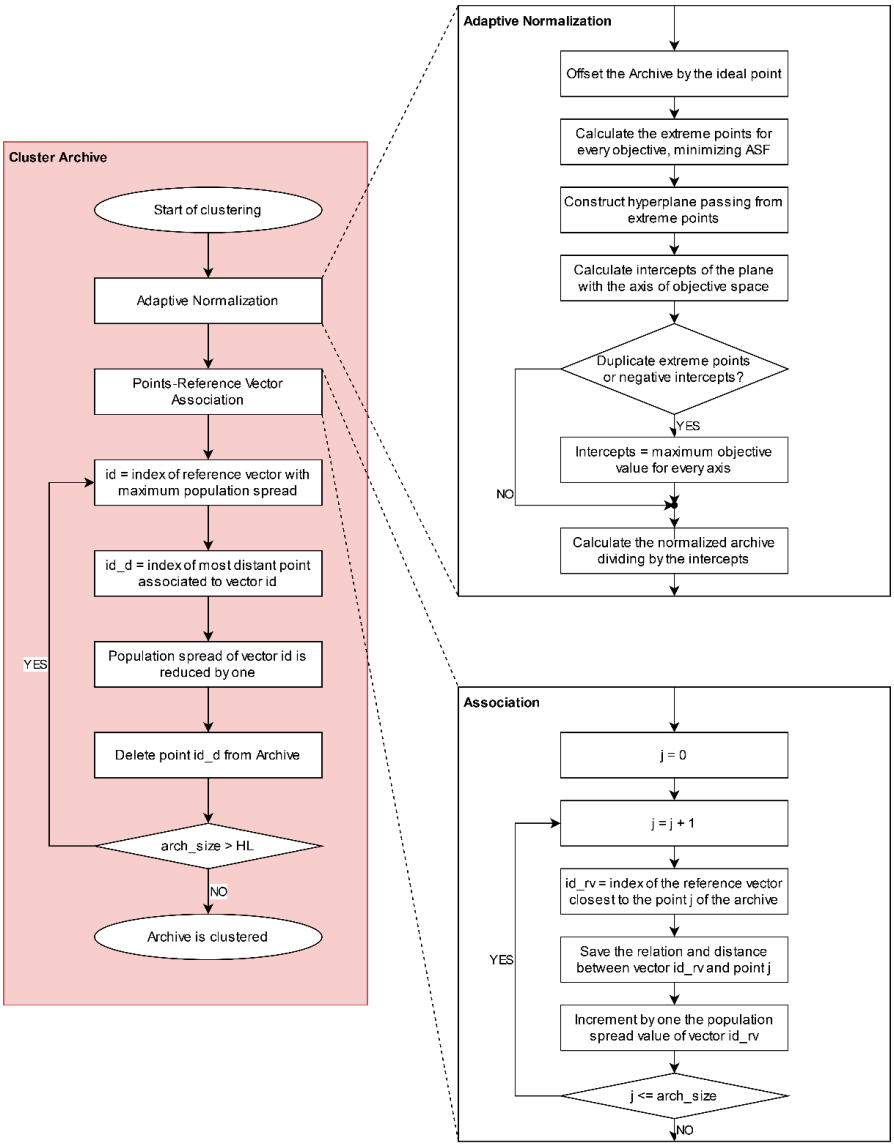


Figure 8: Detailed flow chart of the step of archive clustering of the RSA algorithm.

A loop is then used to remove one by one the solution from the archive until the size decreases from SL or more down to HL. The rationale for deleting solutions is as follows: the reference line with the highest number of points associated is selected and the most distant solution assigned to that line is deleted. This process repeats until enough solutions are removed from the archive, this not only ensures that the final distribution will be equally spread among the reference lines that populate the objective space, but also that only the most promising solutions are kept for the next iterations (the points closer to the lines are, on average, closer to the origin because there the lines are less spaced apart). Figure 9

illustrates the process graphically, in the objective space, for a $M = 2$ case: the black straight lines are the reference lines, the red dots are the solutions. The first plot, on the left, represents the solution distribution before the clustering, the second plot, in the middle, shows the process of identification of the solutions to be removed and on the right the final clustered archive is plotted.

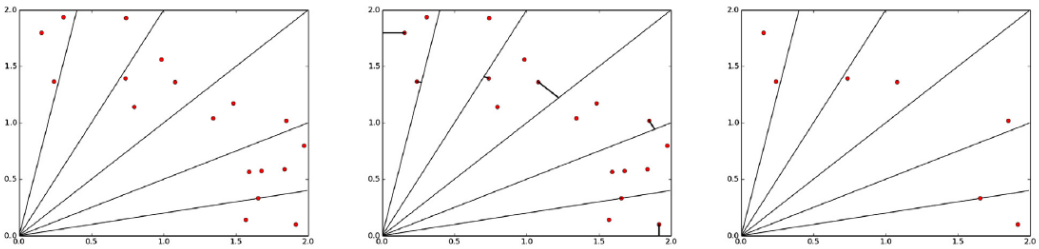


Figure 9: Clustering of solutions in the RSA algorithm [10]

Another detail of RSA is the hybrid mutation strategy, called *mutation switching* by the authors, a flow chart representation of the mutation process can be seen in Figure 10. The algorithm uses an ensemble of 4 different mutation strategies: Simulated Binary Crossover Mutation, Differential Mutation, Polynomial Mutation and Laplacian Mutation [13] [14] [15] [16] [17]. The amalgamation of different mutation strategies improves the effectiveness of RSA, reducing the amount of cost function evaluation required to converge to the optimal solutions and is a key point of its competitiveness among the other MOO algorithms [10]. From the flow chart in Figure 10, it is clear that the behaviour of the mutation switches based on 5 different parameters: SF (switch factor), $flag1$, $flag2$, $prob1$, $prob2$. Changing those parameters modifies the behaviour of the entire RSA algorithm, thus they need to be set properly before starting an optimization. For a generic problem it is suggested setting $SF = 0.75$, $flag1 = 0$, $flag2 = 1$, $prob1 = 0.1$, $prob2 = 0.1$. However, the values can and should be changed to adapt them to the specific problem undertaken: a higher order optimization is possible, different combination of parameters can be tried and used to solve simplified versions of the problem in question to come up with the optimal settings for RSA. This practice is known as *hyper-parameter optimization,* and it is an effective tool to make MOO algorithms extremely versatile and capable of tackling very diverse problems optimally.

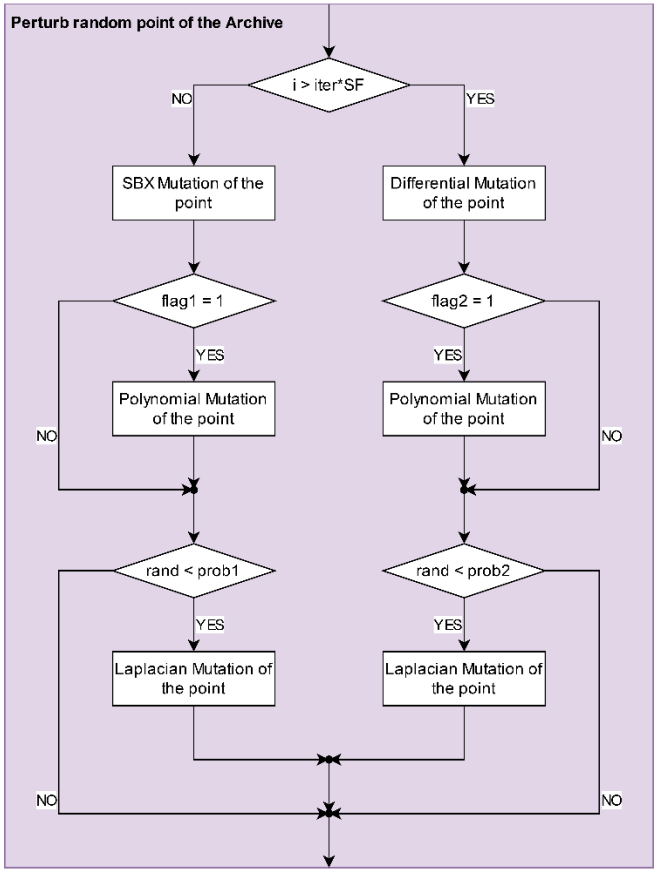


Figure 10: Detailed flow chart on the step of perturbation/mutation of the RSA algorithm.

2.4 Performance Metrics

In this section, a list of the most popular MOO performance metric is presented. The performance metrics are numerical score values assigned to a set of solutions found by an optimization algorithm that judge how well the optimizer performed, under different aspects.

Usually, the metrics are classified by which features they evaluate, the most popular ones are accuracy, spread/distribution, and cardinality [18].

- Accuracy: it is a measure of how well the algorithm converged towards the optimal solutions. It can be seen also how close the set of optimal solutions found by the optimizer is to the theoretical true Pareto front.
- Spread and distribution: closely related, the spread measures the range of values assumed by the solutions, and the extent of the final solution set, while the distribution focuses on the relative distances between each point in the objective space.
- Cardinality: it measures how many solutions are present in the final non-dominated set generated by the optimizer

Another distinction to be made is the classification of unary and n-ary metrics: when comparing multiple algorithms some metrics (unary metrics) require only the knowledge of the set of solutions of the algorithm which is being evaluated, while others require knowing the set of solutions of multiple algorithms to be computed (n-ary metrics).

In the next subsections, a brief description of a selection of the most popular metrics is given, with their definition and classification [19] [20].

2.4.1 Hypervolume Ratio (HR)

The HR is a metric based on the hypervolume (HV) of a set of solutions. HV is defined by the portion of objective space which is dominated by the set of solutions to be measured. The hypervolume requires a reference point in the objective space to define a boundary for its evaluation. Figure 5 illustrates the concept in the case of a 2 objective MOOP.

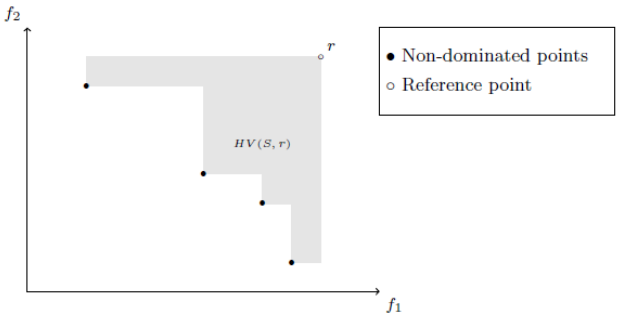


Figure 11: Hypervolume for a 2-objective MOOP [19]

If HV_P is the hypervolume of the true Pareto front and HV_S is the hypervolume of the set of solutions being evaluated, then the hypervolume ratio HR is defined as [19]:

$$HR(S,P) = \frac{HV_P - HV_S}{HV_P} \tag{6}$$

The value of HR is bounded between 0 and 1 and is lower for a set of solutions closer to the true Pareto front, meaning that they form a better approximation of the theoretically optimal solution. To make the comparisons fair, the choice of the reference point must be the same for every set of solutions if multiple algorithms are being compared.

The hypervolume and the other metrics derived from it are by far the most commonly used: one of their advantages is that it measures both convergence and diversity within the same value, however, it is costly to compute. One of the methods used for the calculation uses a Monte Carlo approach, randomly sampling the objective space delimited by the reference points. This method is particularly efficient when the number of dimensions M is larger.

2.4.2 Generational Distance (GD)

The GD is a convergence indicator that measures the distance of the solution from the true Pareto front. Naming S the final set of solutions found by an optimization algorithm and P a discrete representation of the true Pareto front, the GD is defined as [19]:

$$GD(S,P) = \frac{1}{|S|}\left(\sum_{s \in S} \min_{r \in P} \|f(s) - f(r)\|^p\right)^{\frac{1}{p}} \quad (7)$$

Where s represents the members of the set S (the solutions), r are the members of the true Pareto front P, $|S|$ is the number of elements of the set S and p is typically equal to 1 or 2.

The GD is easier to compute compared to the HR, but it is dependent on the number of solutions: since the distance is computed taking a minimum that iterates over the points of the true Pareto front, the sets with fewer solutions that are closer to the Pareto front are favoured and, in the extreme case, if the algorithm finds a single solution on the Pareto front the GD is 0.

2.4.3 Inverted Generational Distance (IGD)

The IGD is very similar to the GD, with the bonus of not being sensitive to the size of the set S and, in general, providing a ranking that intuitively matches more closely the qualities of convergence, spread and distribution [21]. The definition is:

$$IGD(S,P) = \frac{1}{|P|}\left(\sum_{r \in P} \min_{s \in S} \|f(r) - f(s)\|^p\right)^{\frac{1}{p}} \quad (8)$$

Which is the same as the GD, but with the two sets P and S inverted. Like the GD, lower values indicate a higher degree of convergence of S towards P. Other variations of this metric exist in literature.

2.4.4 Maximum Pareto Front Error (MPFE)

MPFE measures the largest minimal distance between elements of the solution set S and their closest neighbours belonging to the Pareto front P [19]. Lower values are better. It is defined as:

$$MP(S,P) = \max_{r \in P} \left(\min_{s \in S} \sum_{i=1}^{M} \|f_i(r) - f_i(s)\|^p \right)^{\frac{1}{p}} \tag{9}$$

Where p is typically equal to 2, and the other symbols have the same meaning as before. This metric suffers from the cardinality of the set S and should be applied to sets having the same size (or similar sizes), to avoid inconsistencies.

2.4.5 Spacing (SP)

SP is a distribution metric that takes into account the distance between a point and its closest neighbour. It is defined according to the following equation:

$$SP(S) = \sqrt{\frac{1}{|S|-1} \sum_{s \in S} \left(\bar{d} - d_i\right)^2} \tag{10}$$

$$\text{With} \quad d_i = \min_{\{s_i, s_j\} \in S,\ s_i \neq s_j} \left\| f(s_i) - f(s_j) \right\|_1$$

Where $\bar{d}$ is the average value of all the distances d_i. The main limitation of this metric is the fact that does not work when the true Pareto front is disconnected. Also, since it does not depend on the true Pareto front set, it does not give information about convergence.

2.4.6 Maximum Spread (MS)

The maximum spread addresses the range of objective function values and takes into account the proximity to the true Pareto front. [20]. It is computed as:

$$MS(S,P) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} \left[\frac{\min\left(\max_{s \in S}\left(f_i(s)\right), \max_{r \in P}\left(f_i(r)\right) \right) - \max\left(\min_{s \in S}\left(f_i(s)\right), \min_{r \in P}\left(f_i(r)\right) \right)}{\max_{r \in P}\left(f_i(r)\right) - \min_{r \in P}\left(f_i(r)\right)} \right]} \tag{11}$$

This metric is intended to evaluate both the spread of the solutions and their closeness to the Pareto front, a higher value of MS reflects that a larger area of P is covered by S.

2.4.7 Pareto Dominance Indicator (NR)

Also called the non-dominated ratio (NR) is a n-ary metric based on cardinality. It is measured as the fraction of solutions belonging to a set S in the set of the overall non-dominated solution provided by all the algorithms. Higher values represent a higher presence of one algorithm on the final non-dominated solutions and are therefore better [20].

Chapter 3

Thermal Design Optimization

3.1 S2T2

The context in which the optimization techniques discussed in the previous chapter will be applied is the development of the MATLAB application “Small Satellite Thermal Toolkit”, S2T2 for short, initially created by D. Calvi, PhD, Politecnico di Torino, Department of Mechanical and Aerospace Engineering. This chapter and the next are intended to be a summary of the work done by the first developer, with a critical analysis of the possibilities and limitations of the software. S2T2 allows assisting the design and development of small spacecraft in the early phases of the project lifecycle through an easy-to-use application with a graphical User Interface (UI) which can perform transient thermal analysis in a simulated on-orbit environment using a Finite Difference Model (FDM) solver. One of the key features of this tool, which differentiates it from other commercial software that performs similar tasks, such as C&R Thermal Desktop, is the seamless integration of a user-friendly TCS design optimizer, which can help guide the design choices of the thermal engineers in their first steps of development of the system.

One of the main parts of the work for this thesis was the review of the work done by the first author of the application, with the intent to expand the potentiality of S2T2 by adding new features, improving the accuracy of the results, optimizing the computations, solving usability inconsistencies and in general enhancing the user experience. These changes, which required radical

modifications and improvements of the source code, have led to the release of a second version of S2T2. Some of these new features will be covered in this thesis, (with a special focus on design optimization) while others are explained in more detail in a parallel thesis, which instead is centred on the upgrade of the geometric aspects of the analysis.

The first version of S2T2 was already capable of performing a full orbital thermal analysis, although with some limitations, and it implemented the following high-level functions:

- Environment definition: in this section, the user can specify the parameters of the central body that the spacecraft orbits, the data of the environmental heat sources, the orbital parameters and the attitude of the satellite. Optionally, two case studies (hot and cold) can be analysed, simultaneously. In Figure 12 an example of filled-in UI is shown.
- GMM definition: this module is used to define and generate the Geometric Math Model (GMM) of the spacecraft, which involves the formulation of the geometric aspects (positions, dimensions, volumes, orientations) of the internal components and subsystems and has the main purpose of creating the FDM nodes and meshes and also computing the radiative view factor of the model. Figure 13 shows the layout of the GMM tab.
- TMM definition: after the input of thermo-physical properties and optical properties of every surface the Thermal Math Model (TMM), which encompasses the conductance coupling between the nodes of the model and the thermal capacitance of the nodes, is created. This section also includes the internal heat dissipation of the subsystems and the thermal link between components. Details are in Figure 14.
- Transient analysis and post-processing: the last step is to gather all the data inputted previously and perform a transient thermal analysis, solving the heat equation in the form of radiative and conductive heat transfers. The UI is designed to aid the user in the visualization of the results: temperature-time plot, orbit visualization and a heatmap are given. Figure 15 and Figure 16 exhibit how the results are organized.
- Design optimization: after the analysis a dedicated module allows to perform a basic multi-objective optimization of some parameters involved in the design of the thermal control system of the satellite studied. This part of the software will be analysed in detail in the next section.

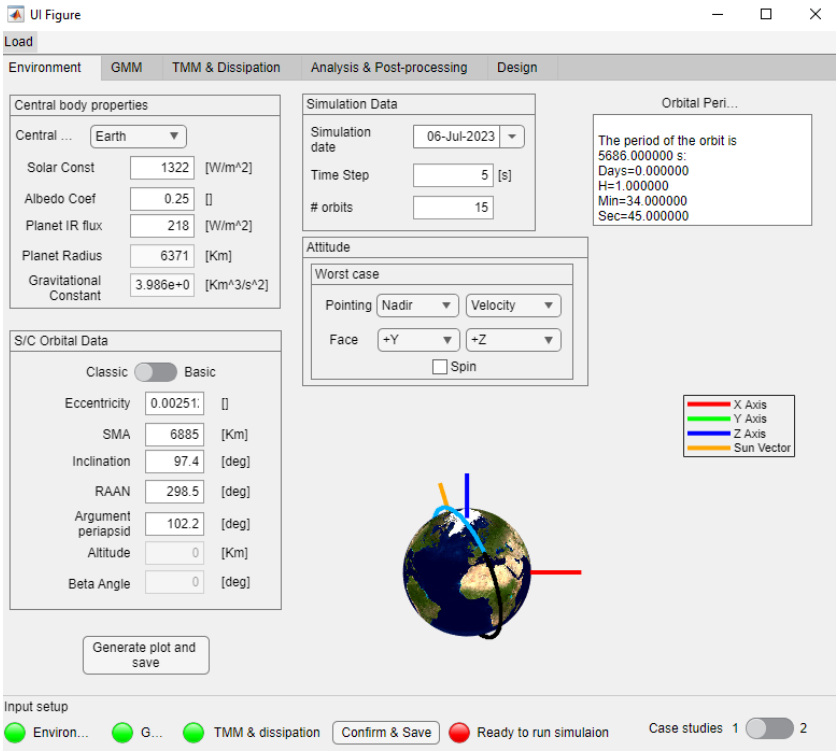


Figure 12: Environment tab of the first version of S2T2

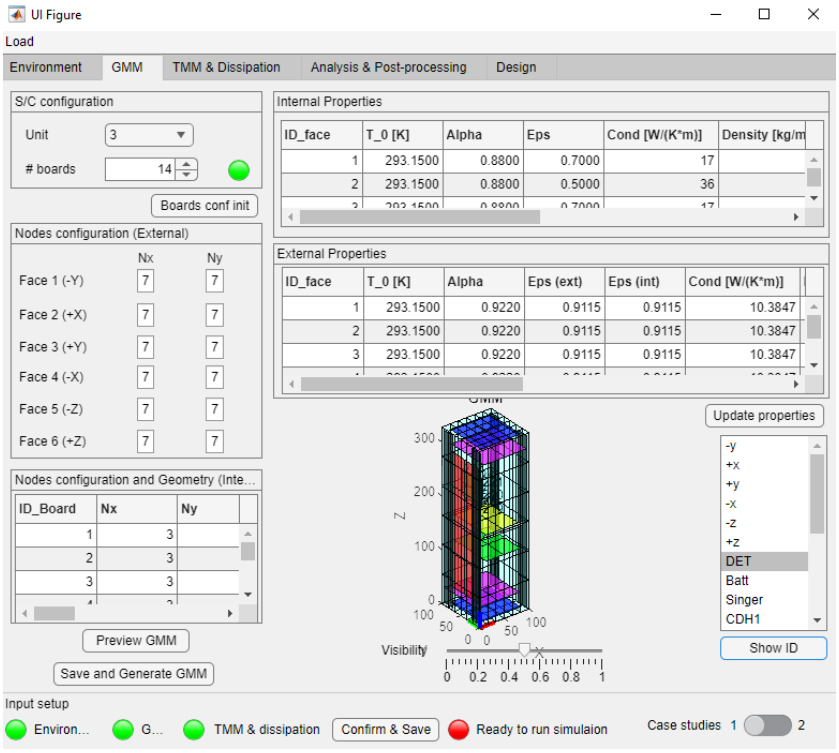


Figure 13: GMM tab of the first version of S2T2

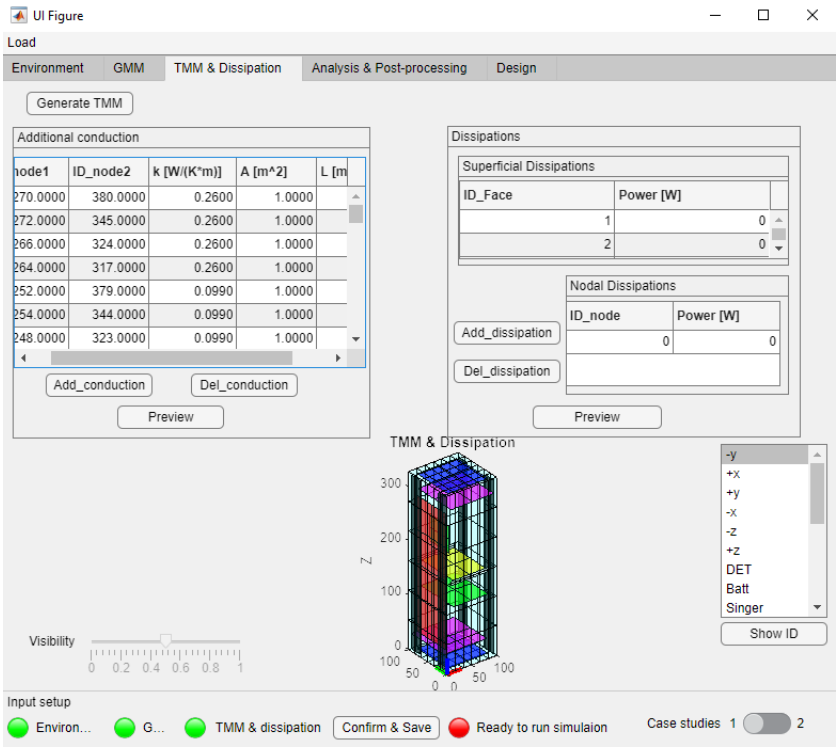


Figure 14: TMM tab of the first version of S2T2

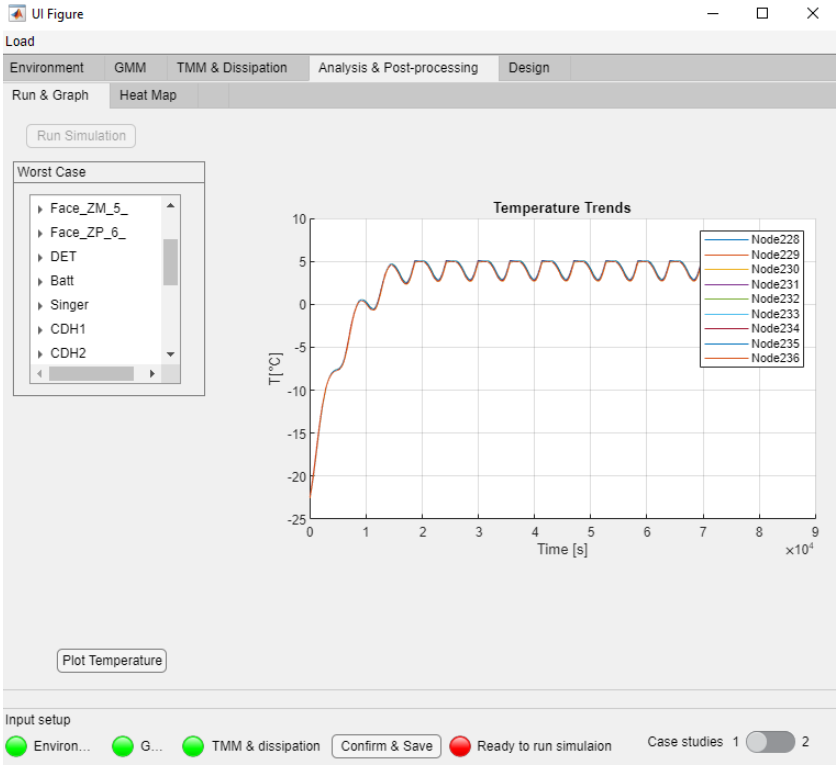


Figure 15: Analysis tab of the first version of S2T2.

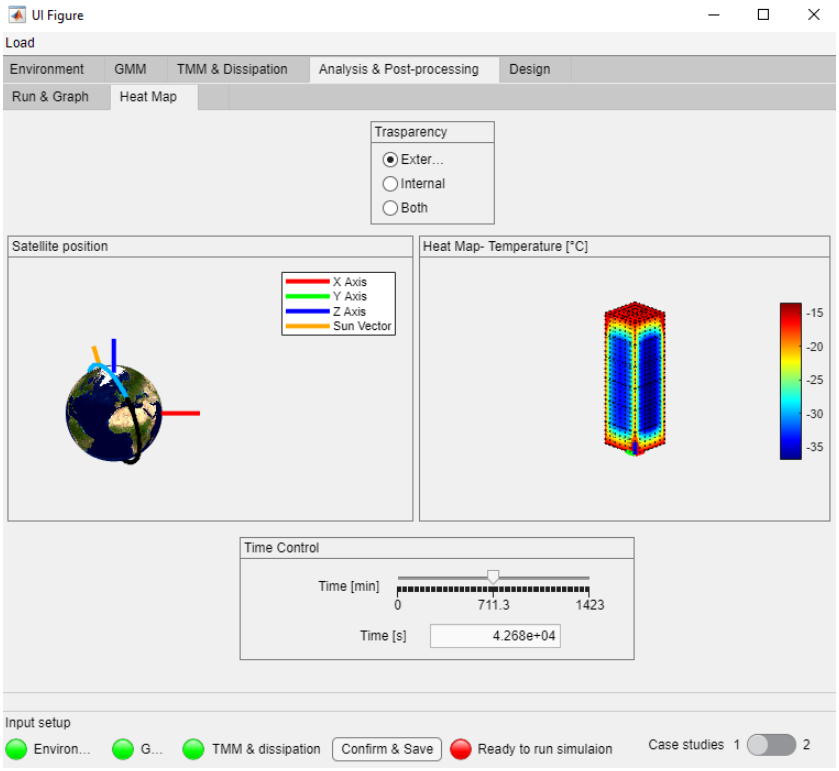


Figure 16: Post-processing tab of the first version of S2T2.

The scope of this work is to improve on the existing features and expand the possibilities of the software, focusing on the design aspect. The work was carried out in parallel, with close cooperation and contact with the work done in another thesis, by Francesco Lucia: “Develop of a Tool for Thermal Analysis of Small Satellites” which is considered the complementary part of this document, regarding the work done in the transition from the first to the second release of S2T2.

3.2 Thermal Design in the first release of S2T2

One of the key aspects of S2T2 is the possibility of performing design optimization studies introducing different design elements typical of small satellites, such as CubeSats or higher form factors. In this section an overview of the current UI, features, and implementation details of the Design tab of the first version of S2T2 is given, followed by an investigation of the main limitations. Then, in the following section, the work done to improve this part of the software is presented, with some first basic performance comparison and an in-depth explanation of the new features and the programming techniques used to improve the source code, along with an explanation of the main drivers behind these choices.

The first version of S2T2 presented a simple Design tab in which three different design elements could be optimized: optical properties of the structure of the satellite, heaters for the internal components and thermal straps between internal components and the structure.

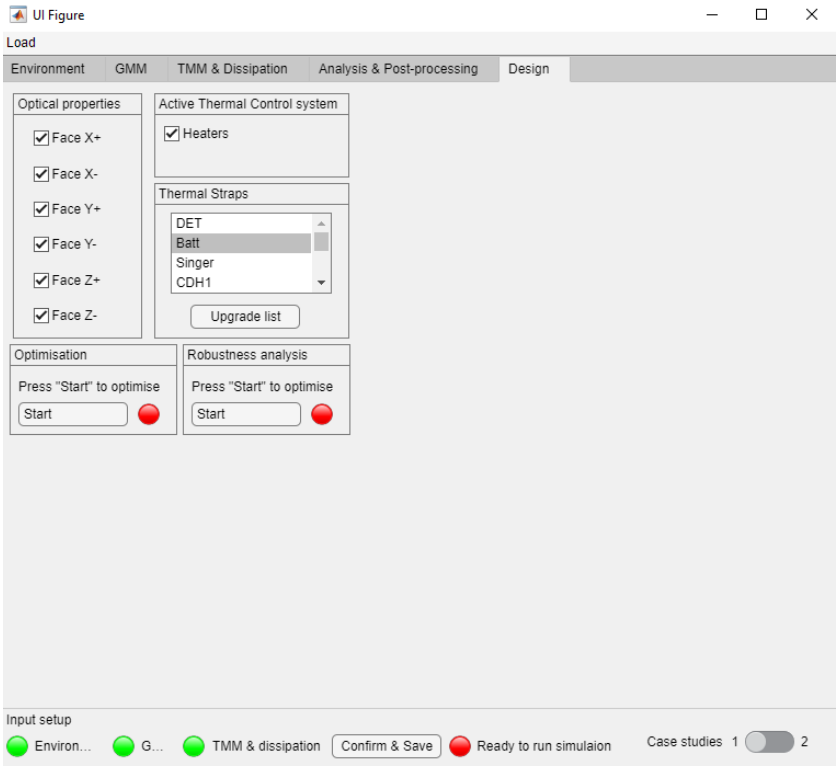


Figure 17: Design tab (input) of the first version of S2T2

3.2.1 Optical Properties

Looking at Figure 17, on the optical properties panel on the left, every face of the structure can be selected using a checkbox to determine which face will have its optical properties changed during the optimization. The choice of limiting the optimization only to the structure of the satellite (which in S2T2 is approximated with a box with six faces made out of thin shell elements) is sensible because, especially during the first phases of the design of a small spacecraft, there is a high degree of freedom in the choice of structure surface finishes, both in term of configuration considerations (e.g. solar cells placements, radiators position, etc.) and also under the aspect of the application of paint, coatings or tape with specific optical properties [22]. Furthermore, the structure in the first version of S2T2 is the only component exposed to the outer environment, and thus the only component that experiences environmental heating, which, in turn, highly depend on surface finishes and optical properties.

The optical properties of the structure surfaces which are optimized are α_{ext}, ϵ_{ext}, ϵ_{int}, solar absorptivity of the external side, IR emissivity of the external and internal side, respectively. If all of them are optimized simultaneously they sum up to $N_{OPT} = 18$ design variables (3 for every structure face).

The lower bound and upper bound of these variables is set to their physical range of possibilities, spanning from 0 to 1, however, it must be kept in mind that 0 and 1 are extreme values, which are only possible in an ideal case, while in real-world scenarios the optical properties always fall in between of these two extremes without reaching them.

3.2.2 Heaters

To the right of the optical properties panel, one or two checkboxes for the optimization of heaters are located. In the case of a single case study analysis the only option is “Heaters”, otherwise with two case studies “Heater Hot Case” and “Heater Cold Case” can be independently selected if in either case heaters are unnecessary.

In this first version of S2T2 when these fields are checked the optimizer assigns an average heater power for every internal component of the satellite. These average powers are treated analogously as the superficial dissipation encountered during the definition of the TMM & Dissipation tab and are summed

to the pre-defined dissipation of every component. Thus, when only one field is selected N_H design variables are considered, with N_H equal to the number of internal items, and when both fields are checked $2N_H$ variables are added instead.

The bounds for the heater power of every component are set to 0 W and 2.5 W and are not editable.

3.2.3 Thermal Straps

Lastly, under the heater panel, visible in Figure 17, the thermal strap panel is visible. Here the components where the thermal straps are placed and optimized can be selected, confirming the choice by clicking on "Upgrade list". During the optimization, for each of the items selected a thermal strap will be placed, connecting the component to the structure. The thermal straps are treated as node-to-node conductions, analogous to the additional conduction defined in the TMM & Dissipation tab.

Every thermal strap has a starting node, which belongs to the set of nodes of the starting items, and an ending node, which is always located on the structure of the satellite. Additionally, the cross-sectional area of the straps A_L is optimized, while the thermal conductivity k_L is fixed at 385 W/m/K (thermal conductivity of copper) and the length of the strap L_L is computed as the Euclidean distance from the starting node to the ending node. For every one of the total N_{item_L} item selected in this panel there will be 3 variables to optimize: starting node, ending node and area, bringing the total variable to $N_L = 3 \cdot N_{item_L}$.

While the lower and upper bounds of A_L are easily definable and are set as 0 mm^2 and 100 mm^2 respectively, the starting and ending node requires special care. Differently from every other design variable, the nodes where the thermal straps are connected are not continuous variables: this poses a significant problem for most optimization algorithms which are meant to work with real numbers. The nodes in S2T2 are enumerated and discerned by a unique integer identifier (ID), those IDs go from 1 up to the total number of nodes, without skips, and are defined in a way that every item has its own set of IDs, in increasing order, again, without skips, with the structure item being always the first.

This regularity can be exploited by the optimization algorithm to set the lower and upper bounds conveniently, however, some modifications to most general-purpose optimization algorithms are still required to allow to correctly manage

those integer decision variables. One other possible solution that does not require modification of the algorithms could be to optimize the starting and ending node of the straps as if they were continuous variables, maintaining the same bounds, and then rounding to the nearest integer to get the ID of the node. This, however, poses a significant problem: the property of continuity of the optimization is not preserved. Ideally, a high level of smoothness of the objective function is desirable, as it implies a more predictable and stable optimization landscape, making it easier to locate and navigate towards optimal solutions. The node numbering is not performed using this concept, thus a small variation of the node ID, say from Node 17 to Node 18 of a generic model, could mean a high variation of the results in case the two nodes are not near each other in the model. Moreover, even if in the majority of cases the nodes with subsequent IDs were near each the opposite may not be true: in Figure 18 it can be seen that while nodes 60, 61 and 62 are near each other and have subsequent IDs that cannot be said for nodes 58, 61 and 64, which although are adjacent to each other in the model do not have subsequent IDs.

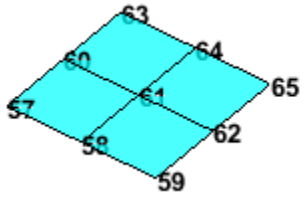


Figure 18: Subsequent IDs do not mean adjacent nodes and adjacent nodes do not mean subsequent IDs.

For these reasons in the first version of S2T2 it was favoured the approach of modifying the optimization algorithm to adapt it to this specific need.

3.2.4 Optimization Algorithm and Post-Processing Sequence

In the first version of the software, the optimization was performed using the MATLAB function *gamultiobj*, a genetic evolutionary multi-objective optimization algorithm based on NGSA-II. *gamultiobj* was utilized in conjunction with custom developer-defined functions for the creation of the initial population, the mating/reproduction process and the mutation of the individuals. This is mainly because continuous and discrete (integer) design variables co-exist in the same optimization process.

Before the optimization, all the inputs of the Design tab are gathered and the lower and upper bounds for the chosen optimization variable are defined. Then

the computation follows the nominal flow used for the transient analysis of the satellite: the attitude is computed, and based on that, the view factors between every face and the Sun and the central body planet are generated, considering the self-shadowing of the satellite. The pre-processing is stopped before computing the environmental heat sources because the heat from the environment depends on the optical properties of the structure which are optimization variables.

Then *gamultiobj* starts, with parameters pre-determined and not editable: the population is set at 10 times the number of design variables $N_{OPT} + N_H + N_L$; the crossover fraction, which is the fraction of the population generated by the reproduction process, is fixated at 0.85; the Pareto fraction, which is the fraction of individuals to keep on the first Pareto front is set at 0.6 and the total number of generations is set at 9 instead. From these parameters, depending on the model, around 2000 and 5000 cost function evaluations are required to meet the stopping criteria of the algorithm.

After the optimizer ends the post-processing starts: *gamultiobj* returns the final population of the first Pareto front with their design variables and relative scores. The number of objectives used by the optimizer may vary from 1 to 4 in the first version of the software and they depend on the cost function and on the variables to optimize. A summary of them is visible below in a graph format, in Figure 19.

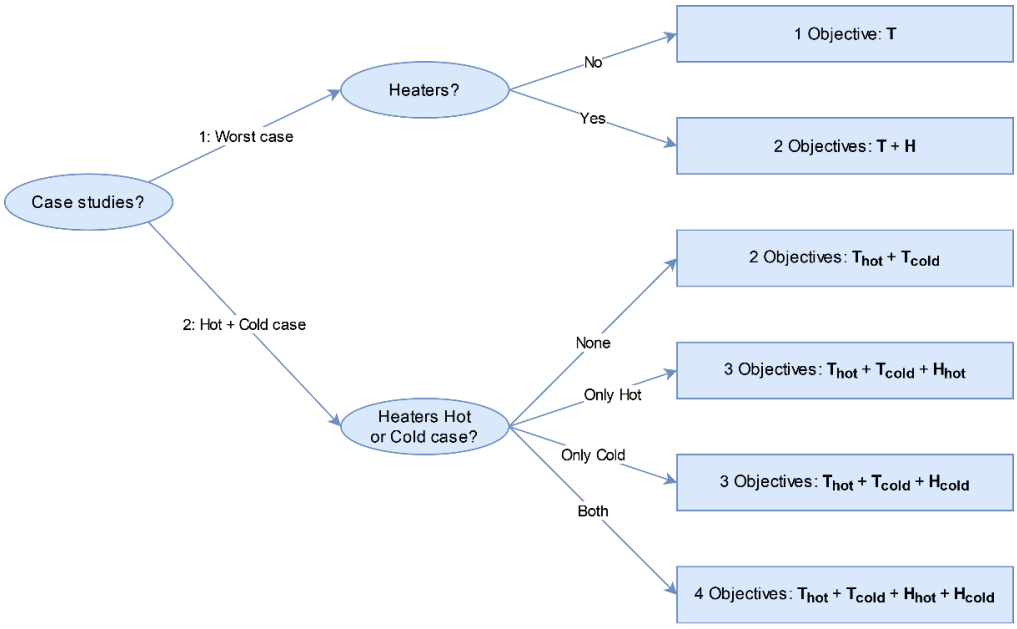


Figure 19: Summary of the number of objectives considered for the optimization in the first version of S2T2.

From the output of *gamultiobj* the values of the optical properties, the heater powers, and the thermal strap data are extracted and organized in suitable data structures. For every solution then a complete transient analysis is performed. After that a decision-maker algorithm assign a comprehensive score to every solution, the utility score, taking into account how large the margin between the extreme temperature (minimum and maximum) of every node and its operating temperatures are, both in the cold and hot case. Each component of the utility metric is averaged using a uniform, not editable set of weights. In more detail: the utility value U is computed starting from the minimum temperature gap G_d for the cold/hot case and low/high temperatures:

$$G_{d_{low}}^{Hot} = \min\left(\left|T_{min}^{C,Hot} - T_{min}^{O}\right|\right) \tag{12}$$

$$G_{d_{high}}^{Hot} = \min\left(\left|T_{max}^{C,Hot} - T_{max}^{O}\right|\right) \tag{13}$$

If there is only one case study for the optimization, the

$$G_{d_{low}}^{Cold} = \min\left(\left|T_{min}^{C,Cold} - T_{min}^{O}\right|\right) \tag{14}$$

$$G_{d_{high}}^{Cold} = \min\left(\left|T_{max}^{C,Cold} - T_{max}^{O}\right|\right) \tag{15}$$

Where T^C are the computed temperatures of every node, in the hot or cold case, and the subscripts min and max indicate the maximum and minimum values concerning time. T^O indicates the operative temperatures of every node, and the subscripts min and max indicate if the value derives from a minimum operative temperature requirement or a maximum operative temperature, respectively.

Linear scaling functions f are used to normalize the value of every solution in the following way (note that if heaters are part of the optimization Equation (20) and (21) are part of the computation depending on the case studies in which heaters are active, otherwise not; in a similar fashion Equation (18) and (19) are only used when both case studies are active):

$$f_{low}^{Hot}: \left[0, \max\left(G_{d_{low}}^{Hot}\right)\right] \rightarrow [0,1] \tag{16}$$

$$f_{high}^{Hot}: \left[0, \max\left(G_{d_{high}}^{Hot}\right)\right] \rightarrow [0,1] \tag{17}$$

$$f_{low}^{Cold}: \left[0, \max\left(G_{d_{low}}^{Cold}\right)\right] \rightarrow [0,1] \tag{18}$$

$$f_{high}^{Cold}: \left[0, \max\left(G_{d_{high}}^{Cold}\right)\right] \rightarrow [0,1] \tag{19}$$

$$f_{H}^{Hot}: [\min(P_{H}^{Hot}), \max(P_{H}^{Hot})] \rightarrow [1,0] \tag{20}$$

$$f_{H}^{Cold}: \left[\min\left(P_{H}^{Cold}\right), \max\left(P_{H}^{Cold}\right)\right] \rightarrow [1,0] \tag{21}$$

Where the max() and min() functions are applied by iterating over the different design solution found by the optimizer and P_H is the total power dissipated by the heaters, in the hot or cold case, for every design solution. Denoting with W_i the weight of the i-th component of U, for every solution it is possible to compute the final utility value U:

$$\begin{aligned} U = [&W_1 \cdot f_{low}^{Hot}\left(G_{d_{low}}^{Hot}\right) + W_2 \cdot f_{high}^{Hot}\left(G_{d_{high}}^{Hot}\right) + W_3 \cdot f_{low}^{Cold}\left(G_{d_{low}}^{Cold}\right) + W_4 \cdot f_{high}^{Cold}\left(G_{d_{high}}^{Cold}\right) + \\ &+ W_5 \cdot f_{H}^{Hot}(P_{H}^{Hot}) + W_6 \cdot f_{H}^{Cold}\left(P_{H}^{Cold}\right)]/(W_1 + W_2 + W_3 + W_4 + W_5 + W_6) \end{aligned} \tag{22}$$

This approach is called the *weighted sum method* and it is one of the simplest ways to convert the results of a multi-objective optimization down to a single objective. The main drawback, as already mentioned in Chapter 2, is that the optimal weights are not known beforehand and may require the intervention of an expert user/developer to be set correctly. In the first version of S2T2, the weights are all fixated at 1.

This completes the optimization and a plot showing the utility values in function of the ID of every solution is generated, as is shown in Figure 20.

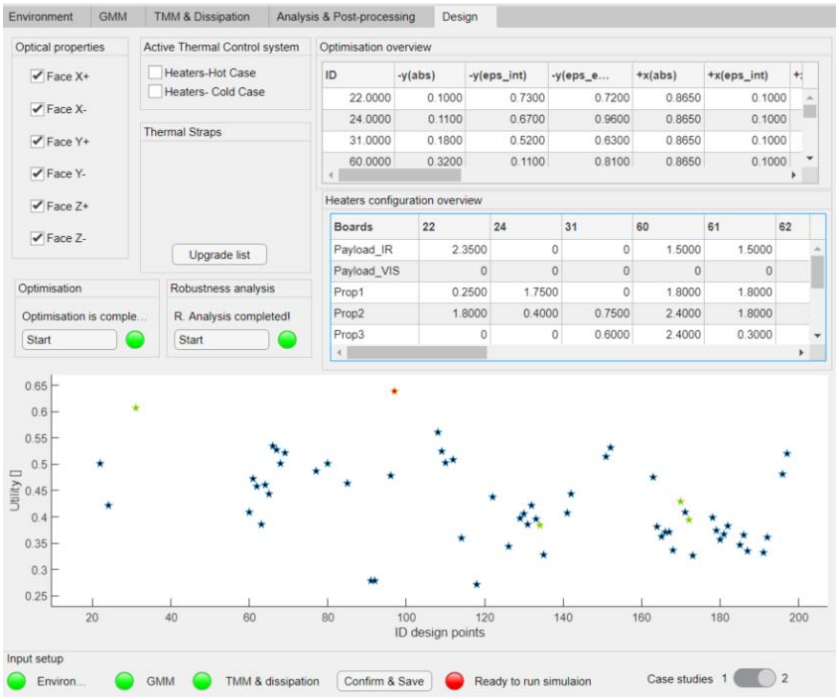


Figure 20: Design tab (input and output) of the first version of S2T2

After the optimization a Robustness analysis can be performed, to isolate the solutions less vulnerable to fluctuations of their design variables. The approach used in the first version of S2T2 is to apply 10 times random perturbations in the positive or negative sense to every design vector, computing again the transient analysis and the value of U each time. For every solution the standard deviation of U is used to measure robustness: low fluctuations of the result mean that the solution is less affected by uncertainty, errors, or degradation of component performances. The maximum intensity of the perturbation is set to 25% of the value of the variable being perturbed. Graphically, the 5 most robust solutions are highlighted in green in the utility plot visible at the bottom of Figure 20.

3.2.5 Main Limitations of the first version of S2T2

Despite an already quite advanced pre-existing infrastructure, several limitations are evident from the previous paragraphs, the most important ones are listed below in a 13-points bullet list:

(1) Every variable to be optimized has the severe problem of having fixated lower and upper bounds, not editable by the user without having access to the source code. One of the key scopes of S2T2 is to offer a simple and user-friendly experience, however, if the bounds are not modifiable in the application and even

not explicitly shown up until the end of the optimization this compromises seriously the range of possible applications of the tool. Moreover, while the bounds hard coded in the first version were suitable for CubeSats or nanosatellites, removing this limitation could make S2T2 scale far better when larger satellites are considered, such as Small Sats. Indeed, one of the directions where the efforts of this thesis and the one complementary to this were focused, was the extension of the use cases of S2T2, transitioning from a CubeSat-centred approach to a wider reach, with Small Sats in mind, going in the direction of closing the gap with other thermal analysis commercial software.

(2) Optical properties of the structure cannot be optimized independently. This is especially undesirable considering that internal and external properties cannot be disconnected in this first version of S2T2. While deciding α_{ext} and ϵ_{ext} at the same for a face of the structure can usually be a realistic use case, not the same can be said about internal ed external properties: many times, the requirements for internal and external surface finishes are different and the final design choice may not be done simultaneously, hence the need of a more customizable optimization of α_{ext}, ϵ_{ext} and ϵ_{int}.

(3) Heaters are affected by a similar problem: during a run of optimization, not every internal component may be desired to be optimized with the addition of heaters. Most of the time after performing the canonical hot case and cold case analysis the thermal engineer already has an idea of which components are too cold and may need active thermal control equipment. If heaters could be independently applied only to the component deemed appropriate by the user, this not only would speed up the computation, because it lowers the number of design variables N_H, but also it would make each solution tried by the optimization algorithm more meaningful, accelerating convergence of the first Pareto front towards better solutions.

(4) The problem of the variables indicating the starting and ending node of thermal straps was already assessed in a previous sub-section: the presence of continuous variables in a design vector composed in almost its entirety by continuous variables poses challenges for the optimization algorithms: the ones that could deal with this discrepancy often require some modification and, regardless of this, the desirable property of continuity of the cost function cannot be assured using the node IDs as optimization variables. Other representations of the starting and ending point of the thermal strap in the design vector may help to overcome this limitation.

(5) As a minor detail, the computation of the length of thermal straps using Euclidean distance is not very representative of the real-world situation: it is frequent that, for weight and volume constraints, inside a spacecraft, the space available is for the large part occupied by subsystems, with a low percentage of empty volume. Because of this, it is not always reasonable to imagine a direct straight-line connection for thermal straps between any two nodes of the model, and other more conservative distances could be used instead.

(6) Thermal straps in general, in the first version of S2T2 can only be applied from an internal item to the structure: while this in most use cases is acceptable since internal heat-dissipating components often require to be linked with radiating surfaces located on the structure, in other specific scenarios it may be useful to consider creating item-to-item thermal connections. A typical situation where this could be convenient is in a cold satellite: the heat produced on board by some component could be used to increase the temperature of other cold items with lower or no dissipation, redirecting heat inside of the satellite using an item-to-item thermal strap, instead of transferring heat to the structure to radiate it away directly.

(7) The choice of *gamultiobj* for the optimization algorithms, although easy to implement, may not be the best performing one. In fact, many other MOO algorithms exist, like the one presented in Chapter 2, that implement more advanced concepts to converge towards optimal solutions (e.g. reference vectors, exploration/exploitation balances, hybridization, decomposition, mutation switching, usage of repositories/archives) and that may be more suitable, especially the ones designed for many-objective optimization.

(8) The optimization process, post-processing and robustness analysis are quite slow, computationally speaking, due to several inefficiencies in the source code and the lack of a process that fully exploits the simplifying assumptions of S2T2: mainly the hypothesis of constant orbit during optimization and the fact that the transient analysis is not performed during the optimization loop (only steady state temperatures are computed, to cut time and allow a vastly higher number of design points to be explored). High computational cost means high computational time, thereby making the experience more difficult for the user, which cannot see the result of his analysis in the same work session and must wait a time in the order of hours before making modifications or accepting the results given by the software.

(9) The settings of the optimization are not customizable, meaning that the user cannot specify any stopping criteria or any other behaviour of the MOO algorithm or the post-processing sequence. This is particularly important in correlation with the previous point: giving the possibility to set an upper limit to the computational time is crucial to allow the user to experiment with the optimizer and find the combination of design variables and optimization parameters better suited for the problem, before starting a large-scale optimization. Giving more control to the user also implies that more specific cases can be treated, and this extends the usage possibilities of the application.

(10) Similarly to the optimization, the robustness analysis also lacks basic settings to control the computational time of the process and the behaviour of the algorithm. Same considerations as the previous point applies.

(11) Another limitation of the robustness analysis is that it does not show a clear quantitative metric for every solution to the user. This limits the possibility to conduct sufficiently informed trade-off analysis at the moment of selection of the final solution among the ones available.

(12) There is a lack of feedback to the user about the status of the optimization and robustness analysis processes. Without having an estimate of the completion percentage of the process it is more difficult to perform multiple analyses because the user has no information of the remaining time. Also, visual cues that inform that the analysis is running and tell which part of the process is being conducted, facilitate the investigation for potential corrective actions in case the results or the process is not deemed satisfying by the user and prevent inadvertently starting multiple queued routines.

(13) In the context of further development and future releases, the readability of the code and lack of comments is also an issue that could be easily improved upon. This is particularly important when the developers of different releases do not coincide with the same person, in these cases a forward-thinking approach to writing code is strongly advised, to cut development time and lengthen the lifespan of the application. Sometimes it is useful to make assumptions and simplifying hypotheses about the models and the data structures to reduce development time, however, there is a balance to be struck to prevent the accumulation of technical debt, which, in the most extreme cases, can lead to a stop of software developments [23].

3.3 Improvements in S2T2 Thermal Design

In the following paragraphs the main improvement to the Design tab of S2T2 are presented, addressing each one of the limitations mentioned in the previous section and explaining the rationale behind the choices. The improvements regard both performance optimizations of the code and the addition of new features to the pre-existing structure.

Performance optimization was deemed the most pressing issue, not only from the user perspective but also for the work of the developer. The goal of delivering a more efficient release of S2T2 is heavily dependent on the possibility of performing development incrementally, testing new features one at a time as they are added, and solving issues the moment that they are encountered. The approach to software development followed for this work focuses on the Unit Test of single isolated sections of code subject to changes (in the case of this thesis: optimization algorithms, cost function, optimization post-processing and robustness analysis), exploiting as much as possible the philosophy of Test-Driven Development, for testing the effectiveness of modifications and the compatibility with the rest of the source code [23]. It is clear from this approach centred on recurrent testing that improving computational performance is the first step to building a robust and efficient development structure, that can be later used to add new features far more easily. For these reasons, the issue of computational cost had to be tackled early on and was the first major part of the work presented in this thesis.

To understand how it is possible to improve the code of the Design tab in the first version of S2T2 a preliminary analysis of the legacy process was performed. In Figure 21 a complete flow chart of the process, from start to end is shown.

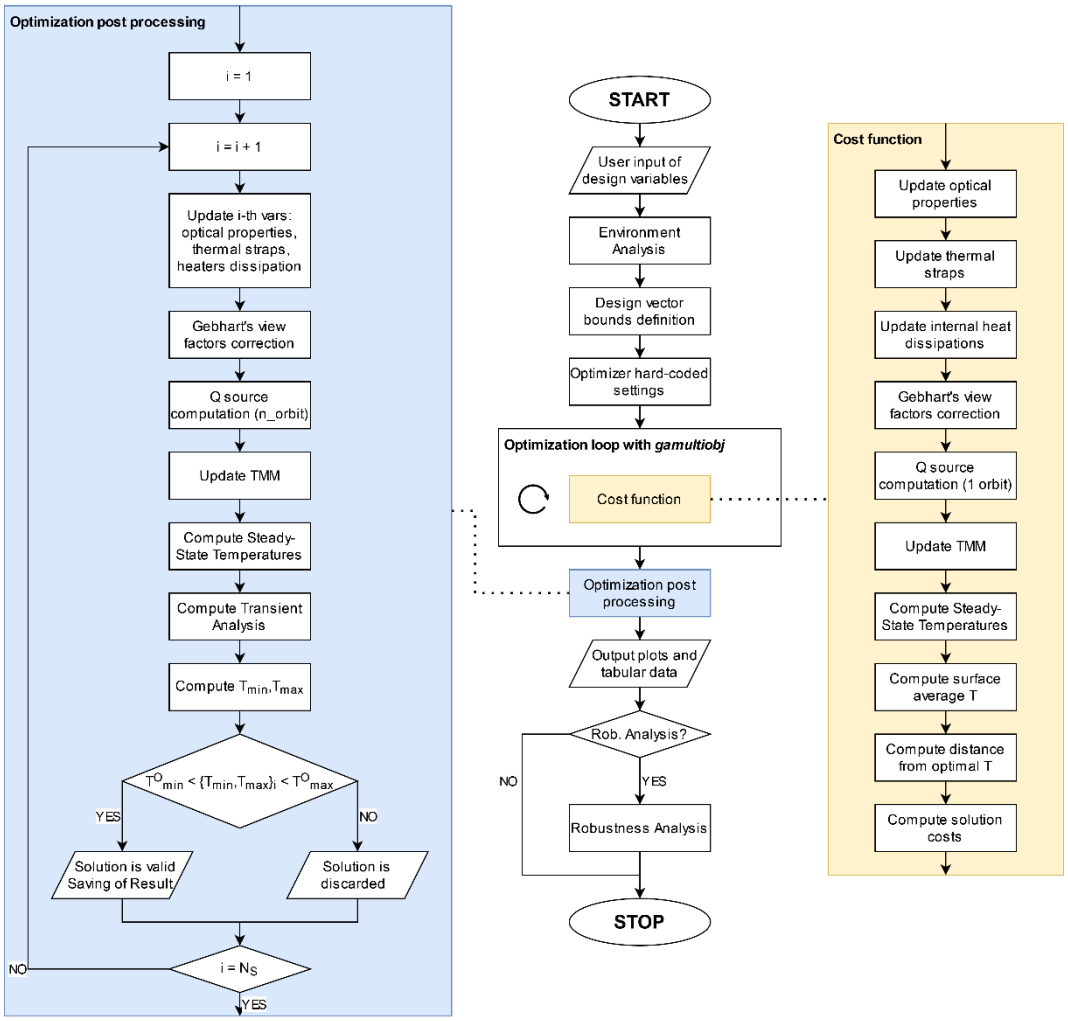


Figure 21: Flow chart of the process followed in the Design tab of the first version of S2T2.

After the definition of the design variables using the UI, the environment analysis function is called: this module computes the trajectory of the satellite for one orbit and derives the view factors between each portion of the satellite exposed to space, the Sun and Earth (or central body of the orbit), also considering eclipse time and self-shadowing (for simple box-like geometries with no holes and no external appendages, the faces of the structure are assimilated to one-sided planar emitters; they are the only surfaces exposed to space and their view factor with space F_{space} can be assumed to be $F_{space} = 1 - F_p$, where F_p is the Earth view factor) [24]. After that, the lower and upper bounds of the design variables are defined as two vectors with length $N_{OPT} + N_L + N_H$, the same as the design vector. Then the optimizer starts its loop, in which different design vectors are generated and evaluated using the cost function. The cost function outputs a

set of values corresponding to the costs of the design vector for each of the M objectives, as already illustrated in Figure 19. The cost function is the pivotal part of the process, and the overall computational efficiency largely depends on it.

3.3.1 Cost Function in S2T2

The cost function in S2T2 represents the simulation environment in which each solution must be tested to attribute its costs. The portion of physical simulation corresponds only to the computation of steady-state temperatures; however, the other computational steps of the cost function are required to prepare all the data needed to complete this step. From Figure 21, it can be seen that when a new design vector is generated the first part of the process to arrive at the evaluation of its costs is to deconstruct the design vector x in its constituent components (visible in Figure 22) and update the satellite data with the data present in x. The update process is done by overwriting the design variables inside a copy of the original data structure of the satellite (the *sat* structure).

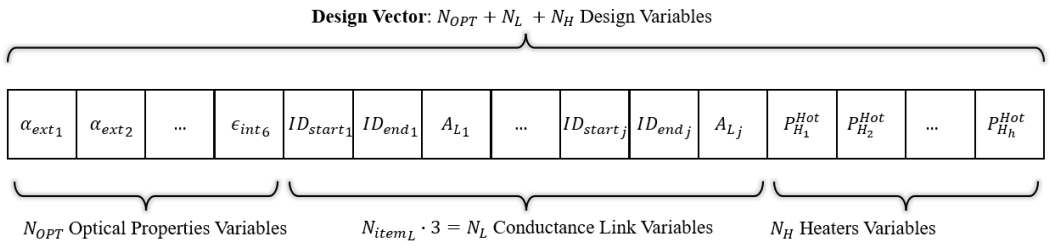


Figure 22: Structure of the design vector x in the first version of S2T2.

The first N_{OPT} variables of x are extracted and the new optical properties of the external faces of the satellite are saved in the *sat* structure. This step has a computational complexity of $O(N_N \cdot N_{F/N})$, where N_N is the total number of nodes of the model and $N_{F/N}$ is the number of surfaces that a node of model touches and typically varies from 1 to 3 in this first version of S2T2. As shown in Figure 23, for board-like and box-like items the nodes can be divided into central nodes (yellow square), edge nodes (green tilted square), or vertex nodes (red circle).

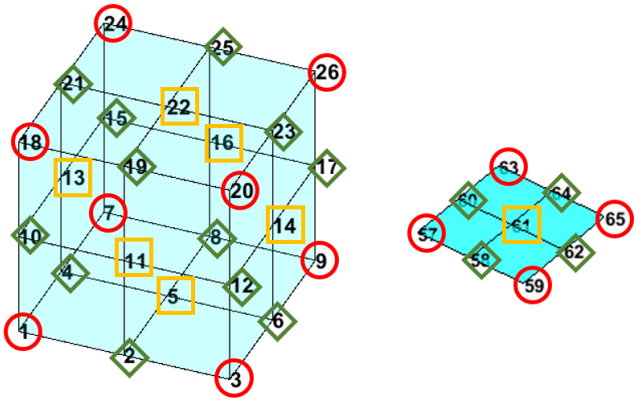


Figure 23: Central nodes (yellow square), edge nodes (green tilted square), and vertex node (red circle) on two different item geometries.

It is easy to see that while central nodes only belong to 1 surface, edge nodes and vertex nodes can, in the case of box-like geometries belong to 2 and 3 different surfaces, respectively. The portion of surface related to a node will be called a surface element, or simply an element. The same two geometries of Figure 23 are drawn again in Figure 24, highlighting this concept: the red lines are the border of each element. Note that while the board on the right has 9 nodes and 9 elements, the box geometry on the left, although having 26 nodes, has $9 \cdot 6 = 54$ elements, this is because thinking of the box as a union of 6 boards the coincident nodes must be merged into a single one, reducing the total number of nodes but leaving unchanged the total elements. This process has the effect of attributing 2 or 3 surface elements to some nodes.

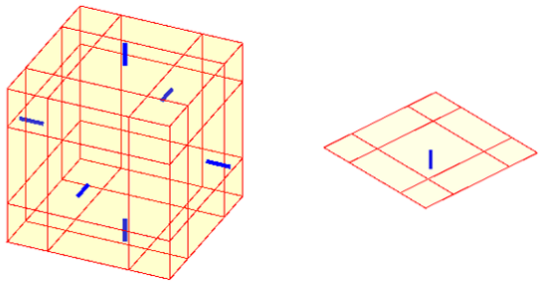


Figure 24: Surface elements of two different geometries. In blue, the default surface normal.

For the next N_L variables of the design vector, a total of $N_{item_L} = N_L/3$ new connections are saved in the connectivity matrix C of the satellite nodes, in positions $C\left(ID_{start_j}, ID_{end_j}\right)$ and $C\left(ID_{end_j}, ID_{start_j}\right)$, symmetrically, since the heat flow in thermal straps is bidirectional and symmetrical. ID_{start_j} and ID_{end_j} represents the IDs of the starting and ending node of the j-th thermal strap.

Analogously, the conductance matrix G_c, which contains the total thermal conductance between each node of the model, is updated in the positions $G_c\left(ID_{start_j}, ID_{end_j}\right)$ and $G_c\left(ID_{end_j}, ID_{start_j}\right)$: the new j-th conductance value is computed as follows:

$$G_{L_j} = \frac{k_L \cdot A_{L_j}}{L_{L_j}} \tag{23}$$

Where $k_L = 385\ W/K$ is the copper thermal conductivity, A_{L_j} is the cross-sectional area of the j-th thermal strap and L_{L_j} is the Euclidean distance between the coordinates of ID_{start_j} and ID_{end_j}. The computational complexity here is $O\left(N_{item_L} \cdot N_N\right)$.

The final N_H variables of x are used to determine the average heat dissipation of the heaters, in the hot or cold case. These average power values are added to the dissipation of the model and are distributed to the nodes of the items subject to heating by active thermal control equipment. The complexity is just $O(N_H)$.

Since the optical properties of the satellites may be changed also on the internal side of the structure faces, Gebhart's view factor must be applied again every time [25]. This step requires solving a $N_N \times N_N$ system of linear equations, and the complexity is, in the worst case is $O\left((N_N)^{\frac{3}{2}}\right)$ [26].

The next step in the flow chart of Figure 21 is to compute the environmental heat sources (heat radiated from the Sun Q_s, albedo heating from the central body Q_a and IR radiation of the central body Q_{ir}). This is a critical step because the heat received by each node depends on the surfaces exposed to the heat source, which in turn depends on the orbit, the Sun and planet view factors and the optical properties α, ϵ of the external surfaces. Since the optical properties may be changed from one design vector to another, the computation of the heat sources must also be performed inside the cost function. Heat sources are calculated for each node, adding up the contributions of each of its surface elements, according to the following equations [27] [24]:

$$Q_{s_j} = I_s \alpha_j A_j F_{sun} \quad \text{with:} \quad F_{sun} = \cos(\theta_j) \cdot F_{ecl} \tag{24}$$

$$Q_{a_j} = a I_s \alpha_j A_j \cos(\beta_j) F_{j,p} \tag{25}$$

$$Q_{ir_j} = I_p \epsilon_j A_j F_{j,p} \tag{26}$$

Where the subscript j indicated the ID of the node, I_s is the solar irradiance (average value of $1367\ W/m^2$), θ_j is the angle between the normal of the elements of node j and the Sun vector (the line connecting the centre of the Sun to the S/C CoM), F_{ecl} is the eclipse factor and can be 0 (when the S/C is in eclipse) or 1 (when the S/C is in sunlight), a is the albedo factor (values between 0 and 1), β_j is the angle between the element normal and the position vector in the ECI frame, $F_{j,p}$ is the view factor between the surface elements of node j and the central body, and I_p is the IR radiation of the central body (averaging $243\ W/m^2$ for Earth). The heat sources are computed for every simulation position, for a single orbit, then, since only steady-state temperatures are involved in the cost function, they are averaged over the entire orbit. The computational complexity of the legacy implementation is $O(T_{orbit}/\Delta t \cdot N_N \cdot N_{F/N})$, where T_{orbit} is the orbital period and Δt is the time step of the simulation.

Then the TMM is re-generated from the ground up in its entirety, with a computational cost of $O((N_N)^2)$. This is because the connectivity matrix C, which has a size of $N_N \times N_N$, must be completely explored to derive the new conductance matrix G_c, the new radiative conductance matrix G_{irr}, and the new capacitance matrix G_{hc}. The physical simulation step of computing steady-state temperatures comes next, and it requires solving a linearized form of the conductive and radiative heat transfer equation, this can be done by iteratively solving a linearized system of N_N equations, until a certain tolerance criterion is met. Depending on the tolerance, the computation has a complexity of $O(K \cdot (N_N)^{3/2})$, where K indicates the average number of iterations required to converge. Once the steady-state temperatures are obtained, the surface-average temperature is computed for every face of the spacecraft in $O(N_N \cdot N_{F/N})$. For the final costs regarding temperature, given that minimum and maximum transient temperatures are not available, a different approach is used instead of computing the temperature gaps G_d, as it is done in the post-processing. The method used to

attribute the costs of the temperature objectives C_T^{HOT} and C_T^{COLD} in the hot and, if selected, in the cold case, is expressed in Equations (27) and (28):

$$C_T^{Hot} = \sum_{j=1}^{N_{item}-1+6} k_{w_j} \cdot \left| T_{SS_j}^{Hot} - \frac{T_{max_j}^O + T_{min_j}^O}{2} \right| \tag{27}$$

$$C_T^{Cold} = \sum_{j=1}^{N_{item}-1+6} k_{w_j} \cdot \left| T_{SS_j}^{Cold} - \frac{T_{max_j}^O + T_{min_j}^O}{2} \right| \tag{28}$$

Where T_{SS_j} is the Steady-State Temperature of the surface j in the hot and cold cases and the term $\frac{T_{max_j}^O + T_{min_j}^O}{2}$ represents the optimal temperature of surface j (assumed to be the middle point between the minimum and maximum operative temperature of that surface). Note that the summation operand index stops at $N_{item} - 1 + 6$, this way every surface is accounted for in the first version of S2T2, namely: $N_{item} - 1$ internal board-like items plus 6 faces for the spacecraft structure. The cost of the heaters is obtained by adding up the heater power applied to each of the $N_{item} - 1$ internal items, analogously to how it is done in the post-processing:

$$C_H^{Hot} = \sum_{j=1}^{N_{item}-1} P_{H_j}^{Hot} \tag{29}$$

$$C_H^{Cold} = \sum_{j=1}^{N_{item}-1} P_{H_j}^{Cold} \tag{30}$$

After a thorough analysis of the cost function, one of the most useful tools to investigate the causes of high computation time in the MATLAB environment is the Profiler, which allows performing a timed execution of a script, outputting a "flame graph" and a time report for every line of code. The flame graph of the cost function of the first version of S2T2 is shown below, in Figure 25: every layer of the graph from the larger ones on the bottom to the ones on top, represents a deeper level of function nesting. More quantitative details are presented in Table 4.

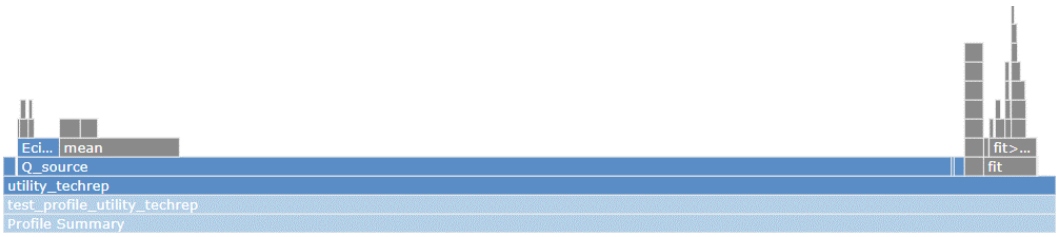


Figure 25: Flame Graph of the cost function of the first version of S2T2

Table 4: Time analysis of the cost function of the first version of S2T2. Total runtime of 17.650 s.

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
Q_source	Function	2	15.659	88.7%	
fit	Function	1	0.892	5.1%	
cfit.subsref	Function	1	0.319	1.8%	
Gebhart	Function	1	0.209	1.2%	
TMM	Function	1	0.175	1.0%	
Steady_comp	Function	2	0.045	0.3%	
mean	Function	2	0.019	0.1%	
Q_dissipation	Function	1	0.015	0.1%	
Q_dissipation2	Function	1	0.015	0.1%	
utility_techrep>@(v)any(v(:)==6+straps(i))	Anonymous function	179	0.003	0.0%	
mpower	Function	10	0.000	0.0%	
Self time (built-ins, overhead, etc.)			0.300	1.7%	
Totals			17.650	100%	

As it is evident from the results of the Profiler run, most of the time is wasted on the function in charge of computing the environmental heat sources, and in particular, inside this function, a lot of time is invested to compute the centre of the surface elements with the MATLAB function *mean* and also to make spacecraft attitude calculations. Apart from the environmental heat sources other notable processes that require attention are the usage of the costly MATLAB *fit* function, the Gebhart method, and the TMM generation. From the result of this first analysis, a summarizing table was created (Table 5), to help guide the code optimization efforts. A priority value based on the observation done up until this point was assigned to each sub-routine of the cost function, with lower values corresponding to higher priorities, meaning that optimization of that part could significantly impact the overall efficiency of the code.

Table 5: Optimizable sub-routines of the cost function.

Cost function sub-routine	Computational Cost	Priority
Update of optical properties	$O(N_N \cdot N_{F/N})$	2
Update of thermal straps data	$O(N_{item_L} \cdot N_N)$	4
Update internal heat dissipations	$O(N_H)$	5
Gebhart's view factors correction	$O((N_N)^{3/2})$	3
Computation of heat sources Q_s, Q_a, Q_{ir}	$O((T_{orbit}/\Delta t) \cdot N_N \cdot N_{F/N})$	1
Generation of TMM	$O((N_N)^2)$	3
Computation of T_{SS}	$O(K \cdot (N_N)^{3/2})$	4
Computation of average temperatures	$O(N_N \cdot N_{F/N})$	3
Compute solution costs	$O(M)$	5

From the time analysis of the Profiler, it can be seen that, although some sections have a higher computational complexity, they are still faster than many others. The difference lies in the implementation: MATLAB is a very high-level language, that under simple one-line operations hides numerous algorithms capable of exploiting the specific nature of the data structures and following the most efficient path to solve a problem. One example is the algorithms for solving a system of linear equations: a developer could, in theory, write a very efficient code to solve a particular case, however, the MATLAB implementation of the backslash operator (also known as the *mldivide* function) already covers all the possibility one could encounter when solving a system of linear equation, automatically choosing the most optimized method (Figure 26). From this it follows that one of the most powerful tools to speed up the code execution in this environment is the elimination of *for* and *while* loops in favour of the vectorization of operation, taking full advantage of the capabilities and optimizations of the programming language in this sense. Citing the MathWorks website, from the "Vectorization" entry of the software Documentation (MATLAB R2023a):

> *Vectorizing your code is worthwhile for several reasons:*
>
> - Appearance*: Vectorized mathematical code appears more like the mathematical expressions found in textbooks, making the code easier to understand.*

- Less Error Prone*: Without loops, vectorized code is often shorter. Fewer lines of code mean fewer opportunities to introduce programming errors.*

- Performance*: Vectorized code often runs much faster than the corresponding code containing loops.*

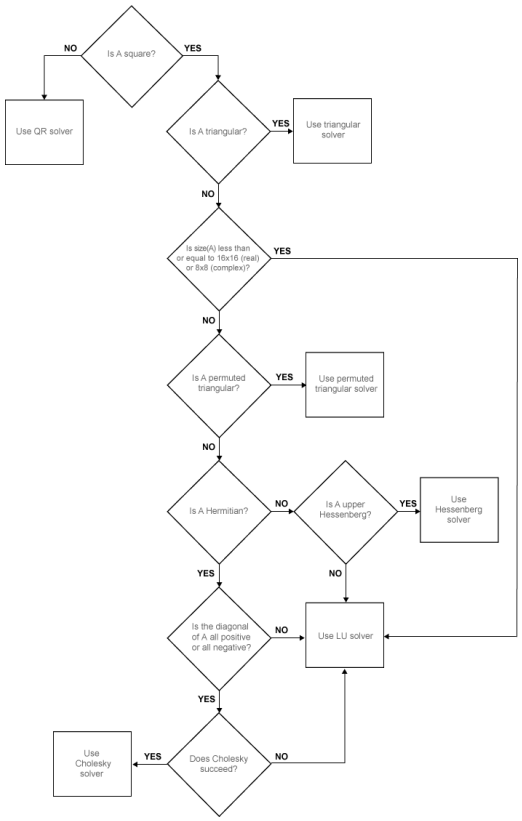


Figure 26: Flow chart of the method used to solve the linear system x=A\b in MATLAB [28].

To illustrate the advantages of vectorized operations, as a practical example, the simple operation of multiplying all the elements of a matrix by a constant is here performed in two ways: the first one consists in using two nested *for* loops to iterate on every element of the matrix, the second one uses the shortened MATLAB syntax which, with just one line of code, completes the operation in a vectorized way. The results are shown in Figure 27, where 10 runs with increasing dimension of the matrix (from 1x1 to 1000x1000) are overlapped on the same graph. The orange lines represent the time taken to complete the element-wise

operation, while the blue lines are relative to the time of execution of the same operation, applied on the same matrix, but using the vectorized syntax.

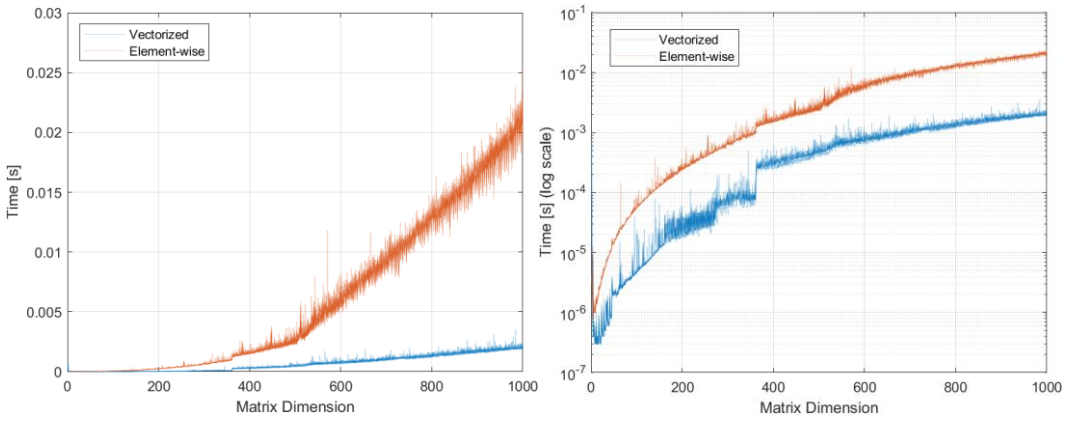


Figure 27: Comparisons of time for scalar multiplication of a random matrix by a constant (vectorized operation in blue, element-wise operation in orange). Linear plot on the left, logarithmic plot on the right.

It is clear from Figure 27 that for every matrix dimension is always more convenient to use vectorized operation when possible: there is about a factor of 10 of improvement in execution time when switching from an element-wise approach to vectorized computations. This was the first and main driver for the modification of the cost function of S2T2.

Another extremely important good practice in writing efficient code is to avoid duplicate calculations: everything that can be computed outside of loops should be completed before entering the loop. Analysing the legacy cost function there are still some data processing operations that can be extracted from deeper nested loops and moved to lower levels, to alleviate the computational load of the section of code executed a high number of times.

Below are listed the 4 main changes done to the cost function to bring down execution time before focusing on all the other new features:

(1) The computation of the linear fit functions was moved outside of the cost function because they did not need to be computed anew for every call.

(2) String comparisons, which are far slower than other type of comparisons, were removed, favouring more efficient integer comparison. Strings were used mainly for node classification, but the same could be done using integer flags and integer variables for indicating node type (e.g., internal and external nodes could be labelled using strings, or, more efficiently with integer numbers). Moreover,

since some operations have to be performed only on the nodes belonging to certain items and since the nodes of the model are saved in a coherent and ordered way no labels are needed to identify the node of a certain item. The total number of nodes of every item is instead exploited for faster indexing.

(3) The TMM function, which is responsible for the creation of the matrices G_c, G_{hc}, G_{irr} should not be run in its totality for every new solution evaluated by the cost function: the GMM is unchanged during the optimization process and only the optical properties of the structure vary. While it is true that computing the Gebhart's correction is required (since ϵ_{int} is an input for the Gebhart's method [29] [30]), not all of the G matrices have to be regenerated. G_c and G_{hc} remain unchanged during the optimization and thus can be computed one time only, before the start of the optimization algorithm. G_{irr} instead, must be calculated again for every solution since it depends on ϵ_{int}, however, some optimizations are still possible. In the legacy version, the G_{irr} matrix was filled using element-wise operations with two nested *for* loops. As it is now apparent from the previous investigation of Figure 27, the optimized way to perform this action is to vectorize the equations, and this can be done using linear algebra applied to the matrices in question, as shown below.

$$G_{irr}(i,j) = \sigma \epsilon_i A_i V_{f_G}(i,j) \tag{31}$$

$$G_{irr} = \sigma \left(\epsilon \cdot A \cdot V_{f_G} \right) \tag{32}$$

Equation (31) illustrates the element-wise computation of G_{irr} from node i to node j, while equation (32) shows how matrix operations can be employed to vectorize the computation. In equation (31) σ is the Stefan-Boltzmann constant, ϵ_i represents the IR emissivity (averaged for every surface element) of node i, A_i is the radiating area of node i and $V_{f_G}(i,j)$ is the view factor (already corrected with the Gebarth's method) from node i to node j. In equation (32) instead ϵ is a $N_N \times N_N$ matrix constructed by N_N column vectors of length N_N, containing ϵ_i in the i-th position, A is also a $N_N \times N_N$ matrix constructed by N_N column vectors of length N_N, containing A_i in the i-th position, and finally V_{f_G} is the complete $N_N \times N_N$ view factor matrix between the nodes of the model. Since equation (32) uses matrices, no iteration loops are required, and the multiplications are managed internally by MATLAB, speeding up the process. Vectorizing the operation also allowed to remove some slow string comparisons. After these changes, in the test

runs the TMM execution time was reduced from 100 ms or more down to only 3 ms, on average, showing the power of vectorization in the MATLAB environment. On top of that the new vectorized code is generally more understandable and compact, which further helps development.

(4) The function for calculating the environmental heat sources, named *Q_source*, is the main bottleneck of the entire cost function. Recalling equations (24), (25) and (26), it is straightforward that, if the orbit is constant, the only part that changes is the multiplication by α or ϵ. This means that every other multiplication can be done beforehand and saved in a matrix with appropriate dimensionality, and the last step, that is the multiplication by α or ϵ, can be done in a vectorized way. The added complication comes from the fact that the heat sources in the cost function need to be evaluated for every temporal step of a single full orbit, for every node of the model and each element belonging to a certain node. Therefore, there are 3 dimensions involved in the computation of all the heat sources, and because of this, the matrices computed beforehand must be 3-dimensional, to correctly save all the information.

$$Q_{s_{t,j,k}} = \tilde{Q}_{s_{t,j,k}} \alpha_{j,k} \quad \text{with:} \quad \tilde{Q}_{s_{t,j,k}} = I_s A_{j,k} F_{sun}{}_{j,k,t} \tag{33}$$

$$Q_{a_{t,j,k}} = \alpha_{j,k} \tilde{Q}_{a_{t,j,k}} \quad \text{with} \quad \tilde{Q}_{a_{t,j,k}} = a I_s A_{j,k} \cos\left(\beta_{j,k}\right) F_{p}{}_{j,k,t} \tag{34}$$

$$Q_{ir_{t,j,k}} = \epsilon_{j,k} \tilde{Q}_{ir_{t,j,k}} \quad \text{with} \quad \tilde{Q}_{ir_{t,j,k}} = I_p A_{j,k} F_{p}{}_{j,k,t} \tag{35}$$

$$t = 1, \dots, T_{orbit}/\Delta t \quad ; \quad j = 1, \dots, N_N \quad ; \quad k = 1, \dots, N_{F/N}$$

To emphasize the tridimensionality: in equations (33), (34) and (35) t iterates over each time step of the orbit, j iterates over every node and k iterates over every surface element that belongs to node j. Note that the areas A depend on both j and k, while the view factors F depend on every one of the tree indexes. $\tilde{Q}_s$, $\tilde{Q}_a$, $\tilde{Q}_{ir}$ are matrices with size $(T_{orbit}/\Delta t) \times N_N \times N_{F/N}$ and can be calculated outside of the cost function and outside of the optimization loop, eliminating a substantial amount of duplicate calculations. The last multiplication by α and ϵ must be done inside the cost function, but, as anticipated, can be performed very efficiently using vectorization. To see how the process is illustrated for Q_s (it is identical for Q_a and Q_{ir}):

$$Q_s = \tilde{Q}_s \cdot \tilde{\alpha} \tag{36}$$

Where $\tilde{\alpha}$ is a matrix with size $(T_{orbit}/\Delta t) \times N_N \times N_{F/N}$ constructed by pairing side by side $T_{orbit}/\Delta t$ matrices with size $N_N \times N_{F/N}$ containing $\alpha_{j,k}$ in positions (j,k). Q_s is also a matrix with size $(T_{orbit}/\Delta t) \times N_N \times N_{F/N}$ and to obtain the resultant heat seen by each node the contribution of all its element must be added up. This is done by summing Q_s over its third dimension, obtaining a final matrix having size $(T_{orbit}/\Delta t) \times N_N$, just like in the first version of S2T2. In the same way as before taking the mean over the first dimension of Q_s a vector of length N_N is obtained, which represents the average heat received by every node from the Sun heat source. The same applies to Q_a and Q_{ir}.

These changes overall are responsible for a very significant performance improvement. The same time analysis as before is run with the help of the MATLAB Profiler. The results are shown in Figure 28 and Table 6.

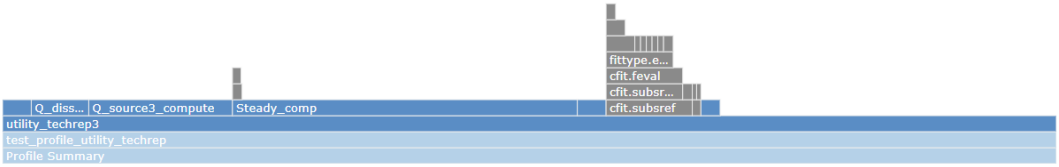


Figure 28: Flame Graph of the cost function after optimization of code.

Table 6: Time analysis of the cost function after optimization of code. Total runtime of 0.110 s.

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
Steady_comp	Function	2	0.037	33.4%	
Q_source3_compute	Function	2	0.015	13.8%	
cfit.subsref	Function	1	0.009	8.3%	
Q_dissipation3	Function	2	0.005	4.4%	
Gebhart	Function	1	0.003	2.8%	
TMM3_only_radiative	Function	1	0.003	2.6%	
utility_techrep3>@(v)any(v(:)==6+straps(j))	Anonymous function	179	0.001	0.9%	
mean	Function	2	0.000	0.3%	
Self time (built-ins, overhead, etc.)			0.037	33.4%	
Totals			0.110	100%	

The most notable change is precisely in the *Q_source* function, which, thanks to a sensible preprocessing step and heavy vectorization of the code, goes from a computational time of more than 15 s to only 15 ms, with a factor of improvement of 3 orders of magnitude.

After further development, S2T2 was updated to work with more complex geometries, namely parallelepipeds, cylinders, and external board-like items [31]. This required a complete rework from the ground up of most of the source code of S2T2. Thanks to a teamwork effort with the author of the companion thesis "Develop of a Tool for Thermal Analysis of Small Satellites" the data structures were improved and changed to be more suitable to the work done in both theses. This rework was taken as an opportunity to fix many bugs in the old source code and to make even more radical changes to the Design tab of S2T2, especially to the cost function. The 7 main changes implemented in this development phase are here summarized, with an emphasis on the ones that were responsible for further improvement of code performances.

(1) The parsing of the optical properties data of the design vector was improved and vectorized, reducing the computational complexity from $O(N_N \cdot N_{F/N})$ to $O(N_{OPT})$.

(2) The parsing of the conductance link (thermal straps) data of the design vector was also improved and vectorized, reducing the computational complexity from $O(N_{item_L} \cdot N_N)$ to $O(N_{item_L})$.

(3) The information on the starting and ending node of the conductance links (thermal straps) in the design vector was encoded using real variables, to preserve continuity of the cost function and to avoid mixing integer design variables with real continuous variables. The information of the starting and ending nodes is now encoded in three variables for each of the two nodes of a link. The three variable represents the coordinates of a point contained in the bounding box that circumscribes the items selected for the connection of a thermal strap. The ID of the node is computed inside the cost function, by finding the node, belonging to the set of items selected, closest to the point identified by the three optimization variables. This process is performed in a computationally efficient way using the MATLAB function *dsearchn*, which requires the Delaunay triangulation of the nodes of the item selected. Fortunately, the triangulation can be derived just one time before the optimization starts, because it depends only on the coordinates of the nodes of the items. The usage of *dsearchn*, instead of having direct access to the node ID, requires more time to complete (about the same time as computing the TMM after being vectorized and optimized) but has the notable advantage of preserving the continuity of the cost function, which has a direct impact on the speed of convergence of the optimization algorithms towards the optimal solutions, as described in the previous section.

(4) Taxicab distance was implemented instead of Euclidean distance to address one of the limitations mentioned in the previous section. The difference is illustrated in the following equations.

$$L_E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{37}$$

$$L_T = |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1| \tag{38}$$

Where L_E is the Euclidean distance, L_T is the taxicab distance, and $x_1, y_1, z_1, x_2, y_2, z_2$ are the x, y, z coordinates in the S/C body reference frame of the two nodes connected by a link. This distance is not only more conservative than the Euclidean one but is also more representative of the real situation encountered in the design of Small Sats: any two nodes of the satellite can rarely be connected with a thermal strap going in a straight line. Most of the time thermal straps, just like electrical and data cable connections, must be routed around other subsystems and may take paths longer than a straight line to reach the destination point.

(5) Some computation done inside every call of Gebhart's method was moved outside of the optimization loop, removing duplicate steps, and thus improving efficiency.

(6) The default time step used in the calculation of the environmental heat sources was increased from 5 s to 60 s, to speed up the computation, without significant loss of accuracy (<5%) because the cost function only requires the average heat over one orbit for the steady-state temperature calculation. This setting, however, can now be modified by the user.

(7) The computation of surface and item average temperatures was vectorized, and the coefficient used for the weighted average (which only depends on geometrical properties such as area and volume) are calculated beforehand, outside the optimization loop, for improved efficiency.

Table 7 sums up the improvement to the cost function computational efficiency. Entries highlighted in green indicate portions of the cost function where the vectorization of the code was utilized to improve efficiency.

Table 7: Improvements of the sub-routines of the cost function of S2T2 (green background indicates that the sub-routine is vectorized).

Cost function sub-routine	Computational Cost (first release of S2T2)	Computational Cost (second release of S2T2)
Update of optical properties	$O(N_N \cdot N_{F/N})$	$O(N_{OPT})$
Update of thermal straps data	$O(N_{item_L} \cdot N_N)$	$O(N_{item_L})$
Update internal heat dissipations	$O(N_H)$	$O(N_H)$
Gebhart's view factors correction	$O((N_N)^{3/2})$	$O((N_N)^{3/2})$
Computation of Q_s, Q_a, Q_{ir}	$O\left(\left(\frac{T_{orbit}}{\Delta t}\right) \cdot N_N \cdot N_{F/N}\right)$	$O\left(\left(\frac{T_{orbit}}{\Delta t}\right) \cdot N_N \cdot N_{F/N}\right)$
Generation of TMM	$O((N_N)^2)$	$O((N_N)^2)$
Computation of T_{SS}	$O(K \cdot (N_N)^{3/2})$	$O(K \cdot (N_N)^{3/2})$
Computation of average Temperatures	$O(N_N \cdot N_{F/N})$	$O(N_N)$
Compute solution costs	$O(M)$	$O(M)$

The flame graph and the time analysis of the final implementation of the cost function are visible below in Table 8 and Figure 29, respectively.

Figure 29: Flame Graph of the cost function in the last release of S2T2.

Table 8: Time analysis of the cost function in the last release version of S2T2 (60 s time step for environmental heat sources computation). Total runtime of 20.7 ms (averaged over 100 executions).

Function Name	Function Type	Calls	Total Time (s)	% Time	Time Plot
Steady_comp	Function	200	1.174	56.7%	
Q_source3_compute	Function	200	0.475	22.9%	
Gebhart2	Function	100	0.103	5.0%	
Q_dissipation3	Function	200	0.039	1.9%	
dsearchn	Function	200	0.028	1.4%	
TMM2_only_radiative	Function	100	0.024	1.2%	
mean	Function	200	0.019	0.9%	
mpower	Function	1	0.000	0.0%	
Self time (built-ins, overhead, etc.)			0.209	10.1%	
Totals			2.070	100%	

From Table 8 and Figure 29 is now evident that the majority of time is used for the computation of steady-state temperatures, which is a process that was already vectorized and optimized from the first version of S2T2. This step is also very well optimized from the aspect of convergence speed of the iterative solving

of the linearized system of heat equations; thus, no other impactful improvements are deemed possible at the moment.

3.3.2 Optimization Process

The whole optimization sequence of the Design tab has been improved. Many new features, designed specifically to address the limitations of the previous version of S2T2, were implemented, and the UI was renovated. The changes are visible in Figure 30. The new utilization flow of this module is explained in this sub-section, while also presenting the new features. The typical sequence of input required to prepare an optimization task takes the user from one panel to another, following the reading order: optical properties, conductance links, heaters, optimization algorithm settings and finally decision-maker settings.

One of the most prominent new features is the possibility of editing the lower and upper bounds of each optimization variable. This is a key point, because it provides the user a higher level of freedom and customization when preparing an optimization task, while also allowing S2T2 to be used for satellites larger than CubeSats, scaling better to Small Sats in general. To illustrate the importance of giving control over lower and upper bounds to the user, an example is used: the subject is a satellite where one of its external faces can be optimized considering various surface finishes/coatings/tapes with α_{ext} and ϵ_{ext} between 0.05 and 0.95, but the face has a constraint: 60% of the area is covered by solar cells ($\alpha_{ext} = 0.903$, $\epsilon_{ext} = 0.85$) and cannot be altered. Since there is a constraint, if optimization of the optical properties of the face in question with S2T2 has to be carried out, it is not sensible to have the two design variables corresponding to α_{ext} and ϵ_{ext} bounded between 0 and 1. In fact, only the remaining 40% of the free surface can be optimized, and since S2T2 considers only 1 material for each surface of the satellite structure, a good approach would be to average the optical properties averaged over their surface. In the example: the lower and upper bounds of α_{ext} and ϵ_{ext} would be:

$$\begin{cases} \alpha_{ext}^{(L)} = 0.6 \cdot 0.903 + 0.4 \cdot 0.05 = 0.5618 \\ \alpha_{ext}^{(U)} = 0.6 \cdot 0.903 + 0.4 \cdot 0.95 = 0.9218 \\ \epsilon_{ext}^{(L)} = 0.6 \cdot 0.85 + 0.4 \cdot 0.05 = 0.53 \\ \epsilon_{ext}^{(U)} = 0.6 \cdot 0.85 + 0.4 \cdot 0.95 = 0.89 \end{cases} \quad (39)$$

These values can be now easily copied inside the appropriate UI field to perform an optimization bounded only to the feasible solutions, saving computational time by avoiding wasting cost function evaluations.

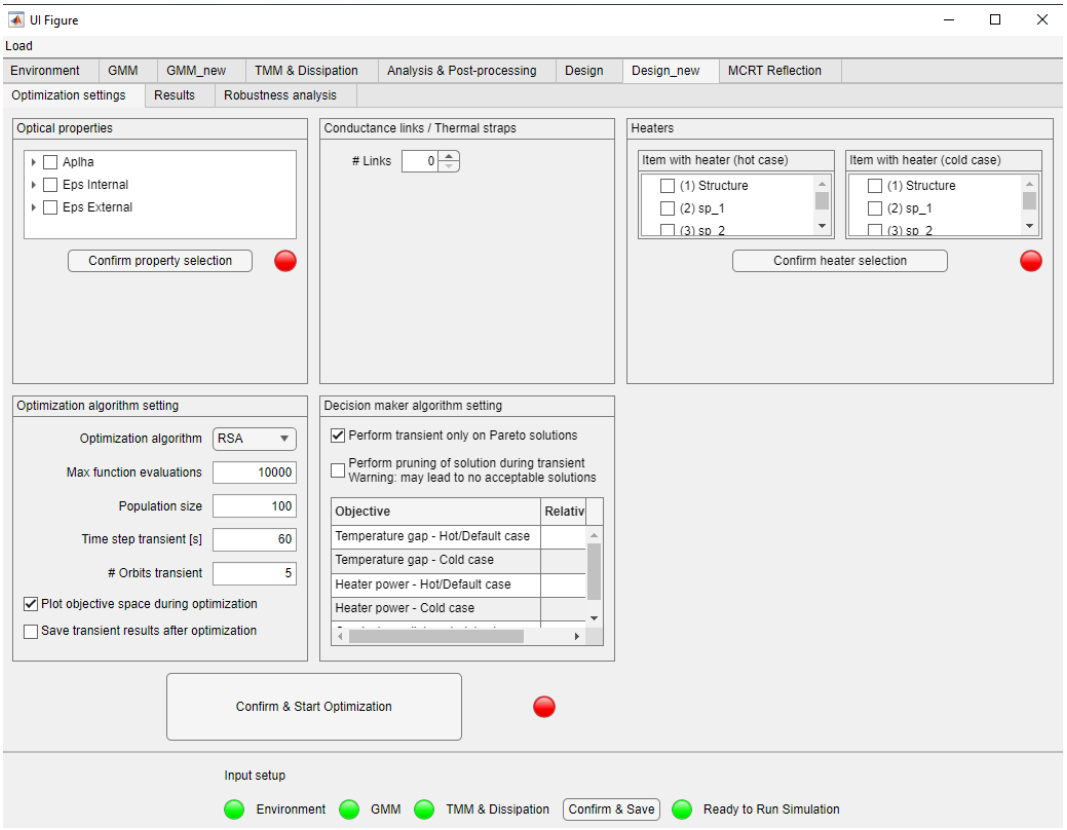


Figure 30: New UI of the Design tab of S2T2

Each panel visible in Figure 30 will now be analysed in more detail. For the optical properties panel a tree data structure was created, to allow the selection of single optical variables instead of always considering all the tree variables α_{ext}, ϵ_{ext} and ϵ_{int} simultaneously, giving more freedom to the user, with the idea of separating the optimization of external and internal properties. This can be useful if the design of one of the two sides is already fixated while the other still has some degree of freedom. After selecting the desired variable from the 18 available

(α_{ext}, ϵ_{ext} and ϵ_{int} for each one of the 6 faces of the S/C structure), clicking on "Confirm property selection" a table opens, allowing the user to enter the lower and upper bounds of these variables (only real numbers between 0 and 1 are accepted). The layout is visible in Figure 31.

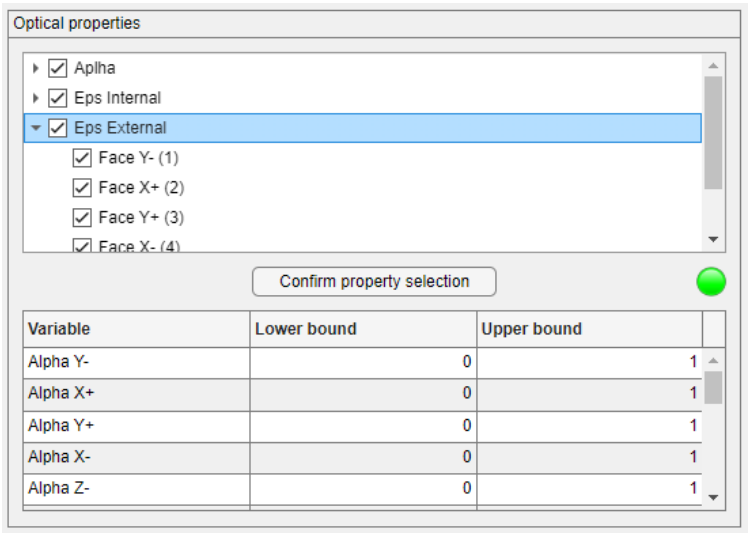


Figure 31: Detail on the Optical properties panel of the new Desing tab of S2T2.

In the conductance links / thermal straps panel initially, only a spinner is present. Increasing the value of the spinner, which represents the maximum number of conductance links to be added in the optimization, causes additional UI elements to appear (Figure 32).

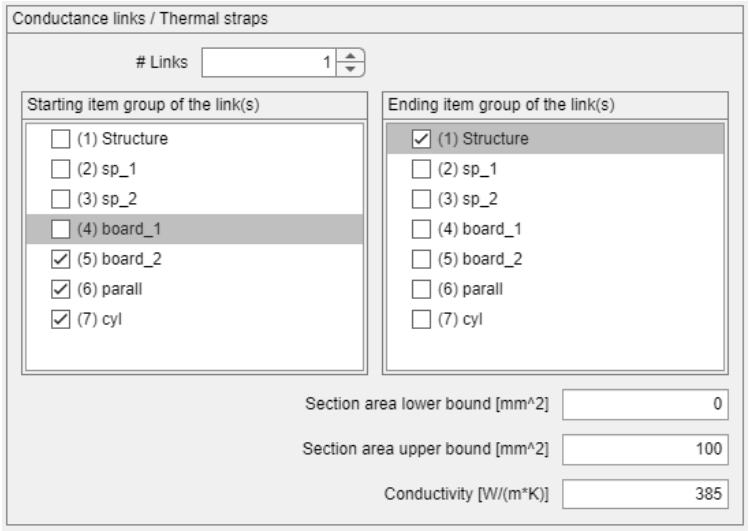


Figure 32: Detail on the Conductance links / Thermal straps panel of the new Desing tab of S2T2.

The approach to thermal straps was changed compared to the previous version of S2T2: to expand the possibilities of optimization the user can now select from two checkbox lists the items of the model to consider as “starting group” (on the left) and to one to consider as “ending group” (on the right). Now, the nodes of the model belonging to the item in the starting group constitute the starting set of nodes, while those belonging to the item in the ending group form the ending set of nodes. Note that these two sets can be disjoint or can have a non-empty intersection, depending on user input. The starting point and ending point of each conductance link are determined during the optimization by three x, y, z cartesian coordinates, as explained in the previous sub-section, which are bounded between the maximum and minimum values of the x, y, z coordinates of the nodes in the starting and ending sets, respectively. The ID of the starting and ending nodes is calculated by finding the node closest to the three x, y, z cartesian coordinates used as optimization variables, with the help of Delaunay triangulation. In Figure 33 two example items, a board and a cube, belonging to the starting (or ending) group are shown; the red transparent bounding box represents the volume in which the starting (or ending) point of the conductance link may lie. The main drawback of this method is the fact that the probability density is not uniform for the nodes of the same set: nodes near the middle of the bounding box have a higher probability of being selected, while nodes closer to the border are not favoured. However, due to the evolutionary nature of the optimization algorithms, even if the optimal placement of thermal straps corresponds to a situation harder to reach, the optimization is still capable of converging to those solutions, escaping from local optima.

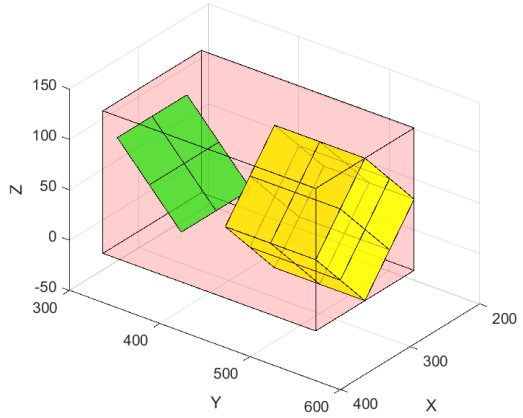


Figure 33: Bounding box around two items, used to decide the starting or ending point of thermal straps.

This new approach simplifies the optimization process and lets the optimizer explore some possibilities that were not possible in the previous version of S2T2:

for example connecting two internal items to each other, useful when the heat coming from one of them must be redistributed to some other cold internal component, but also connecting two thermal straps to the same item, creating a "thermal bridge" where the capacitance of the item in the middle can be exploited to dampen temperature oscillations. Moreover, this approach, as explained before, has the beneficial effect of restoring the continuity of the cost function.

After the selection of the starting and ending item for the conductance links, the user can then specify a lower and upper bound for the cross-sectional area of the link (here the potential for the new version of S2T2 to scale towards larger satellites is evident) and can also edit the thermal conductivity k_L of the material used for the link. k_L remains constant during the optimization because the main degree of freedom is the area A_L of the strap, this way the conductance of the link G_L has two fixated parameters (k_L, user-defined, and L_L, depending on ID_{start} and ID_{end}) and one free parameter, A_L.

To prevent the algorithm from always using the maximum possible amount of material to create conductance links between the nodes of the model, a new objective, C_L, has been introduced, which considers, for each solution, the total volume of material used for the thermal straps:

$$C_L = V_L = \sum_{j=1}^{N_{item_L}} A_{L_j} \cdot L_{L_j} \tag{40}$$

Where N_{item_L} is the maximum number of conductance links in the optimizations and corresponds to the value of the UI spinner labelled "# Links". C_L represents a cost and thus it is a minimization objective of the optimization.

On the right, in the heaters panel, one or two checkbox lists can be found (one for the hot case and one for the cold case if two case studies in S2T2, or just one for the default case otherwise). Contrary to what was possible on the first version of the software, now there is the possibility to apply a heater for every single item, independently, in a differentiated way for the hot and cold case, if desired. This improves the flexibility of the optimization and does not force the cost function to consider wasteful situations in which, for example, heaters were applied to components already hot. After the selection, similarly to the optical properties panel, clicking on the "Confirm heater selection" button one or two new tables appear (visible in Figure 34), and the user can enter the upper and lower bounds in

the form of heater power for each item selected. Both here and in the optical property panel if the user changes its selection the confirmation button must be clicked again for the tables to update.

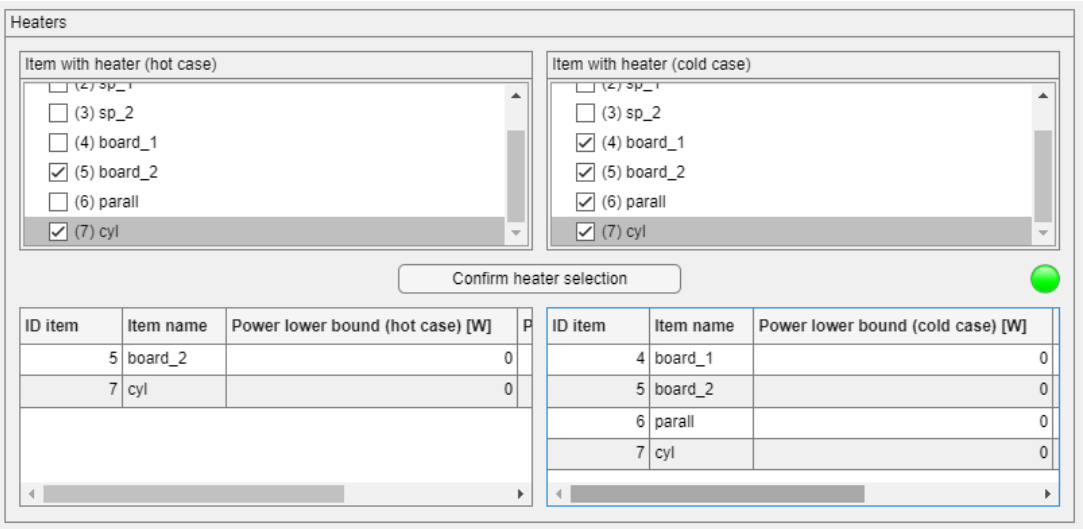


Figure 34: Detail on the Heaters panel of the new Desing tab of S2T2.

In the panel of the optimization algorithm settings, the optimization algorithm can be selected. The legacy implementation which used *gamultiobj* was adapted and it is now named "NGSA-II", from the name of the parent algorithm. All the optimizers mentioned in Chapter 2 were implemented but, after a process of benchmarking (presented in Chapter 4), only 7 of them made the final cut and are available in the new release of S2T2, as is visible in Figure 35.

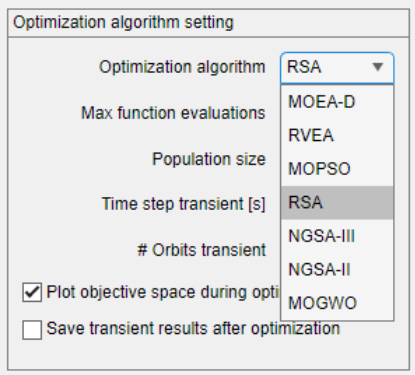


Figure 35: Detail on the Optimization algorithm settings panel of the new Desing tab of S2T2.

The number of function evaluations can now be set by the user, to give more freedom of choice: sometimes faster optimization to test the parameters entered may be desirable, but the possibility of long-running detailed optimizations

remains. The suggested value is 1000 for very fast trial optimizations and 10000 to 50000 or more function evaluations for more detailed searches of the design space. The population, which is the number of solutions that the algorithm keeps track of during the optimization, can be set in the field below. The same logic applies here: for faster trial optimization a number around 20 is suggested while for detailed optimizations 100 to 200 or more individuals are recommended. The population size has also a direct influence on the number of final Pareto front solutions: depending on the algorithm used, the solution returned can be equal to or less than the population size.

Having a variety of different optimization algorithms to choose from is beneficial for the user experience and for the quality of the solutions found. The way it is suggested to deal with these 7 possibilities is to try fast optimizations (with fewer function evaluations and smaller populations) for each algorithm and, looking at the costs of the final solution, determine which one is more suitable for the specific problem the user has to face. When in doubt, it is recommended to use RVEA, MOEA/D or RSA, because of their superior performances, on average (see Chapter 4 for details).

In the same panel, a new time step can be set for the transient analysis that is performed on the final solution in the postprocessing phase. Since the main goal of the transient analysis is to derive the minimum and maximum temperature experienced by the items for every solution, a coarse time step can be used without compromising the process. This, however, depends heavily on the orbit of the satellite and is up to the user to find a good compromise value between accuracy and performance. During the testing of the code, for LEO orbits, a value of 30 to 60 seconds was found to be appropriate to differentiate between good and bad solutions. The field where the number of orbits is specified is also used in the transient analysis and has a minor effect: propagating for more orbits improves the convergence of computed temperatures towards more precise values. The cause of the problem is the initial guess for the temperature distribution, which by default it is assumed equal to steady-state temperatures. However, if the orbit has non-zero eclipse time this distribution may be far from the real distribution at the starting point of the simulation. Propagating for multiple orbits usually this effect is corrected. From the tests conducted, for LEO trajectories after 5 full orbits the absolute error (measured on the temperatures) caused by this effect is usually in the order of 10^{-2} °C. It is useful to be able to update this value after the classical simulations to save time in the post-processing phase, especially if the number of orbits previously set in the Environment tab is very high. The checkbox “Plot

objective space during optimization" turns on a plot of the objective space of the current solutions. If the number of objectives M is higher than 3, only the first 3 objectives are plotted. This setting gives the user an immediate visual interpretation and feedback on the quality of the results, however, it can slow down the optimization. It is suggested to select this option when performing test runs and to disable it for longer searches, to save computation time. The checkbox "Save transient results after optimization" instead saves in a MATLAB .mat format the temperature results and the satellite data of each transient analysis performed (1 for each of the final solutions returned by the optimizer), organizing them in the subfolder "results/trans" of the local software path. The filenames of these results are identified by the ID of the solution and the date and time of creation, to improve the management of the data.

In the final panel, the decision-maker setting can be adjusted (layout in Figure 36). The first checkbox, "Perform transient only on Pareto solutions", when selected, discards all the solutions that are not on the first Pareto front, keeping only the best-performing ones, the non-dominated solutions. It is advised to leave this checkbox on if the population size is sufficiently large, to save time in the post-processing, focusing the computation only on the most promising design points. The second checkbox, "Perform pruning of solution during transient" implements a pruning of the solutions that discards the solutions where the minimum or maximum temperature of one or more items of the model exceed the operative temperatures (both in the high or low sense). The solution is discarded as soon as this condition is met, stopping the simulation and moving to the next one, saving time in the postprocessing phase. This, while on one hand saves time, on the other may reject a large portion of the final design points, or, in some cases, all of them. For this reason, if the user has no particular time constraint it is recommended to leave the option unchecked; instead, in case a time saving is needed, and the option is selected, it is recommended to already implement some changes on the model to improve the temperature ranges if too many solutions are being deleted. In the table below the two checkboxes, the weights $W_1, W_2, \ldots$ used by the decision-maker to assign the utility score U can be set (see equations (22) and (42)). By default, the weights are all equal and are set to 1.

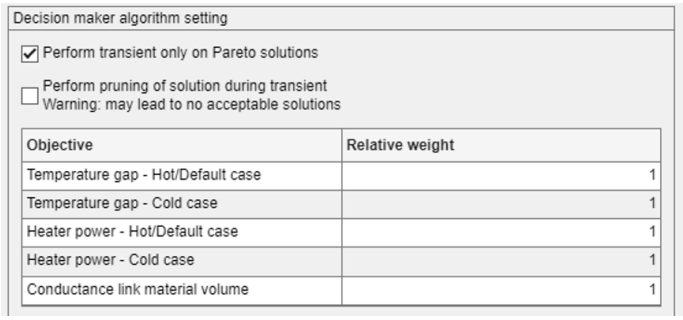


Figure 36: Detail on the decision-maker settings panel of the new Desing tab of S2T2

After entering all the parameters and deciding all the settings, clicking on "Confirm & Start Optimization" starts the optimization. During the process, the button name changes to indicate the status of the computation and to give an estimation of the total time required. The final layout after the optimization is visible in Figure 37. The plot on the left will show the costs of the final solutions that will be post-processed.

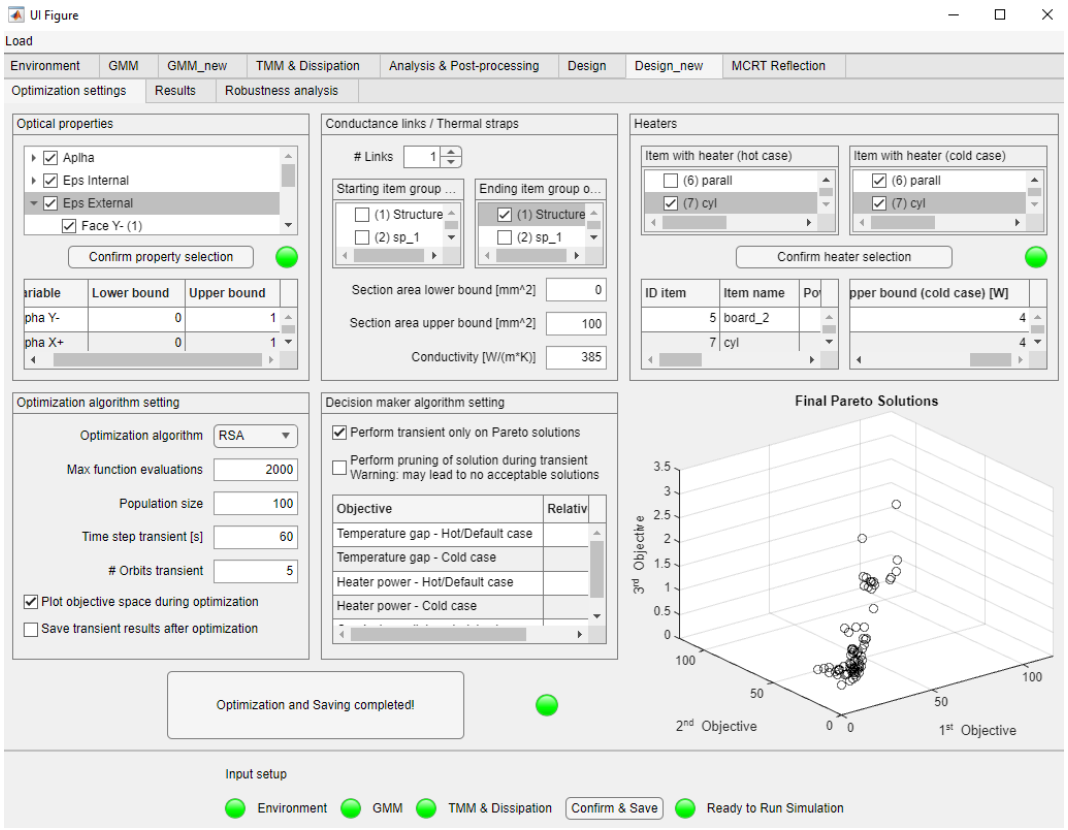


Figure 37: Final layout of the S2T2 Design tab after the optimization process.

Based on the explanation given in this sub-section and the previous one, the new structure of the design vector used during the optimization is presented in Figure 39; while Figure 38 shows the new summary of objectives present in any optimization job, depending on the design variables entered by the user.

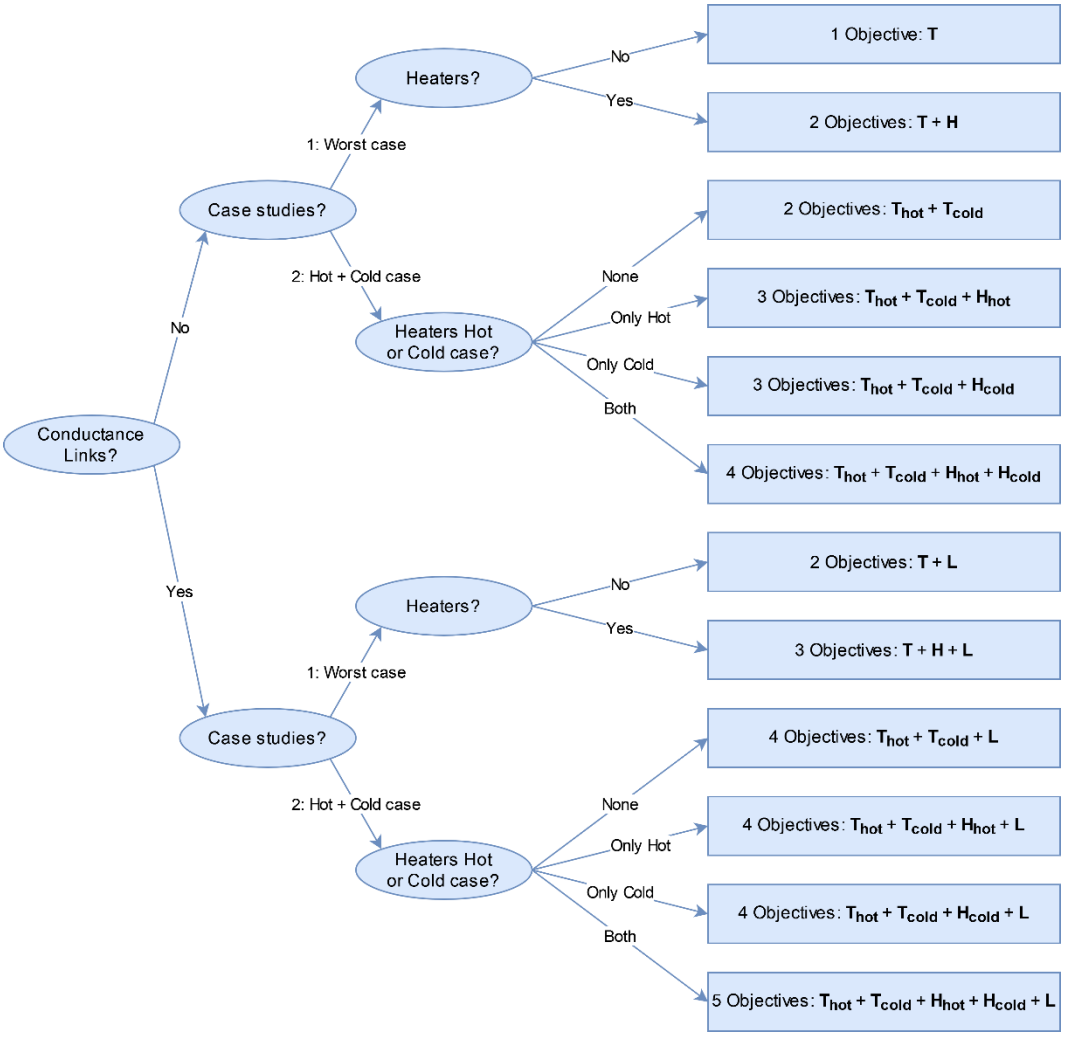


Figure 38: Summary of the number of objectives considered for the optimization in the new version of S2T2.

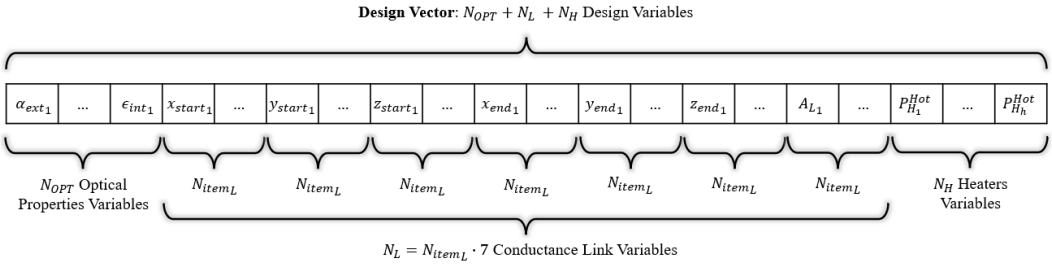


Figure 39: Structure of the design vector x in the new version of S2T2.

3.3.3 Post-Processing

The main improvements implemented in the post-processing phase were centred on the adaptation of the legacy code to the new features described in the previous subsections.

One very impactful change for the transient analysis of every solution was to exploit the new approach to the computation of the environmental heat sources already implemented for the cost function. As explained before, since the orbit is the same for every solution there is no need to compute every time the view factor with the Sun, the central body, and the space, and even the computation of the heat absorbed by each node from the environment can be preserved, up until the last multiplication by the optical properties, the only variable quantity in this process. Thus, analogously as before, the quantities $\tilde{Q}_s$, $\tilde{Q}_a$, $\tilde{Q}_{ir}$ can be computed one time before iterating over the solutions and the final values of Q_s, Q_a, Q_{ir} can be calculated efficiently, using vectorization, in the same way as before. To give an idea of the time saved with this method it is possible to recall the tests performed on the cost function: from the old version to the new one each computation of *Q_source* is faster by a factor of ~1000.

Another small variation was to consider only the final orbit to compute the minimum and maximum temperature of the items during the transient analysis of the optimal solutions. This way errors on the steady-state computation are as small as possible, however, a sufficient number of orbits must be specified beforehand in the optimization settings.

One relevant change regards the introduction of the new objective C_L which considers the volume of material used for thermal straps. This new objective is used in the final computation of the utility value U, similarly to the heater power (equations (20) and (21)): first an additional linear scaling function is defined:

$$f_L: [\min(V_L), \max(V_L)] \rightarrow [1, 0] \tag{41}$$

Where V_L is the total volume of material used for conductance links by each solution found by the optimizer. The function f_H^{Cold}, similarly to the other objectives, is used to normalize the costs C_L of each solution before the final computation of U, which is performed similarly as before:

$$U = \frac{\sum_{i=1}^{7} W_i \cdot F_i}{\sum_{i=1}^{7} W_i} \tag{42}$$

With:

$F_1 = f_{low}^{Hot}\left(G_{d_{low}}^{Hot}\right)$, $F_2 = f_{high}^{Hot}\left(G_{d_{high}}^{Hot}\right)$, $F_3 = f_{low}^{Cold}\left(G_{d_{low}}^{Cold}\right)$, $F_4 = f_{high}^{Cold}\left(G_{d_{high}}^{Cold}\right)$, $F_5 = f_H^{Hot}(P_H^{Hot})$, $F_6 = f_H^{Cold}\left(P_H^{Cold}\right)$, $F_7 = f_L(V_L)$

After the post-processing, the layout of S2T2 switches to the Results tab (the layout is visible in Figure 40.). Here a complete table of the data for every solution is given: each row represents a different solution, while on the columns there are the values of the design variables, the partial utility metrics F_i, the final values of U, and the costs computed by the optimizer. Under the table a new button to conveniently extract the results to a spreadsheet format was introduced, to facilitate management of large quantities of data. Below, a plot similar to the one available in the first version of S2T2 is generated: it shows all the final design points, with their utility value. The solution with maximum utility is always highlighted in a red circle.

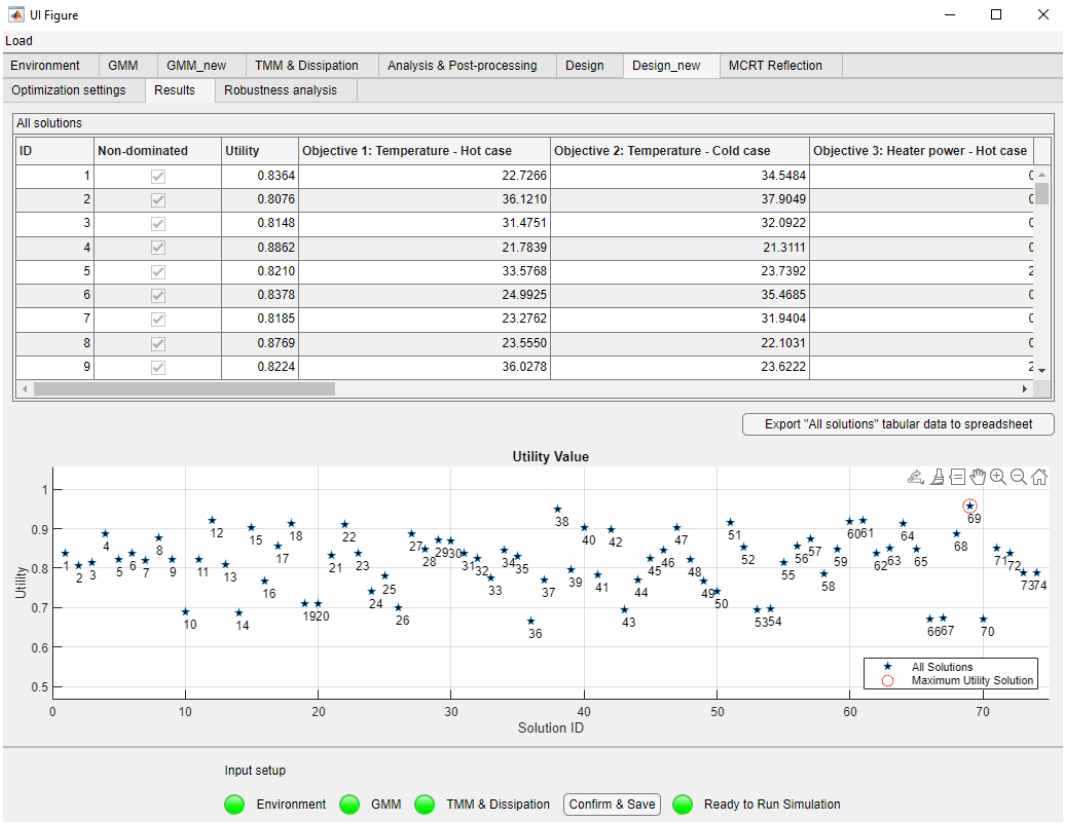


Figure 40: Layout of the Results tab after the optimization process of the new version of S2T2.

3.3.4 Robustness Analysis

The robustness analysis was expanded with some new features and was improved with some considerations similar to those presented in the previous subsection.

Since the analysis consists in performing different transient simulations where the design vector is slightly changed each time, the same consideration for the *Q_source* function applies here: by computing the terms $\tilde{Q}_s$, $\tilde{Q}_a$, $\tilde{Q}_{ir}$ just one time before the iterations and then vectorizing the last step the same amount of time as before is saved, for each transient simulation.

Some new settings to improve the customization of the process were introduced, they are visible in the new layout of Figure 41. The number of perturbations applied to every solution is responsible for the computation time of this process, by default it is set to 10, like in the old release of S2T2 but it can be changed by the user: a higher value means higher accuracy of the final robustness

metric, but also means longer computational times. In the field below the intensity p of the perturbation can be set, in the form of a fractional value. This value is used according to the following equation [24]:

$$x_{p_i} = x_i + (-1)^s \cdot p \cdot r \cdot x_i$$
$$\text{With:} \quad i = 1, \dots, N_{OPT} + N_L + N_H \tag{43}$$

Where s is a random integer that can be 1 or 2, r is a random number between 0 and 1 and x_{p_i} is the perturbed i-th design variable. This behaviour can be changed: if the perturbation is intended to be applied directly to the design vector (default option) the button "Apply perturbation based on solution parameters" must be selected, if instead, the perturbation p refers to a fraction of the full domain of each variable, the button "Apply perturbation based on upper and lower bounds" must be selected, in this case, the perturbed design vector is computed as shown in equation (44).

$$x_{p_i} = x_i + (-1)^s \cdot p \cdot r \cdot \left(x_i^{(U)} - x_i^{(L)}\right)$$
$$\text{With:} \quad i = 1, \dots, N_{OPT} + N_L + N_H \tag{44}$$

The metric used in the first version of S2T2 to estimate the robustness of each design point was the standard deviation σ_U of the U value of the perturbed solutions: a set of perturbed solutions with a high value of σ_U indicates a low robustness starting solution. The new layout of this tab is visible in Figure 41.

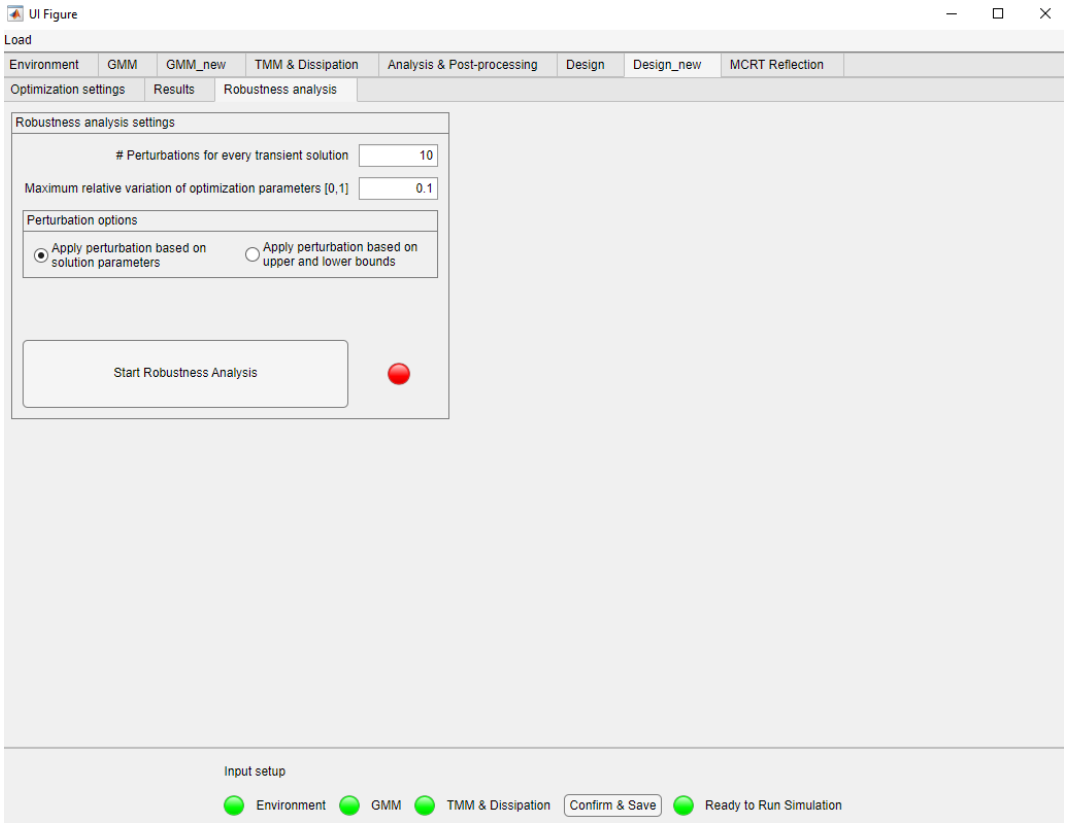


Figure 41: Robustness analysis tab in the new version of S2T2.

A change implemented for this second release was to give a more visible quantitative meaning to the σ_U metric: after the robustness analysis, the value of σ_U of every solution is saved in the table of the Results tab, mentioned in the previous subsection. Additionally, two new plots were introduced in the Robustness Analysis tab, to help the user select the final optimal solution (Figure 42).

The first plot, below the settings, shows the value of σ_U for each solution, to give an immediate visual interpretation of the best-performing design point in the robustness analysis. The solution with the lower value of σ_U is the most robust one and it is labelled with a red circle. The second plot, on the right of the settings, is intended to help the user perform the final trade-off: the horizontal axis indicates the value of σ_U, while the vertical axis refers to the value of U. Ideally, the best solutions to pick are located in the top left of this graph, having a high utility score and high robustness (low σ_U), however the final decision is up to the user, which may favour performances in terms of utility rather than robustness, or vice versa.

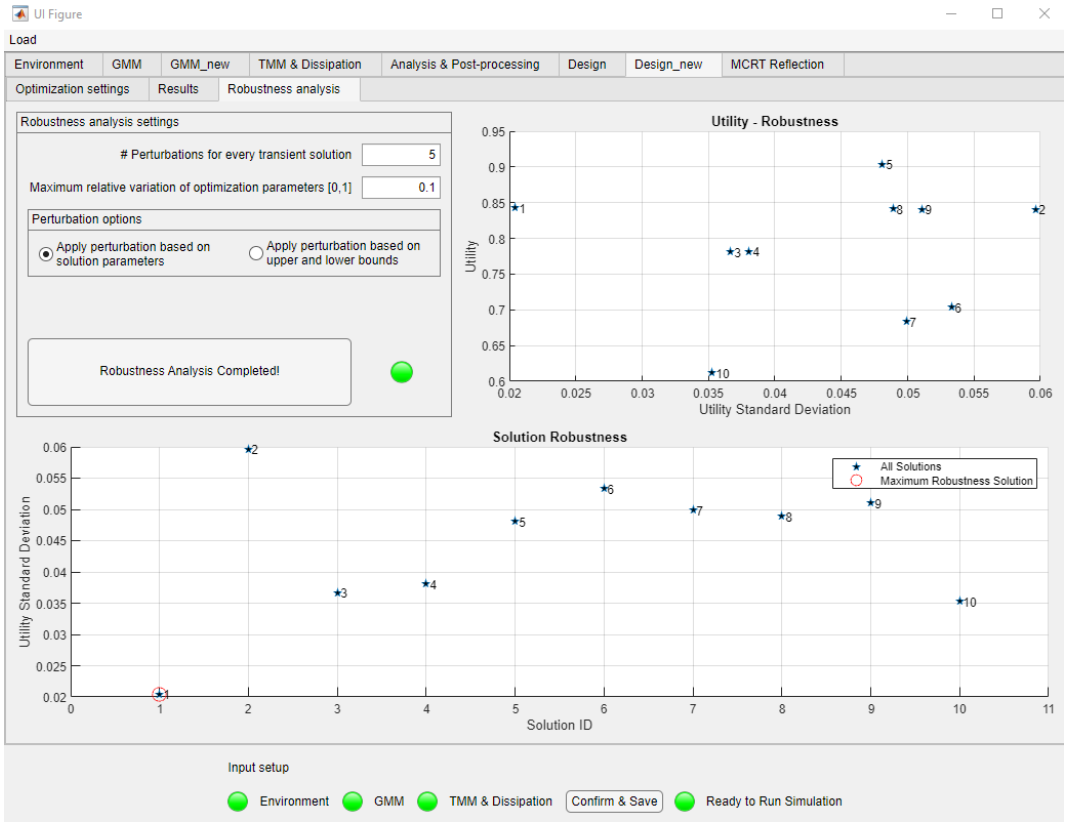


Figure 42: Results of the Robustness analysis tab in the new version of S2T2.

After the robustness analysis, the plot of the utility value U of each solution, found in the Results tab (see Figure 43) is updated by adding error bars to each solution that indicates the standard deviation on the value of U computed in the analysis. This is also useful for the final trade-off assessment by the user.

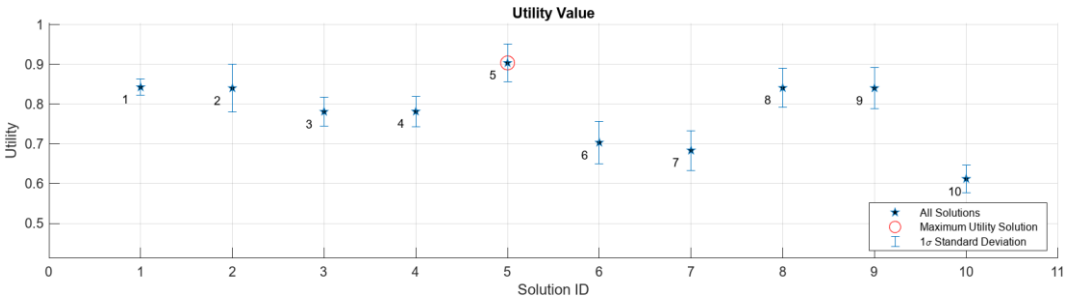


Figure 43: Utility value plot with error bars indicating the standard deviation of the utility value.

3.3.5 Final Overview

After the implementation of all the changes presented in these sections, the new flow chart of the optimization process is shown in Figure 44.

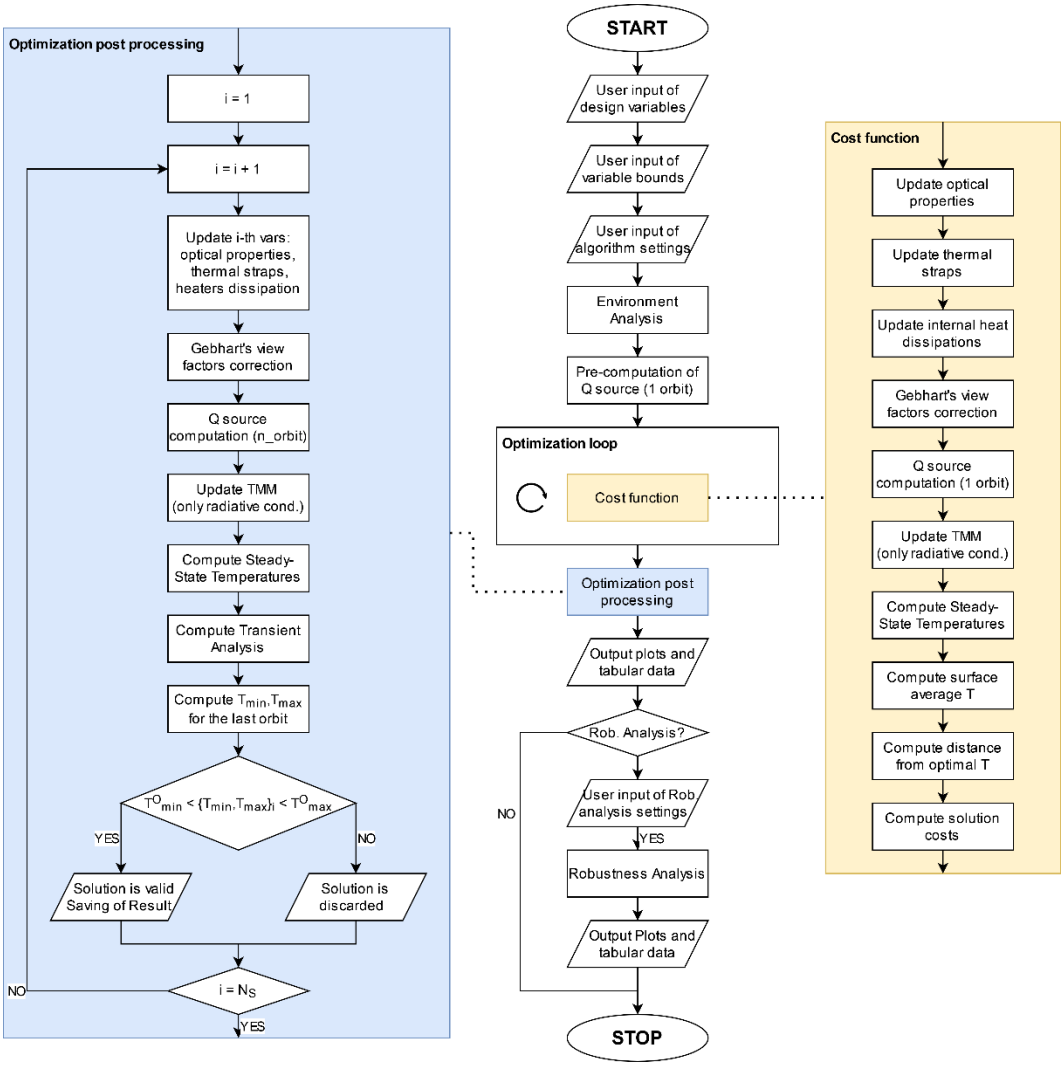


Figure 44: Flow chart of the optimization process in the new release of S2T2.

Another type of graph, that illustrate the sequence of user inputs required to correctly utilize the software was created: the utilization flow paths graph. In Figure 45 the part of the utilization graph regarding the Design tab is shown. The blue rectangle represents the possibility to load data from the .mat file saved from previous optimizations, the yellow ones represent a step where the user can review the data generated by the simulation, and the dotted lines represent use cases different from the nominal flow: in this case, they represent the possibility

to change some setting of the previous analysis and running again the corresponding simulation to update the data. These off-nominal use cases were not correctly managed by the previous version of S2T2, thus, some work was invested to make the application more robust against backtracking user action or improper usage of the functionalities. On top of that many other small quality-of-life changes were introduced, with the idea to enhance the user experience and to speed up the time required by users to input large volumes of data.

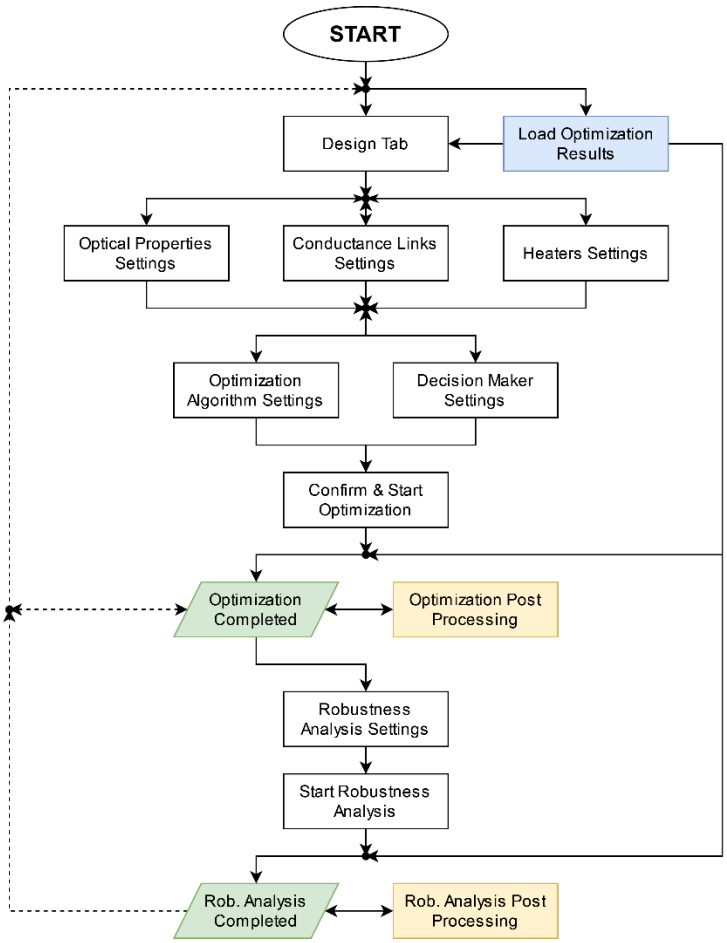


Figure 45: Allowed utilization flow paths of the Design tab. Dotted lines represent off-nominal use cases that are correctly managed by the application in its new release.

Chapter 4

Applications and validations

To understand which new optimization algorithms could be implemented in S2T2 it is of paramount importance to study the performance of the available state-of-the-art optimizers, both in a benchmark scenario and also in a real-life case study. The first section will focus on the benchmarking of the algorithms already mentioned in Chapter 1, which were chosen considering both state-of-the-art performances from scientific literature and availability in the MATLAB environment, to facilitate the implementation and the adaptation for the new release of S2T2. In the second section, the optimization algorithms were tested within S2T2, with the new cost function and the new features, using the SROC mission as a case study. Finally, the third section instead focuses on how the new version of the software was used in support of the design of the TCS of the Spei Satelles mission.

4.1 Benchmark Problems

The benchmark problems can be considered a playing field for comparing different algorithms and techniques, both in terms of quality and speed of computation. In the context of MOO, exist many benchmark suites that include a multitude of test problems, each with their unique characteristics, that can be used to assess and compare the performance of optimizers. The true Pareto front of these problems is usually known explicitly, can be generated simply and can act as the reference set for the evaluation of different performance metrics. The test

problems chosen in this work were the popular ZDT and DTLZ test suites, widely used in the scientific literature.

4.1.1 ZDT Problems

The ZDT problems are a collection of 6 2-objective cost functions, which aims at challenging the algorithms with different level of difficulties, focusing on some of the weaker points of many optimizer. Below the ZDT problems used for the benchmarking are listed, with their definition, the formulation of the true Pareto front and some of their peculiarities and specific challenges [32] [33]. The general formulation for the ZDT problems is:

$$\begin{cases} \text{minimize} \quad f_1(x) \\ \text{minimize} \quad f_2(x) = g(x) \cdot h(f_1(x), g(x)) \end{cases} \tag{45}$$

Where f is the cost function. The optimum variables are indicated by x^*.

4.1.2 ZDT1

ZDT1 is a 30-variable problem ($n = 30$) with a convex Pareto-optimal set.

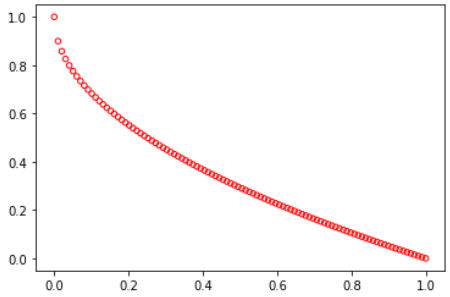


Figure 46: ZDT1 true Pareto front

Definition:

$$f_1(x) = x_1$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{46}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

Bounds:

$$0 \leq x_i \leq 1 \text{ for } i = 1, \dots, n \tag{47}$$

Optimum:

$$0 \leq x_1^* \leq 1 \quad \text{and} \quad x_i^* = 0 \text{ for } i = 1, \dots, n \tag{48}$$

4.1.3 ZDT2

ZDT2 is also a 30-variable problem ($n = 30$) with the added difficulty of having a non-convex Pareto-optimal set.

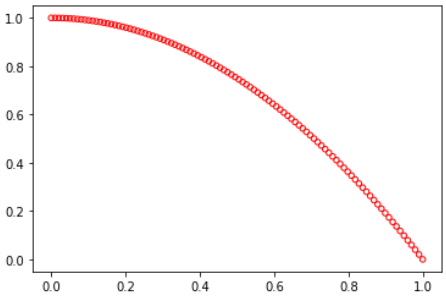


Figure 47: ZDT2 true Pareto front

Definition:

$$f_1(x) = x_1$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{49}$$

$$h(f_1, g) = 1 - (f_1/g)^2$$

Bounds:

$$0 \leq x_i \leq 1 \text{ for } i = 1, \dots, n \tag{50}$$

Optimum:

$$0 \leq x_1^* \leq 1 \quad \text{and} \quad x_i^* = 0 \text{ for } i = 1, \dots, n \tag{51}$$

4.1.4 ZDT3

ZDT3, similarly, is a 30-variable problem ($n = 30$) with the challenge of having several disconnected Pareto-optimal fronts.

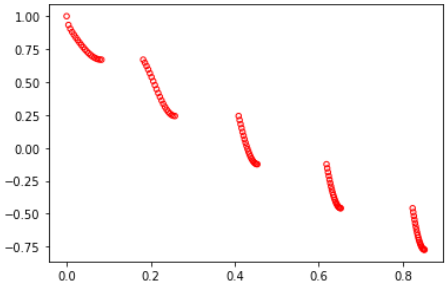


Figure 48: ZDT3 true Pareto front

Definition:

$$f_1(x) = x_1$$

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i \tag{52}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g) \cdot \sin(10\pi f_1)$$

Bounds:

$$0 \leq x_i \leq 1 \text{ for } i = 1, \dots, n \tag{53}$$

Optimum:

$$0 \leq x_1^* \leq 0.0830$$

$$0.1822 \leq x_1^* \leq 0.2577$$

$$0.4093 \leq x_1^* \leq 0.4538 \tag{54}$$

$$0.6183 \leq x_1^* \leq 0.6525$$

$$0.8233 \leq x_1^* \leq 0.8518$$

$$x_i^* = 0 \text{ for } i = 1, \dots, n$$

4.1.5 ZDT4

This is a 10-variable (n=10) problem having a convex Pareto-optimal set. There exist many local Pareto-optimal solutions in this problem. Therefore, algorithms can easily get stuck in a local optimum.

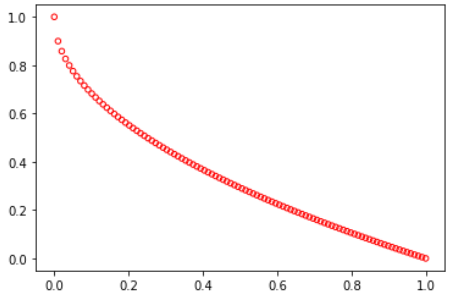


Figure 49: ZDT4 true Pareto front

Definition:

$$f_1(x) = x_1$$

$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 10\cos(4\pi x_i)) \tag{55}$$

$$h(f_1, g) = 1 - \sqrt{f_1/g}$$

Bounds:

$$0 \le x_1 \le 1 \quad \text{and} \quad -10 \le x_i \le 10 \text{ for } i = 2, \ldots, n \tag{56}$$

Optimum:

$$0 \le x_1^* \le 1 \quad \text{and} \quad x_i^* = 0 \text{ for } i = 1, \ldots, n \tag{57}$$

4.1.6 ZDT5

ZDT5 was not used in the benchmarking process since the variables need to be decoded by bit-strings, and all the use cases of the new version of S2T2 are real variable optimizations.

4.1.7 ZDT6

ZDT6 is a 10-variable ($n = 10$) problem having a nonconvex Pareto-optimal set. The density of solutions across the Pareto-optimal region is non-uniform.

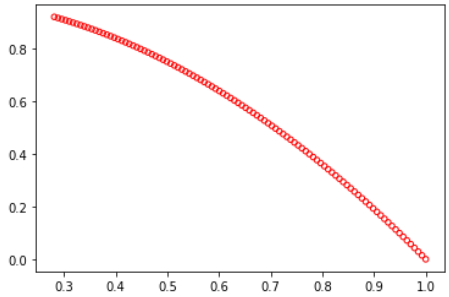


Figure 50: ZDT6 true Pareto front

Definition:

$$f_1(x) = 1 - \mathrm{e}^{-4x_1} \cdot \sin^6(6\pi x_1)$$

$$g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/9\right]^{0.25} \tag{58}$$

$$h(f_1, g) = 1 - (f_1/g)^2$$

Bounds:

$$0 \leq x_i \leq 1 \text{ for } i = 1, \ldots, n \tag{59}$$

Optimum:

$$0 \leq x_1^* \leq 1 \quad \text{and} \quad x_i^* = 0 \text{ for } i = 1, \ldots, n \tag{60}$$

4.1.8 DTLZ Problems

Similarly to the ZDT test problems the DTLZ suite, composed of 7 different problems, is designed to test different capabilities of MOO algorithms, with a special focus on many-objective optimization algorithms. One fundamental aspect of these problems is the possibility to change the number of objective M to an arbitrarily high value, this is achieved by a parametric definition of the cost functions [34] [33].

4.1.9 Benchmark Optimizations

For the benchmarking process, the algorithms to be tested were implemented in MATLAB; the selection of algorithms tested is the same as the list presented in Chapter 2:

- NGSA-II
- PS
- MOPSO
- GODLIKE
- RSA
- NGSA-III
- MOEA/D
- RVEA
- MOGWO

For the test problems, 5 out of the 6 ZDT problems and the complete DTLZ 7-problems suite were executed. For the DTLZ the number of objectives was set to $M = 4$, to differentiate them from the ZDT ones and to try a value of M similar to those encountered in the new version of S2T2. A summary of the benchmark problems implemented is displayed below, in Table 9.

Table 9: Benchmark problems used for comparing MOO algorithms.

Problem Name	Number of Objectives
ZDT1	2
ZDT2	2
ZDT3	2
ZDT4	2
ZDT6	2
DTLZ1	4
DTLZ2	4
DTLZ3	4
DTLZ4	4
DTLZ5	4
DTLZ6	4
DTLZ7	4

The parameters of the algorithms and the stopping condition were set to values similar to those expected from an average optimization run on the new version of S2T2. The population was set to 200 individuals for each optimizer and

the maximum number of cost function evaluations was set to 20000. A total of 10 runs were performed for each problem and for each algorithm, to reach a statistically significant collection of samples.

After the optimization, the performance metrics described in Chapter 2 were used to compare the algorithms. They are recalled and summarized in Table 10.

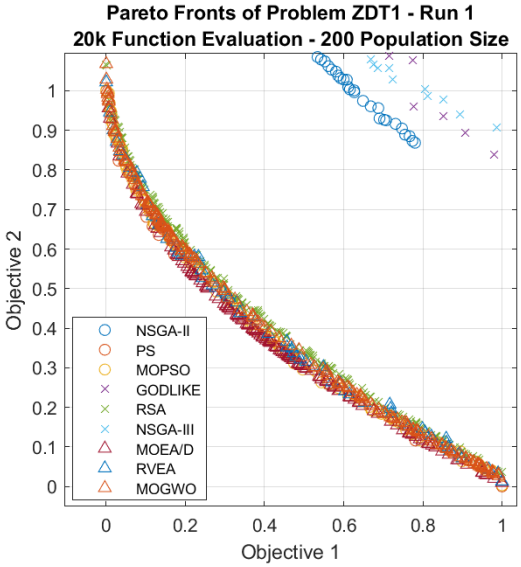
Table 10: Summary of MOO performance metrics used for benchmarking.

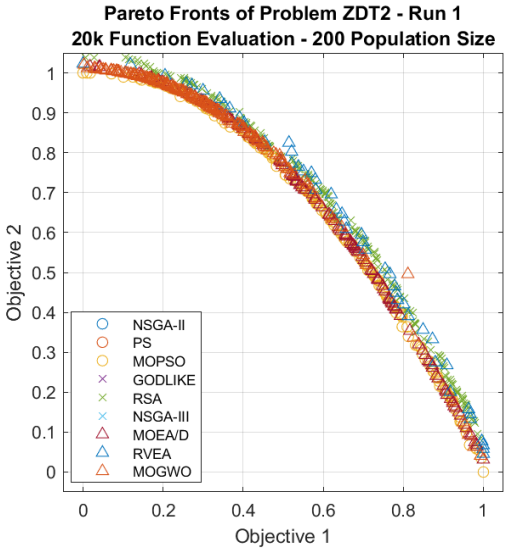
Metric	Measures	Maximized/ Minimized	Unary/ N-ary	Requires true PF	Weight (W)
NR	Cardinality	MAX	N-ary	NO	1
GD	Convergence	MIN	Unary	YES	1
IGD	Convergence/ Distribution	MIN	Unary	YES	5
MPFE	Convergence	MIN	Unary	YES	1
SP	Distribution	MIN	Unary	NO	1
MS	Distribution	MAX	Unary	YES	1
HR	Convergence/ Distribution	MIN	Unary	YES	5
TIME	-	MIN	Unary	NO	1

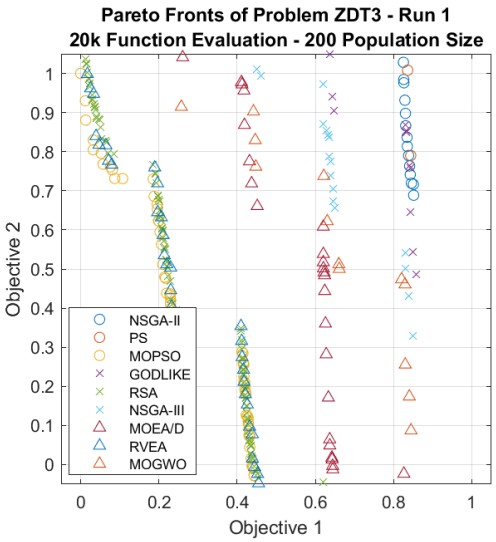
For each metric a weight was assigned, visible in the last column of Table 10, they were derived by analysis of the most used performance metric in the recent scientific literature. Since most of the works of the last years in the field of MOO usually include IGD and HR when making comparisons, those were given a high value compared to the other ones. IGD and HR are indeed particularly meaningful because they can evaluate both the convergence aspect and the spread of the solutions in the objective space in one single value.

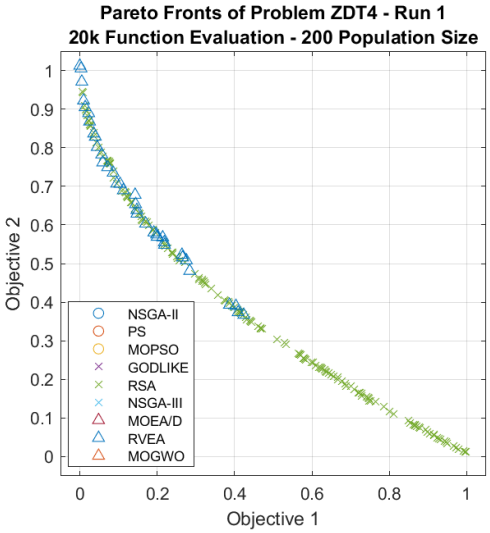
4.1.10 Benchmark Results

In this section, an analysis of the performances and strengths of the 9 algorithms tested is presented, with some graphical representation to help visualise the results. Below, in Figure 51, the final Pareto fronts returned by the 9 algorithms are displayed, for the first run of each ZDT problem. The plots are zoomed near the area of the true Pareto front, which always lies between 0 and 1 for each of the two objectives. More details and graphs relative to the benchmarking are given in Appendix A.


Pareto Fronts of Problem ZDT1 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


Pareto Fronts of Problem ZDT2 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


Pareto Fronts of Problem ZDT3 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


Pareto Fronts of Problem ZDT4 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO

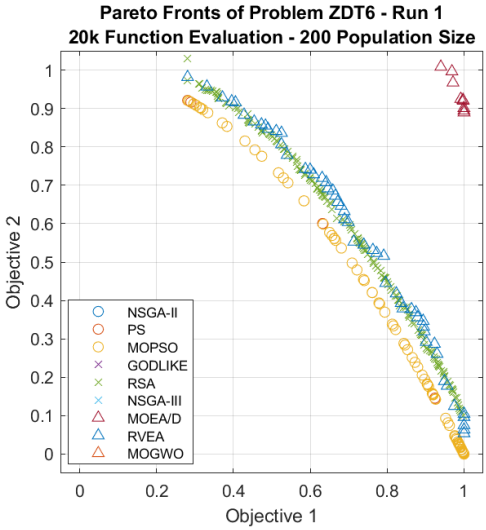


Figure 51: Final Pareto fronts of the 9 algorithms tested for the 2-objectives ZDT problems.

The first problem, ZDT1, does not pose significant challenges, however, even with a high number of total function evaluations (20000) the different convergence speeds can be distinguished: every algorithm closely approached the Pareto front, except for NGSA-II, NGSA-III and GODLIKE which on average requires more function evaluations to converge. This behaviour remained the same also in ZDT2, with a non-convex optimal Pareto front. The challenge of having a disconnected Pareto front, posed by ZDT3, affected the performance of many algorithms; only RVEA, RSA and MOPSO managed to touch the true Pareto fronts, with MOEA/D and MOGWO not far behind. ZDT4 was the hardest problem to solve due to the presence of many local optimum points far from the true Pareto front; while most algorithms got stuck RVEA and RSA reached the optimal front, probably thanks to their more advanced mutation strategies. The clustering technique of RSA, with reference points and adaptive normalization, has proven to be particularly effective also at achieving a very well-balanced distribution of points. Finally, ZDT6 was solved only by the MOPSO algorithm, although with a not very uniform distribution, with RVEA and RSA close behind, overcoming more easily the added difficulty of having a non-uniform optimal Pareto front thanks to the usage of reference vectors.

To give a quantitative measure of the comparisons, the 8 performance metrics were evaluated for each algorithm, for each benchmark problem and each one of the 10 runs executed. The values of the metrics were then normalized in scores between 0 and 1, using linear scaling functions. The average of the normalized scores is shown in the two following tables, conditional formatting was used to

highlight the results from worst (red) to best (green). Table 11 shows the score obtained by each algorithm in each metric, averaging the results from all the 12 problems used. Table 12 instead displays the overall score of each algorithm in each problem, obtained by taking the weighted average across all 8 metrics mentioned before.

Table 11: Average score of the algorithms listed by metric.

	Algorithm	1	2	3	4	5	6	7	8	9
Metric		NSGA-II	PS	MOPSO	GODLIKE	RSA	NSGA-III	MOEA/D	RVEA	MOGWO
W = 1	NR	0.0095	0.282	0.4443	0.00188	0.259	0.1791	0.51041	0.198	0.11043
W = 1	GD	0.2673	0.407	0.6264	0.24242	0.775	0.5772	0.85663	0.743	0.55731
W = 5	IGD	0.2328	0.475	0.6406	0.18523	0.82	0.562	0.77325	0.844	0.48461
W = 1	MPFE	0.1775	0.433	0.6386	0.11422	0.711	0.5678	0.83774	0.899	0.39607
W = 1	SP	0.2086	0.42	0.6812	0.21822	0.732	0.6381	0.87921	0.835	0.58196
W = 1	MS	0.5287	0.624	0.6611	0.5718	0.528	0.2378	0.32662	0.567	0.59997
W = 5	HR	0.2326	0.521	0.7188	0.19601	0.792	0.4035	0.64203	0.833	0.39319
W = 1	TIME	0.9222	0.906	0.9041	0.91783	0.909	0.0867	0.33811	0.922	0.31185

Simply looking at the number of green cells in Table 11 it is clear that the best-performing algorithms were MOPSO, RSA, MOEA/D and RVEA with RVEA scoring highest in the 2 most significant metrics, IGD and HR. RSA is a close second, with good convergence distribution achievements.

Table 12: Average scores listed by problem and final rank of the algorithms.

	Algorithm	1	2	3	4	5	6	7	8	9
Problem		NSGA-II	PS	MOPSO	GODLIKE	RSA	NSGA-III	MOEA/D	RVEA	MOGWO
2 OBJ	ZDT1	0.0949	0.8449	0.9632	0.1082	0.9130	0.2328	0.9328	0.9182	0.8763
2 OBJ	ZDT2	0.1692	0.6547	0.9475	0.1692	0.8641	0.1994	0.7407	0.8387	0.8138
2 OBJ	ZDT3	0.2224	0.1254	0.9998	0.1832	0.8378	0.2792	0.6427	0.9073	0.6192
2 OBJ	ZDT4	0.4039	0.2315	0.3316	0.2798	0.9376	0.4471	0.3494	0.8143	0.2624
2 OBJ	ZDT6	0.2228	0.6391	0.9199	0.2568	0.8485	0.2235	0.7421	0.8326	0.4106
4 OBJ	DTLZ1	0.2175	0.7449	0.6944	0.2779	0.8669	0.7602	0.4386	0.9625	0.6260
4 OBJ	DTLZ2	0.3796	0.6652	0.7016	0.4679	0.9365	0.9239	0.9414	0.9669	0.2296
4 OBJ	DTLZ3	0.4477	0.6708	0.7029	0.1182	0.8041	0.7367	0.7401	0.9527	0.4656
4 OBJ	DTLZ4	0.1701	0.2206	0.6558	0.3679	0.9772	0.7775	0.8028	0.9219	0.4156
4 OBJ	DTLZ5	0.8023	0.4708	0.1936	0.5698	0.6927	0.7181	0.8946	0.3568	0.4916
4 OBJ	DTLZ6	0.3481	0.6035	0.7250	0.2820	0.5184	0.2559	0.9094	0.7769	0.1544
4 OBJ	DTLZ7	0.1296	0.6739	0.9016	0.1468	0.5315	0.2256	0.6609	0.9474	0.2790
AVG SCORE		0.3007	0.5455	0.7281	0.2690	0.8107	0.4817	0.7330	0.8497	0.4703
FINAL RANK		8	5	4	9	2	6	3	1	7

Table 12 shows that every problem posed a different level of challenge for the algorithms. Looking at the overall best-performing optimizers, MOPSO performed well in the 2-objectives test problems but had some difficulties with the problems having non-uniform distributions and on the DTLZ suite in general. RSA had very good overall performances but struggled on the last two DTLZ problems because of the many discontinuities of the optimal Pareto front and the

non-uniform distributions. MOEA/D instead struggled in two problems, where jumping out of local optima was required. RVEA performed very well and showed some problems only on DTLZ5, where it always remained behind the other algorithms and did not find any non-dominated solution compared to the other optimizers. All the other algorithms, while in some problems showed acceptable performance, in general, are not suited to face the difficulties posed by many different problems, thus they have limited versatility and are less interesting in the context of optimization within S2T2.

After averaging all the scores a final ranking, visible at the bottom of Table 12, was produced: the best three algorithms implemented in the MATLAB environment were, in order, RVEA, RSA and MOEA/D. RVEA and RSA, in particular, are very close and are the only two algorithms that reached a final score above 0.8. A strong argument in favour of RSA can also be made: since it uses different tuneable parameters it is more flexible than other algorithms and, although in this thesis it was used with constant parameters, in theory, hyperparameter optimization could be used to adapt the behaviour of the code on the specific difficulties posed by different problems, further improving performances.

4.2 SROC

In this section, the second application will be discussed. A short overview of the SROC mission is first given, and then the thermal model of SROC is briefly presented. After that, the settings of the optimization are discussed and finally, an analysis of the results is given, with some comparisons between the old and new versions of S2T2. Since SROC was already one of the case studies of the work of the first developer of S2T2 [24], here only the main aspects are recalled, and more space is given to the analysis of the results and the performances of the optimizers.

4.2.1 SROC Mission

The Space Rider Observer Cube (SROC) mission aims at demonstrating the critical capabilities and technologies required for successfully executing a rendezvous and docking mission in a safety-sensitive context. The SROC space system is constituted of a nanosatellite and a deployment & retrieval system. The system will perform a mission featuring Proximity Operations in the vicinity of the Space Rider (SR) vehicle before docking and re-entering Earth with the mothership.

The SROC project aims at developing and testing in space novel key technologies in the area of proximity operations, such as Propulsion systems (cold gas), Guidance Navigation and Control (hardware and software), Electro-optical systems (visual camera), Mechanisms (docking, deployment and retrieval), and at improving Autonomous Operations.

The SROC mission has been thought of as an add-on to complement the Space Rider project. From the SROC side, the mission will demonstrate enabling technologies in the proximity operations domain, which can also be transferred to other targets. From the Space Rider side, there is the opportunity to demonstrate the capability to deploy & retrieve payloads, thus expanding the range of possible applications of the vehicle.

4.2.2 SROC Space Segment and its Thermal Model

SROC is a 12U CubeSat, organized into three bays hosting different subsystems: a top section for the main bus avionics, including batteries and transponders, a centre section for the thruster module and a bottom section for

hyperspectral payload, navigation and observation cameras, LIDAR and docking system [24]. The internal configuration can be seen in Figure 52.

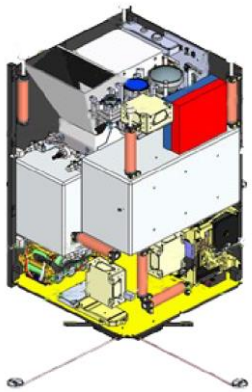


Figure 52: Internal view of SROC spacecraft

The operative orbit of SROC is a dawn/dusk SSO orbit, the specific environmental data are visible in Figure 53, in the S2T2 UI.

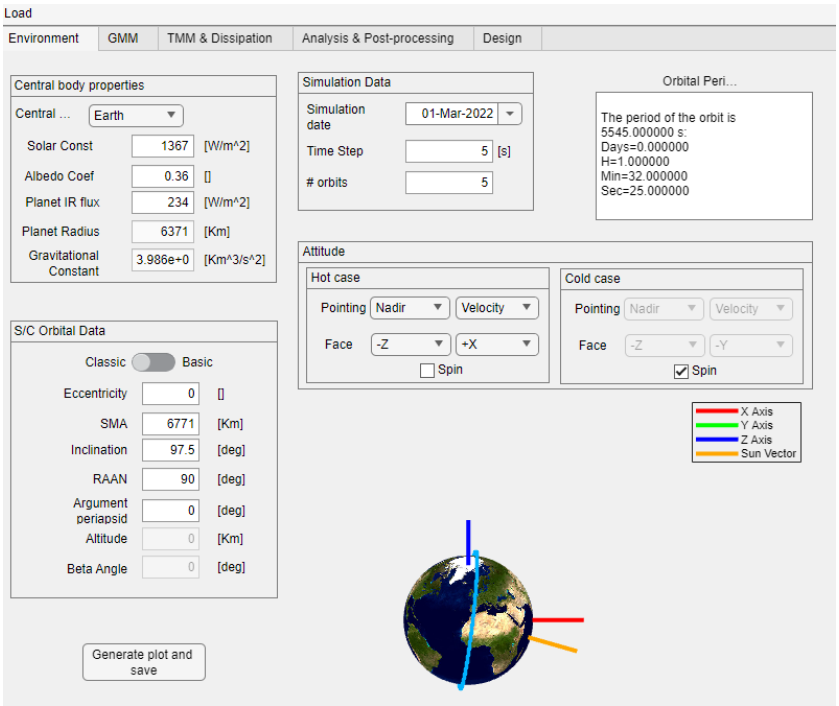


Figure 53: Environmental data of SROC simulations.

The GMM and TMM of SROC were already available from the previous work of the original creator of S2T2, Figure 54 shows an overview of the

geometries, as well as the node-node conductor between each item and the internal heat dissipations.

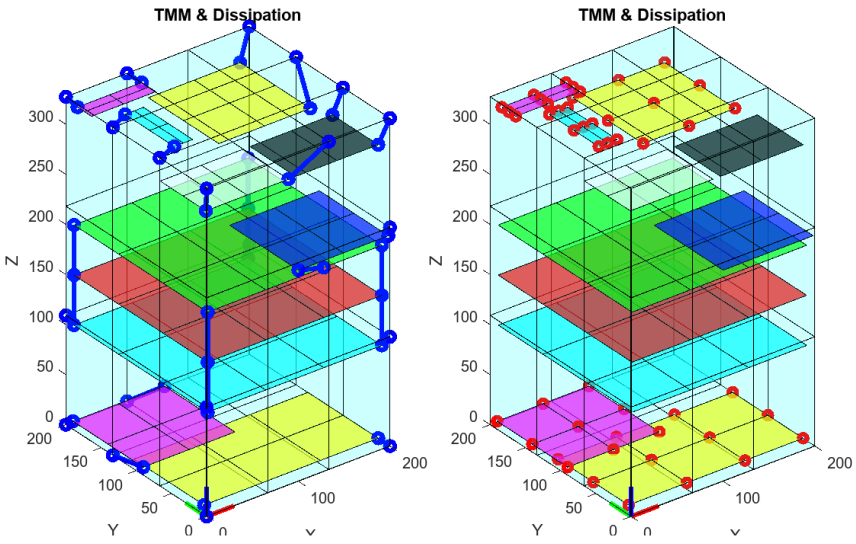


Figure 54: TMM of SROC, with additional conductions (left) and internal dissipations (right) [24]

Note that each component was originally modelled with board-like items because this was the only possibility available for the first version of S2T2. For the application presented in this thesis, to make fair comparisons, the GMM, TMM and environmental data were left unaltered.

After performing the transient analyses for the hot and cold case the settings of the optimization were entered. Following the same rationale of the work already available from Daniele Calvi, PhD, the optical properties ($\alpha_{ext}, \epsilon_{ext}, \epsilon_{int}$) of all the external faces were optimized, with bounds between 0 and 1, three copper thermal straps were optimized with a cross-sectional area between 0 mm^2 and 100 mm^2, starting from the payloads and the avionics item and ending on the structure of the satellite, and heaters were also applied both in the cold and hot case. These settings were identical to those chosen for the first design optimization performed on SROC by the creator of S2T2 [24], to better study the effect of the source code changes implemented. With these setting a total of 4 objectives are present: the distance from optimal item temperatures in the hot and cold case, and the total heater power, in the cold and hot case.

Analogously to the previous benchmarking study, the maximum number of function evaluations was set to 20000, with a population of size 200. A total of 10 optimization runs for each of the 9 algorithms were executed, to gather a

meaningful set of statistical samples, and make more informed comparisons using MOO performance metrics.

4.2.3 SROC Design Optimization Results

After the optimization runs the final solutions in the objective space for each algorithm and each run were gathered and used to generate different plots. Since there were 4 different objectives, there is no effective way to graphically represent the objective space in its entirety without losing some information. The approach chosen in this work was to plot the sum of the two temperature objectives on the horizontal axis (objective 1 + objective 2) and the sum of the two heater power objectives on the vertical axis (objective 3 + objective 4).

This way the trade-off aspect of the objective space plots is preserved visually, but the dominance relationships are lost in the plot because they depend on each of the 4 objectives simultaneously, thus points that in the graph are closer to the origin of the plot are not necessarily dominant to the others.

Figure 55 shows the final optimal solutions of one of the 10 runs performed, for every algorithm, similarly to the benchmarking application. Each different symbol identifies a different algorithm, and the black dots represent the non-dominated solutions, considering all runs and all algorithms. From the plot, a general trend can be easily deduced: even without knowing the single costs for the hot and cold case, the solutions closer to the y-axis are farther from the x-axis, meaning that to obtain a smaller distance from the optimal item temperatures the price to pay is the increased power consumption of the heaters; the vice-versa is also true. This effect generates a distribution that resembles a convex curve, which is typical of trade-off problems like the one encountered in the field of engineering.

Looking at Figure 55 some preliminary conclusions on the performances of the algorithms can be said, even without knowing the score of the single metrics. Certainly, the algorithm that got closer to the theoretical true Pareto front is MOEA/D because the majority of its design points are closer to the origin, however, the solution seems very clumped together, meaning that the optimizer focused more on the exploitation of its solutions rather than on the exploration of the design space. If the solutions are too clumped the user has a limited range of possibilities to choose from; in this case, MOEA/D solutions are very competitive in terms of the temperature objective, but if the user, after the optimization, wants

to move towards solutions with low power consumptions the set provided by MOEA/D may be unsatisfactory. RVEA and RSA on the contrary were a bit behind in matters of convergence but reached a more balanced final spread. The same could be said for MOPSO, which trails just behind. NGSA-III performed better than NGSA-II (*gamultiobj*) since it was designed to better handle many-objective optimization. MOGWO, GODLIKE and *paretosearch* performances instead were not at the same level as the other algorithms, with most of their solutions dominated by a large set of better ones.

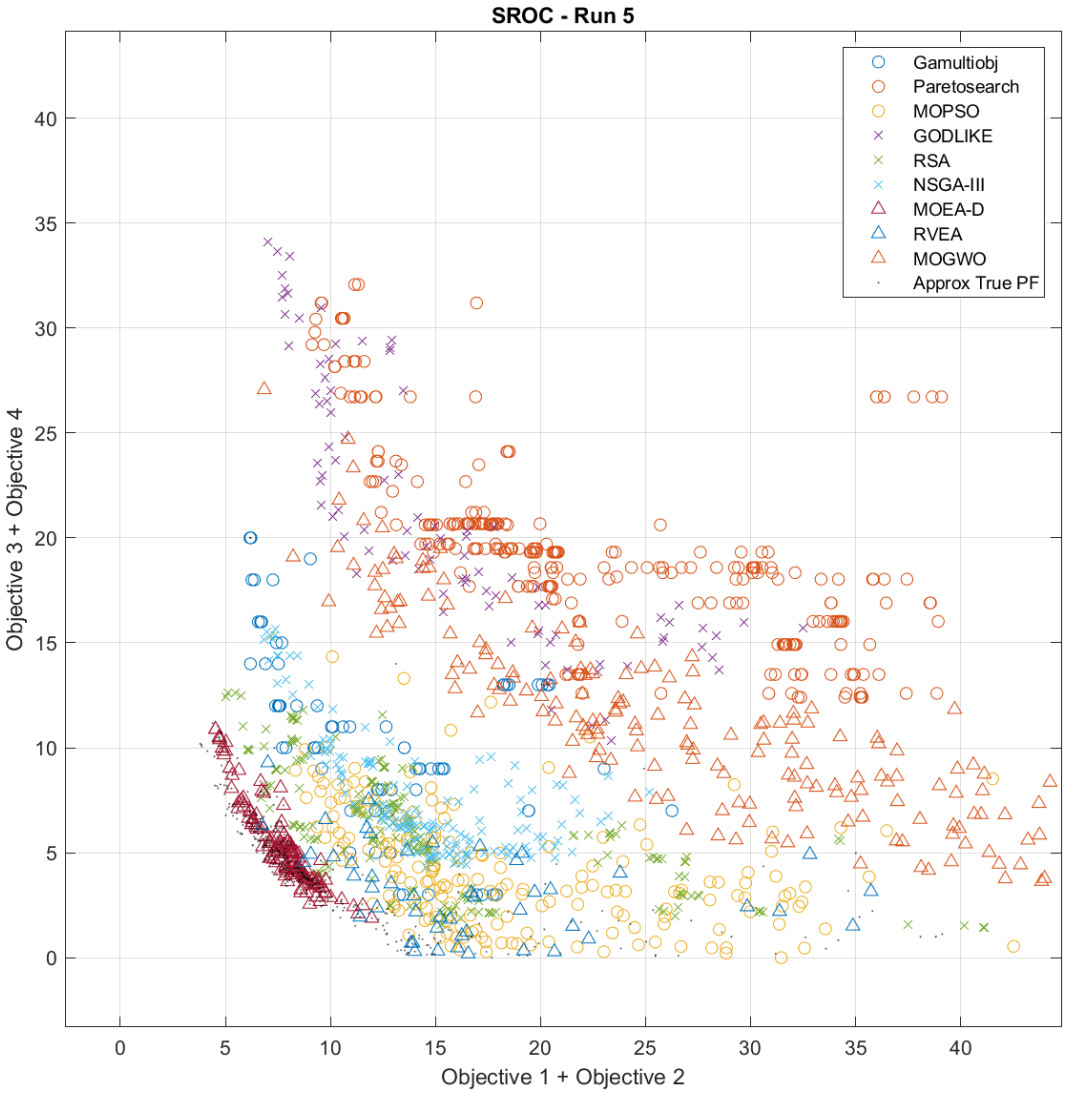


Figure 55: One of the runs of the SROC design optimization. Results from different algorithms are visible with different symbols, indicated in the legend on the top left. The black dots represent the elements of the set of non-dominated solutions, across all runs and algorithms.

Since the theoretical true Pareto front is not available like in the analytical benchmark functions, an approximation must be made: the non-dominated solution coming from the set that includes the design points of every algorithm and every run can be used as an approximation of the real Pareto front. This set is represented in Figure 55 by the black dots and is plotted by itself in Figure 56, using two visualization techniques. The plot on the left is generated by adding together the first with the second objective and the third with the fourth, while on the right only the first two objectives (hot and cold case temperatures, respectively) are used in the axis, which is also a common approach.

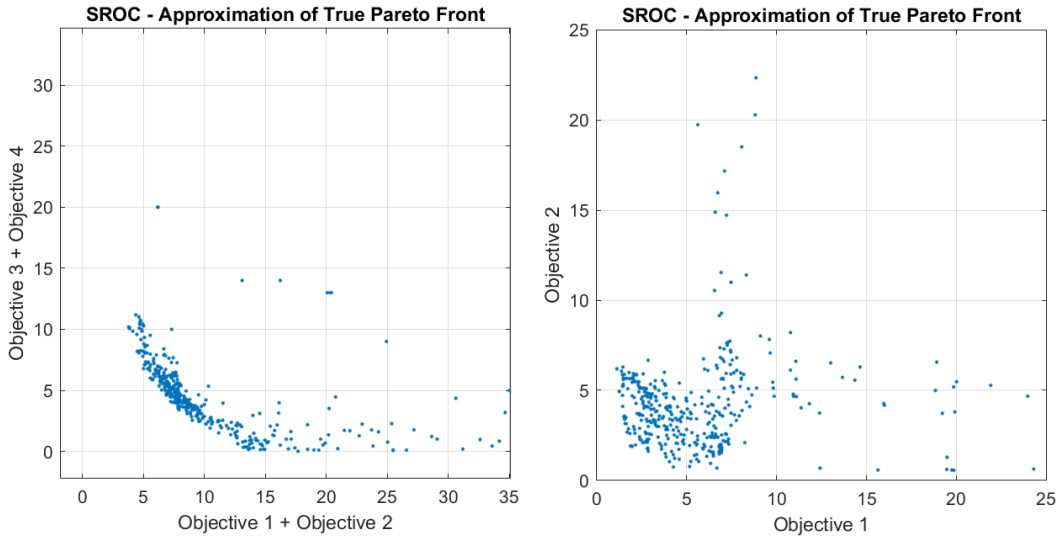


Figure 56: Approximation of the true Pareto front for the SROC thermal design. The data points correspond to the non-dominated solutions, across all runs and algorithms.

The plot on the left of Figure 56 shows even more clearly the effect of the trade-off curve. Another interesting aspect emerges: different objectives have different difficulties in reaching an arbitrarily low cost. While reaching cost zero for the heater power is simple (it is sufficient to set the design variables of the heart power to zero) the temperature objective is significantly harder and, for a given design, may not be possible to reach the desired steady-state temperature simply by controlling optical properties, conductance links and heater power. The plot on the right however shows that, at least for one case (either hot or cold) is possible to come very close. Unfortunately, for this design and this number of function evaluations, no solutions which have optimal temperatures both in the hot and cold case were found, with the best ones being in the 0°C to 5°C range of difference.

In a similar way as the benchmarking study, the same metrics were evaluated, using the approximated true Pareto front instead of the analytical true Pareto front. The other difference from before is that here only one "test problem" is present, the SROC spacecraft, which simplifies the analysis. The metrics were then normalized between 0 and 1 just like before and the average across the 10 runs was taken. Table 13 shows, for each algorithm, the actual values of the 7 metrics computed (the metric of computational time was not evaluated here because there is a fixed number of function evaluations; the cost function is the most resource-demanding part, thus the execution times were very similar and are not significantly affected by the algorithm used). The second to last row contains the weighted average of the score in the rows above (the same weights as before were used) and the last row shows the final ranking, based on that.

Table 13: Average values of performance metric for every algorithm.

	Alg.	1	2	3	4	5	6	7	8	9
Metric		NSGA-II	PS	MOPSO	GODLIKE	RSA	NSGA-III	MOEA/D	RVEA	MOGWO
W = 1	NR*	4.21E-02	0.00E+00	6.27E-02	0.00E+00	2.49E-02	2.09E-03	7.14E-01	1.54E-01	0.00E+00
W = 1	GD	5.09E-02	3.87E-02	2.94E-02	5.12E-02	2.65E-02	2.85E-02	6.47E-03	3.47E-02	3.97E-02
W = 5	IGD	1.92E-01	6.97E-01	9.59E-02	5.99E-01	9.63E-02	1.68E-01	6.94E-02	1.12E-01	3.52E-01
W = 1	MPFE	1.10E+00	1.07E+00	8.65E-01	1.26E+00	4.56E-01	3.37E-01	7.03E-02	2.99E-01	7.08E-01
W = 1	SP	2.48E-01	1.87E-01	1.28E-01	2.20E-01	9.16E-02	4.74E-02	1.14E-02	7.41E-02	8.73E-02
W = 1	MS*	8.60E-01	7.23E-01	9.11E-01	6.96E-01	6.57E-01	4.73E-01	3.88E-01	5.99E-01	8.51E-01
W = 5	HR	2.76E-01	8.26E-01	1.15E-01	7.35E-01	1.20E-01	3.37E-01	9.52E-02	9.59E-02	4.30E-01
AVG SCORE		0.6179	0.1462	0.8168	0.1946	0.8160	0.6784	0.9375	0.8243	0.5467
FINAL RANK		6	9	3	8	4	5	1	2	7

As predicted from the plot in Figure 55, MOEA/D obtained the highest scores and was classified first on the SROC leaderboard. The only real problem of MOEA/D is the clumping of its solutions, which was already noted in the benchmarking study. This meant that its MS value, which measures the spread of the solutions, was the lowest across all algorithms. This is typical of algorithms that lean towards exploitation rather than explorations: the final solutions are better, but only a fraction of the objective space is covered. RVEA had a similar problem, but because it found fewer overall non-dominated solutions its GD and IGD, which measures the distance from the true Pareto front values are lower. MOPSO and RSA obtained a very similar final score but while MOPSO outperformed RSA in terms of maximum spacing MS, RSA instead had a better MPFE score (which measures the maximum distance from the true Pareto front). All the other algorithms performed considerably worse, and their final score is negatively affected by their poor values of HR and IGD.

Table 14 summarizes the results of both the benchmarking process and the SROC case study. By averaging once again the score between these two a definitive ranking can be generated, this is visible in the second to last row.

Table 14: Final average scores of every algorithm in the two case studies.

	Algorithm	1	2	3	4	5	6	7	8	9
Case Study		NSGA-II	PS	MOPSO	GODLIKE	RSA	NSGA-III	MOEA/D	RVEA	MOGWO
1	BENCHMARK	0.3007	0.5455	0.7281	0.2690	0.8107	0.4817	0.7330	0.8497	0.4703
2	SROC	0.6179	0.1462	0.8168	0.1946	0.8160	0.6784	0.9375	0.8243	0.5467
AVG SCORE		0.4593	0.3458	0.7724	0.2318	0.8133	0.5800	0.8352	0.8370	0.5085
FINAL RANK		7	8	4	9	3	5	2	1	6
IMPLEMENTED?		YES	NO	YES	NO	YES	YES	YES	YES	YES

The last row of Table 14 contains the final decision regarding the implementation of the algorithms in the S2T2 application. The two worst-performing algorithms were left behind. The 7th algorithm in the ranking is the one used in the first version of S2T2: it was kept in the final version but because it is quite old and because it does not scale well with higher numbers of objectives, it is not recommended for the TCS design optimization of S2T2. The final top 3 algorithms were RVEA, MOEA/D and RSA, in this order. It is important to notice that two of them make use of reference vectors, which are an excellent tool to guide the optimization, improving the diversity of the solutions as well as increasing convergence speed, as a bonus. For future developments a hybridization of those top-performing algorithms could be explored, to balance out their strengths and limitations.

A final closing comparison between the run performed with the first version of S2T2 by the original author of the software and the approximated Pareto front generated is shown in Figure 57 and Figure 58. Although the design variables and the model were identical, in the new version the number of function evaluations of each run was about 10 times bigger, due to an improved computational complexity. For this reason, Figure 57 and Figure 58 are intended to be more of a showcase of the new capabilities of S2T2, rather than a direct one-to-one comparison.

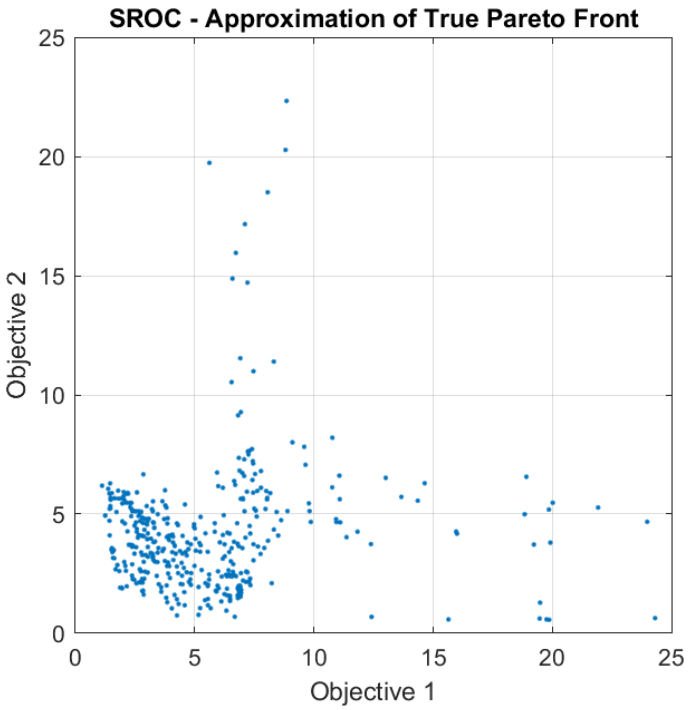


Figure 57: Approximation of the true Pareto Front, objectives 1 and 2.

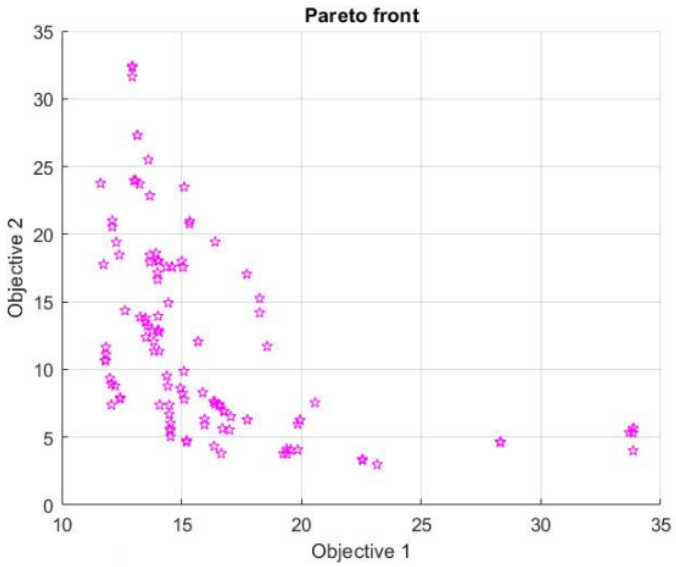


Figure 58: Pareto front obtained with the legacy implementation of S2T2.

4.3 Spei Satelles

In this section, the third case study is discussed and the role of S2T2 in the early design phases of the Spei Satelles mission is presented. The first sub-sections contain an overview of the mission and the SpeiSat CubeSat, and then some brief descriptions of the input for the thermal analyses performed are given. More space is devoted to the comments on the result and the TCS design optimization of SpeiSat. A conclusive part reports some specifications on the detailed thermal model used for the final design phases of SpeiSat, realized with the software C&R Thermal Desktop (abbreviated to TD), and some comments on the first temperature data available from orbit, a few days after the launch of June 12, 2023.

4.3.1 Spei Satelles Mission Overview

The Spei Satelles mission originates from the will to diffuse a message of hope by Pope Francis to all people in the world. This message of hope, shared for the first time on the 27th of March 2020 during the COVID pandemic and known as *Statio Orbis*, has been transcribed into a miniaturized chip in binary language by the Italian National Council of Research (Consiglio Nazionale delle Ricerche – CNR). The miniaturized chip is referred to as “Nanobook” and it is hosted onboard the SpeiSat spacecraft. Hence, the SpeiSat symbolically becomes “a guardian of hope”, as the name Spei Satelles translates from Latin.

Technically speaking, Spei Satelles is a telecommunications mission that sends messages that are received and shared with people through a network of amateur ground stations. Therefore, the main mission objective is to transmit sentences of hope from space to ground. The sentences are transmitted in Italian, English and Spanish.

Furthermore, as a secondary mission, the spacecraft aims at collecting data to characterize the CubeSat behaviour and the space environment. A Sensing Suite, equipped with an Inertial Measurement Unit, magnetometers and about 30 temperature sensors, is integrated within the spacecraft to fulfil the secondary mission objectives. The data collected will be used to: validate the thermal and attitude models that supported the analyses during the design, to assess the behaviour of several platform items, and to characterise the magnetic environment. The Spei Satelles mission has five objectives.

Primary mission objectives:

(1) To host the Nanobook and bring it to LEO.

(2) To transmit text messages of hope to ground stations. The messages are sentences collected in a file saved on the onboard computer memories. They are transmitted in three languages: Italian, English and Spanish.

Secondary mission objectives:

(3) To characterize the internal and external thermal environment of the spacecraft.

(4) To characterize the internal magnetic field of the spacecraft and map the Earth's magnetic field.

(5) To characterize the angular motion of the spacecraft.

ConOps and Mission Timeline

The description of the Concept of Operations (ConOps) is presented in Table 15, while the mission timeline is visible in Figure 59.

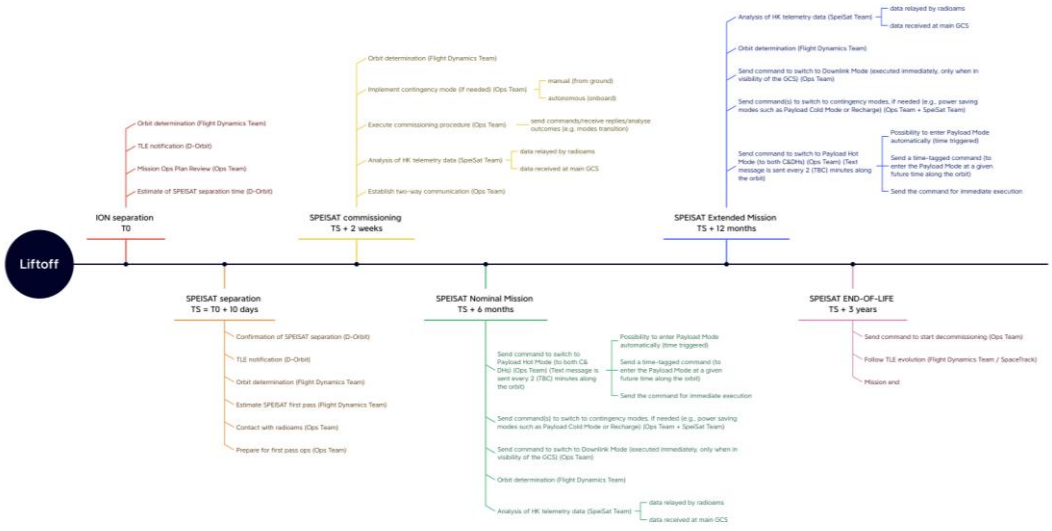


Figure 59: Spei Satelles mission timeline

Table 15: Spei Satelles high-level ConOps

Mission Phase	Mission Sub-phases	Description
Integration & pre-launch (IPLP)	- Transportation to launch base (TRP) - Integration in the PSL12U deployer (DIP, deployer integration phase) - Integration in the ION carrier (IIP, ION Integration Phase) Integration on Falcon9 LV (LIP, Launcher Integration Phase)	- Objective: to get ready for launch into orbit - Duration: < 1 month - Initial condition / start event: SpeiSat shipped to VSFB (CA, USA) - Final condition / end event: SpeiSat integrated into the PSL12U deployer onboard the ION carrier and installed on Falcon9 LV - Environment: VSFB
Launch and Early Operations Phase (LEOP)	- Launch Phase (LAP) - Pre-Separation Phase (PSP) - Separation Phase (SEP) - Commissioning Phase (CMP)	- Objective: to get ready to start the mission in orbit - Duration: < 3 months (target < 6 weeks) - Initial condition / start event: Falcon9 Liftoff - Final condition / end event: in-orbit commissioning completed - Environment: launch + orbit
Mission Operations Phase (MOP)	- Primary Mission Phase (PMP) - Extended Mission Phase (EMP)	- Objective: to execute the mission - Duration: > 6 months (target > 1 year) - Initial condition / start event: command from ground to start nominal operations - Final condition / end event: mission completion - Environment: orbit
End-of-Life Phase (ELP)	- Decommissioning Phase (DEP) - Disposal Phase (DSP)	- Objective: to dismiss the spacecraft after mission completion - Duration: < 25 years (target < 5 years) - Initial condition / start event: mission accomplished - Final condition / end event: SPEISAT burned up in Earth's atmosphere - Environment: orbit

4.3.2 SpeiSat Architecture and Configuration

The Spei Satelles satellite is a 3U CubeSat designed upon the platform developed at Politecnico di Torino in the framework of its CubeSat programme.

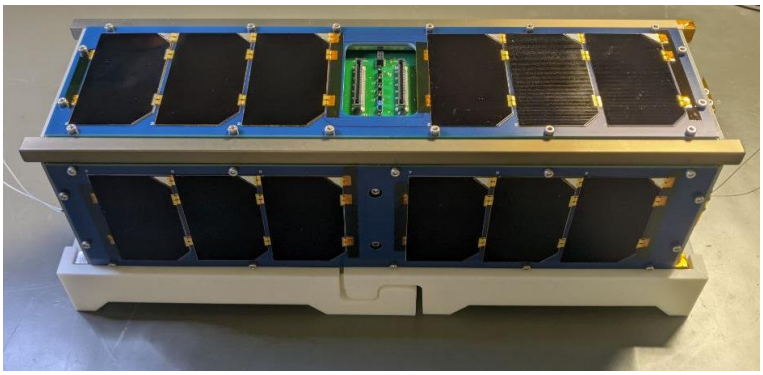


Figure 60: Final configuration of SpeiSat.

The spacecraft was designed to guarantee the full redundancy of the transmission function; therefore, the SpeiSat is equipped with two independent C&DHs and communication systems. The ensemble of one C&DH and one communication system constitutes a bus. The two buses are interfaced and coordinated thanks to an interface and distribution system of onboard functions (Backplane). The two buses are independent for most of the functions; however, bus 1 can turn off bus 2 to save power during specific mission phases. Furthermore, the two buses alternate for the transmission and their arbitration is coordinated by the arbitration circuit located on the Backplane.

The Backplane also performs the function to distribute power among all subsystems, interfacing the Electrical Power System with all other components. The EPS is constituted of four solar panels body mounted on the external faces of the spacecraft and a lithium-ion battery pack. The spacecraft is equipped with a magnetic attitude stabilization system made of permanent magnets and hysteresis rods to stabilise the attitude and dump attitude oscillations. A Sensing Suite equipped with an IMU and up to 32 temperature sensors is used to monitor the state of health of the platform and to collect the data required to fulfil the scientific goals of the secondary mission.

As for the payload, the platform symbolically hosts the nanobook provided by CNR, while the sentences to be transmitted are saved in a file stored in both the C&DHs' memory. These systems are installed in an Al 7075 aluminium alloy structure suitably treated with SurTec plus hard anodizing where required (e.g., rails).

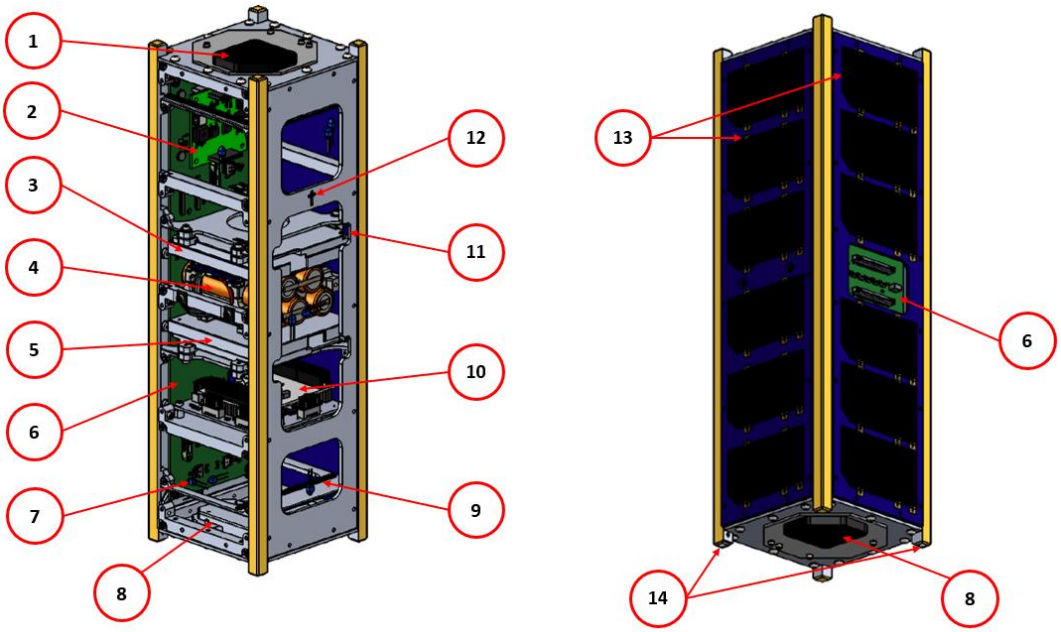


Figure 61: Views of the CAD model of SpeiSat, elements 1 to 14 listed in Table 16.

Table 16: List of SpeiSat main elements.

Element #	Quantity	Component
1	1	ComSys 2
2	1	DET
3	1	Command & Data Handling 2
4	1	Battery
5	1	Command & Data Handling 1
6	1	Backplane
7	32	Temperature sensor
8	1	ComSys 1
9	6	Hysteresis rod
10	1	Sensing suite
11	2	Permanent Magnet
12	1	Nanobook
13	4	Solar Panels
14	2	Deployment Switches

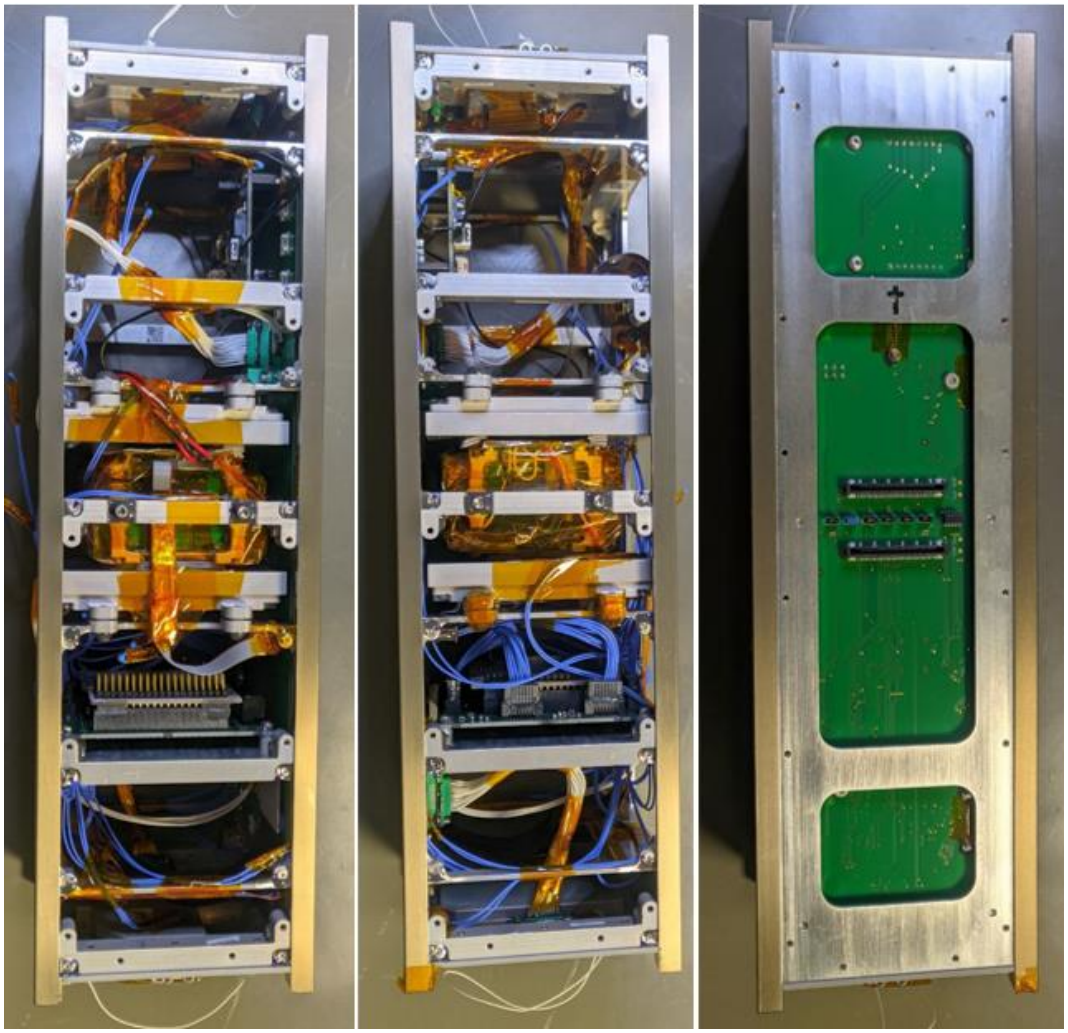


Figure 62: Internal views of SpeiSat (from left to right: -Y face, +Y face, +X face).

4.3.3 SpeiSat Thermal Analysis Inputs

The first step for thermal analysis concerns defining the temperature requirements for each component installed on board. in particular, it is necessary to identify not only the operative temperature ranges of the components but also the survival temperature ranges. Operative temperatures indicate the limits in which the component can operate, while survival temperatures indicate the limits in which the component will not fail if it is kept off. Table 17 collects all temperature limits for each component.

Table 17: Operational and Survival Temperatures of SpeiSat.

Component	Operational MIN [°C]	Operational MAX [°C]	Survival MIN [°C]	Survival MAX [°C]
Antenna 1	-100	100	-120	120
Antenna 2	-100	100	-120	120
Backplane	-40	85	-55	125
Battery	0	40	-10	50
C&DH 1	-30	45	-40	75
C&DH 2	-30	45	-40	75
Solar Cells	-55	125	-65	150
ComSys 1	-30	60	-40	70
ComSys 2	-30	60	-40	70
DET	-40	80	-50	90
Sensing Suite	-40	80	-50	90

At this point, it is necessary to identify the two thermal worst-case scenarios. This definition is derived from the analysis of the different mission phases and the comparison of the dissipations obtained in the different operative modes. Two extreme cases called Hot Case and Cold Case were identified.

In the hot case, the S/C experiences the highest environmental heating, and it is in the operative mode with the highest power consumption, the Payload Hot operative mode. In the cold case instead, the S/C is exposed to the lowest environmental heating of its life, and it is in the operative mode with the lowest power consumption, the Recharge operative mode. Figure 63 shows the different orbits for the two case studies, generated with the S2T2 software.

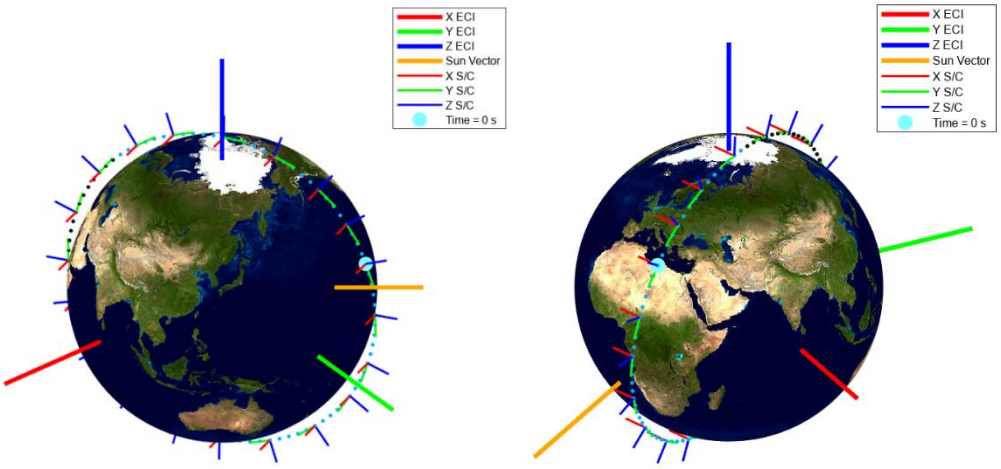


Figure 63: SpeiSat Cold Case Orbit (left), Hot Case Orbit (right)

Table 18 collects the simulation data used for the analysis in the two cases. In this table are also reported the S/C attitudes, which in the early stages of the project were considered fixed. At a later stage, a time-varying attitude was used, derived from the trajectory analysis.

Table 18: SpeiSat Cold and Hot Case simulation data.

Name	Aphelion (Cold Case)	Perihelion (Hot Case)
Date/Time [GMT]	5 Jul 2023 00:00:00.000	2 Jan 2024 00:00:00.000
Semi-major Axis (km)	6900.533728	6879.903852
Eccentricity	0.001612	0.000392
Inclination (deg)	97.388	97.633
RAAN (deg)	297.476	117.232
Arg of Perigee (deg)	156.415	147.934
True Anomaly (deg)	40.592	249.338
Mean Anomaly (deg)	40.472	249.38
Solar Flux [W/m^2]	1322	1414
Albedo	0.25	0.4
Earth IR Flux [W/m^2]	218	243
Nadir Attitude	Z-	Z-
Velocity Attitude	Y+	Y+

The selected operative modes for the two case studies are presented below, Payload Hot Mode is assigned to the hot case while Recharge mode is assigned to the cold case:

- Payload Hot Mode: Both buses' onboard computers are active, and the sensor suite is active and collecting data. Both buses send four consecutive messages. The first three messages contain the same payload hopeful sentence in Italian, English, and Spanish, while the fourth message contains a telemetry packet made up of the system and the sensor suite telemetry data. The four messages are alternated between the two buses so that each bus sends one sequence every 2 minutes.
- Recharge Mode: Only bus 1 onboard computer is active, while bus 2 is shut down. The sensing suite is active and collects sensor data. bus 1 transmits telemetry packets which contain the system telemetry data, the payload telemetry data, and the sensing suite data every 2 minutes. This mode of operation is triggered automatically when the battery voltage goes below 11.4 V. This mode of operation is timed and automatically reverts to the previous operative mode when the timer of 11 hours runs out. Its duration is designed so that it reverts to the previous mode when the battery is at full charge.

Table 19 lists the power consumption for the two operative modes with also the duty cycles for each different phase and the time-average power dissipation.

Table 19: Recharge and Payload Hot modes power consumption.

	Cold P ON [W]	Cold P OFF [W]	Cold DC ON [%]	Cold DC OFF [%]	Cold P AVG [W]	Hot P ON [W]	Hot P OFF [W]	Hot DC ON [%]	Hot DC OFF [%]	Hot P AVG [W]
ComSys1	3.5	0.1	0.2	99.8	0.1	3.5	0.1	3.4	96.6	0.2
ComSys2	0.3	0.1	100	0	0.3	3.5	0.1	3.4	96.6	0.2
DET	0	0	0	0	0	0	0	0	0	0
Backplane	0.4	0	100	0	0.4	0.4	0	100	0	0.4
C&DH1	0.9	0	100	0	0.9	0.9	0	100	0	0.9
C&DH2	0	0	0	0	0	0.9	0	100	0	0.9
Sensing Suite	0.2	0	100	0	0.2	0.2	0	100	0	0.2
Battery	0	0	0	0	0	0	0	0	0	0

The other inputs needed for the thermal analysis are the mechanical and optical properties of the components on the satellite. in particular, the optical properties of each surface finish are grouped in Table 20. In the early stages of the project, using S2T2 software, surface-weighted average optical properties were calculated, the formulas are given below [4].

$$\alpha_{effective} = \sum_{j=1}^{N} \left(\frac{A_j}{A_{total}}\right) \alpha_j \quad \epsilon_{effective} = \sum_{j=1}^{N} \left(\frac{A_j}{A_{total}}\right) \varepsilon_j \qquad (61)$$

Where A_j is the portion of the external facing surface area of the j-th component, A_{total} is the sum of every A_j, α_j is the solar absorptivity of the single material, ϵ_j is the IR emissivity of the single material.

These formulas were used, for example, to determine the optical properties of the side faces of the satellite. They are formed by the overlapping of three different layers, the first being the structure, the second being the PCB, which represents the base of the solar panel, onto which the six solar cells are soldered, which constitute the third layer, as shown in Figure 64. This simplification was used inside the S2T2 software and the first iteration of the model created on TD. For the last iteration performed on TD, individual layers were modelled as shown in Figure 64.

Table 20: Optical Properties for each surface finish of SpeiSat.

Material	Solar Absorptivity	IR Emissivity
Aluminium Alodine	0.42	0.06
SurTec 650	0.34	0.05
Black Anodized Aluminium	0.72	0.82
PCB Blue	0.89	0.9
Solar Cell	0.663705	0.85
PCB Green	0.88	0.7
Bare Aluminium	0.13	0.06

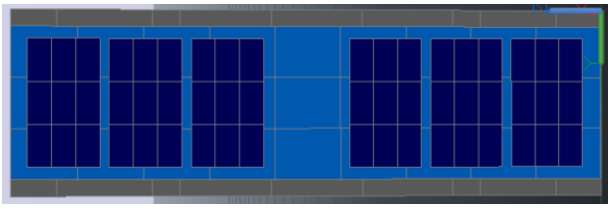


Figure 64: Lateral outside -X face of the SpeiSat TD model, with each layer modelled.

The following formulas, which take into account the different materials the components are made of, were used to calculate the average mechanical properties of each item [4].

$$c_{p_{effective}} = \sum_{j=1}^{N} \left(\frac{m_j}{m_{total}}\right) c_{p_j} \qquad \rho_{effective} = \frac{m_{real}}{V_{modelled}} \tag{62}$$

Where: c_{p_j} is the specific heat of the j-th component material by which the item is made, m_j is the mass of the j-th component, m_{total} is the total mass of the modelled component, m_{real} is the real measured mass of the component, $V_{modelled}$ is the total volume of the primitive shapes used to model the layers.

Regarding thermal conductivity, it is necessary to distinguish between isotropic materials and laminated materials. In the isotropic materials k has a constant value over the whole volume, while in laminated materials, on the other hand, there are two different conductivities, k_{xy} pertains to in-plane conductivity, while k_z pertains to out-of-plane conductivity.

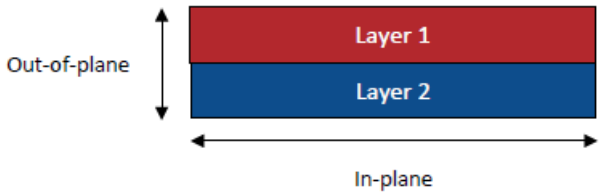


Figure 65: In-plane versus Out-of-plane directions.

Specifically, all PCBs consist of one or more layers of FR4, two or more layers of copper, and two solder mask layers, which characterize the optical properties. The following formulas were used to calculate these two conductivity contributions.

$$k_{xy_{effective}} = \frac{1}{t_{total}} \sum_{j=1}^{N} (k_j t_j) \qquad k_{z_{effective}} = \frac{t_{total}}{\sum_{j=1}^{N} \left(\frac{t_j}{k_j}\right)} \tag{63}$$

Where t_j is the thickness of the j-th layer, t_{total} is the sum of every t_j, k_j is the isotropic conductivity of the j-th layer.

Table 21 contains the values of the mechanical properties of the single material used, while the materials assigned to each component are reported in Table 22.

Table 21: Mechanical properties of each material of SpeiSat.

Material	Conductivity (in-plane) [W/m/K]	Conductivity (out-of-plane) [W/m/K]	Density [kg/m^3]	Specific Heat [J/kg/K]
Aluminium 7075	130	130	2810	960
PCB Skin	5.80903	0.2536	1952.2	586.146
PCB DET	28.8981	0.2699	2376.67	541.356
PCB Backplane	28.8981	0.2699	2376.67	541.356
PCB Sensing Suite	29.2744	0.27027	2383.58	540.758
PCB C&DH	28.8981	0.2699	2376.67	541.356
PCB ComSys	28.8981	0.2699	2376.67	541.356
Solar Cells	0.00329	60.6	5320	324
Battery	34.53	34.53	3002.28	667.37

Table 22: Materials assigned to each component of SpeiSat.

Item	Material Properties	Surface Finishes
Structure	Aluminium 7575	SurTec 650
ComSys1	PCB ComSys	PCB White
ComSys2	PCB ComSys	PCB White
DET	PCB DET	PCB Green
Backplane	PCB Back Plane	PCB Green
C&DH1	PCB C&DH	PCB Blue
C&DH2	PCB C&DH	PCB Blue
Sensing Suite	PCB Singer	PCB Green
Battery	Battery	SurTec 650 + PCB Blue (+Z face)

Once the various inputs for thermal analysis were collected, the next step was to create the satellite GMM. Table 23 groups the type of item used to model the geometries.

Table 23: SpeiSat items modelled with S2T2.

Component	Type of Item	Colour
Structure	Empty box	Light cyan
ComSys	Solar panel	Blue
DET	Board	Green
Backplane	Board	Green
C&DH	Board	Magenta
Sensing Suite	Boards	Green
Battery	Parallelepiped	Yellow
Shelfs	Board	Black
Case of C&DH	Boards	White
Battery Stiffeners	Parallelepipeds	White

Conductive connections between the various components were modelled through node-to-node conductors, as for the dissipations, they are evenly distributed over all the nodes of each heat-dissipating item, as shown in Figure 66.

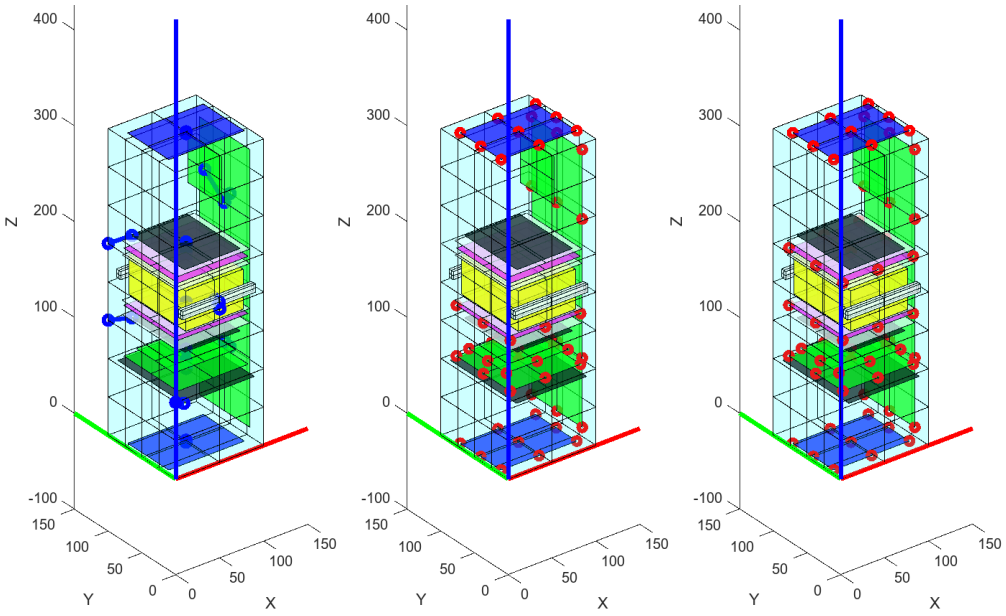


Figure 66: Additional conduction (left), Cold case dissipation (middle), Hot case dissipations (right).

4.3.4 SpeiSat Thermal Analysis Results

After the definition of all the inputs and the finalization of the thermal model in S2T2 two transient analyses, one for the hot case and one for the cold case were performed. To validate the S2T2 model and the new release of the S2T2 application in general a TD model was created, trying to keep this as close as possible to the representation of S2T2. In particular, the validation model created in TD, compared to the S2T2 model, features:

- Same representation of internal components, using identical geometries, which naturally leads to the same number of nodes with the same distribution between the two models.
- Same optical and thermophysical properties, considering isotropic materials.
- Same representation of the structure and solar panels, using average uniform properties.
- Same Keplerian orbit.
- Same constant attitude.
- Same node-node conductors, with the same starting node, ending node and conductance value.

- Same constant, time-average heat loads.

The only difference between the two models lies in the computation of the S/C-Earth view factors: while in TD this is intrinsically performed using a Monte Carlo ray tracing method, in S2T2 it is performed analytically, using the approximation of flat plate absorbing and emitting on one side [35] and solving the problem of satellite self-shadowing by casting a single ray from every surface element of the S/C to the centre of the heat source, which in this case is the Earth. If these rays, on their path, intersect other satellite surfaces they determine the shadowed elements, for which the view factor with the heat source is set to 0. This approximation allows to simplify the calculation of the environmental heat sources, saving computational time and code development time, while simultaneously maintaining a high level of accuracy, as demonstrated in the following results.

The temperature trends of the most important component of SpeiSat, namely the battery, the two C&DHs, the Sensing Suite and the two ComSys, are displayed below in Figure 67. The results from TD (red) and S2T2 (blue) are overlapped, to show the high level of accuracy reached by the new version of S2T2, even with the high number of geometries of SpeiSat, arranged in a complex configuration. The plots on the left are relative to the cold case, while the ones on the right are relative to the hot case.

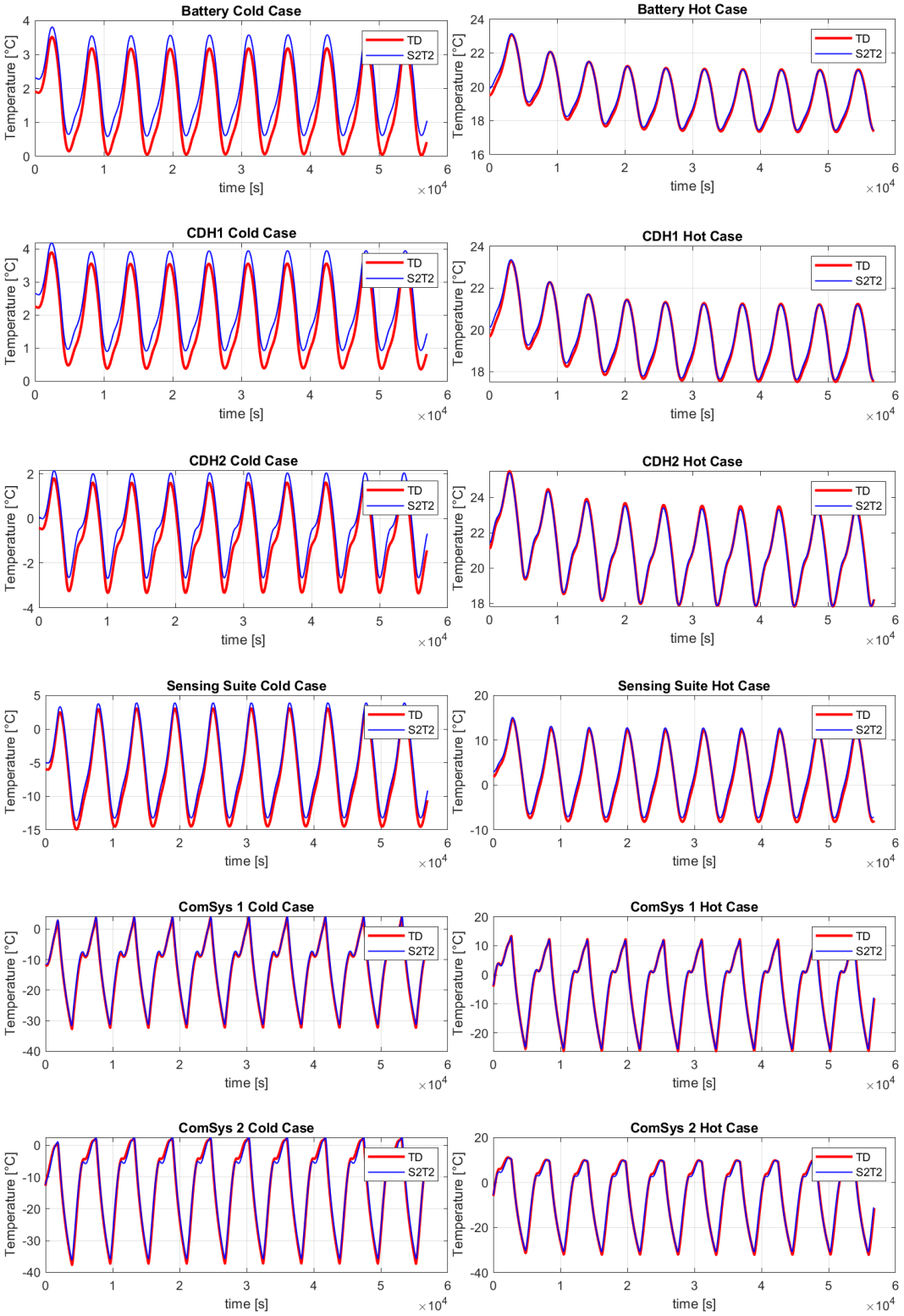


Figure 67: Temperature trends comparisons between S2T2 and TD, Cold Case (left) and Hot Case (right).

For the majority of the components, it is evident from the plots that the temperatures never exceed a difference of 1°C, with the only exception of the two ComSys which, are subject to much wider oscillation since they are external equipment, and thus the difference between the two model increases too, reaching an absolute error of ~1.5°C. Overall another positive aspect of the S2T2 simulations is the shape of the temperature oscillation, which closely matches the one computed by the commercial software.

To better understand the entity of the accuracy of S2T2 Table 24 and Table 25 were generated. For each of the two cases and each component the minimum and maximum computed temperatures are listed, with both software (excluding the first 5 orbits, to avoid convergence error from the steady-state temperatures). In the second to last column, the absolute error is given in terms of the temperature difference between S2T2 and TD, while the last column contains the relative error of S2T2 compared to TD, computed with the following formula:

$$\text{Relative Error} = \left| \frac{\Delta T}{T_{max}^{TD} - T_{min}^{TD}} \right| \tag{64}$$

Where ΔT is the absolute temperature error between S2T2 and TD ($\Delta T = |T_{max}^{S2T2} - T_{max}^{TD}|$ or $\Delta T = |T_{min}^{S2T2} - T_{min}^{TD}|$) and T_{max}, T_{min} are the maximum, and minimum computed temperatures by the two software.

Table 24: Comparison between S2T2 and TD for the Cold case.

Cold Case					
Item		TD [°C]	S2T2 [°C]	ΔT [°C]	Relative Error [%]
Battery	Min	0.02	0.60	0.58	18.29
	Max	3.18	3.58	0.4	12.74
C&DH1	Min	0.36	0.93	0.57	17.71
	Max	3.55	3.94	0.39	12.27
C&DH2	Min	-3.35	-2.67	0.68	13.77
	Max	1.60	2.04	0.44	8.89
Sensing Suite	Min	-14.58	-13.27	1.31	7.42
	Max	3.08	3.92	0.84	4.74
ComSys1	Min	-32.32	-31.31	1.01	2.8
	Max	3.72	4.06	0.35	0.96
ComSys2	Min	-37.30	-35.83	1.47	3.73
	Max	2.07	2.39	0.33	0.83

Table 25: Comparison between S2T2 and TD for the Hot case.

Hot Case					
Item		TD [°C]	S2T2 [°C]	ΔT [°C]	Relative Error [%]
Battery	Min	17.32	17.43	0.11	3.03
	Max	21.05	21.01	0.04	1.04
C&DH1	Min	17.50	17.60	0.11	2.82
	Max	21.27	21.23	0.04	1.13
C&DH2	Min	17.83	17.84	0.01	0.24
	Max	23.52	23.35	0.17	2.99
Sensing Suite	Min	-8.32	-7.38	0.93	4.54
	Max	12.26	12.75	0.49	2.4
ComSys1	Min	-26.33	-25.64	0.69	1.79
	Max	12.25	11.96	0.3	0.77
ComSys2	Min	-32.23	-30.98	1.24	2.95
	Max	9.90	9.85	0.05	0.11

It is interesting to note that the magnitude of the absolute error depends on the amplitude of the oscillations: for many components the larger the oscillation is, the larger the ΔT. This can be explained by considering the different approaches for the computation of the environmental heat sources. The approximated method of S2T2 carries a small error in the value of incoming heat, mainly in the heat coming from the central planet. The effect of this error is then amplified when the components are subject to high-temperature variations, for example when they are exposed to highly variable heat exchanges, such as the external components of a satellite in an orbit with an alternation between sunlight and eclipse, like SpeiSat.

On the other hand, the relative error depends heavily on the difference between the peaks and valleys of the temperature trends (it is the term on the denominator of the formula), thus it can be predicted that the items that experience lower oscillations will have, on average a higher relative error. This is the case of the Battery and C&DHs which have relative errors higher than 10%. Due to TCS design choices, those components were conductively insulated from the rest of the spacecraft, using low-conductivity polymeric bushes and washers on the bolts; this has the beneficial effect of dampening the temperature oscillations, by limiting the amount of heat exchanged at their interfaces.

To further investigate the effects of this insulation on the internal components and the overall temperature distribution of the satellite, different heatmaps were generated with S2T2. Below are displayed 3 different views (one for each row) of the same extreme instants: on the left the temperature distribution of the coldest

instant of the cold case simulation is shown, while on the right the hottest moment of the hot case simulation is shown instead. The top two images are an external view of SpeiSat, the middle two show the internal components, while the bottom ones combine the internal view with the external one, using transparency (Figure 68, Figure 69 and Figure 70, respectively).

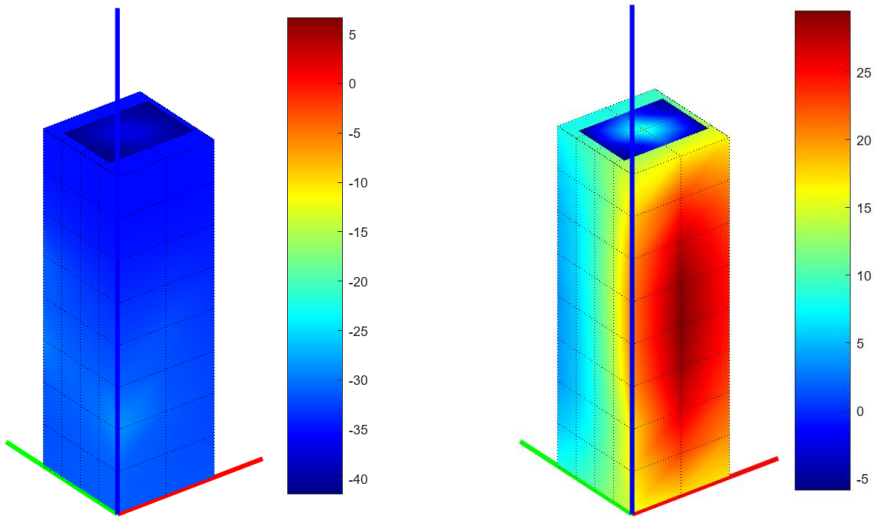


Figure 68: External heatmap, Cold case (left), Hot case (right).

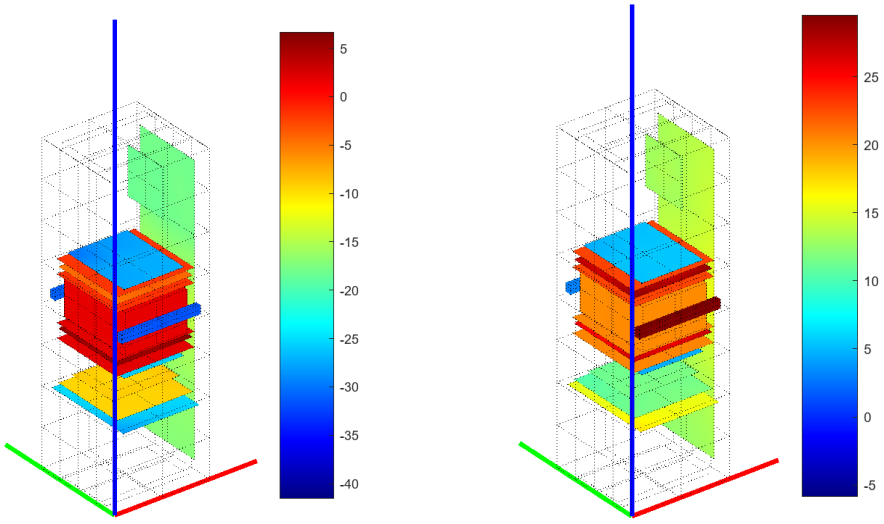


Figure 69: Internal heatmap, Cold case (left), Hot case (right).

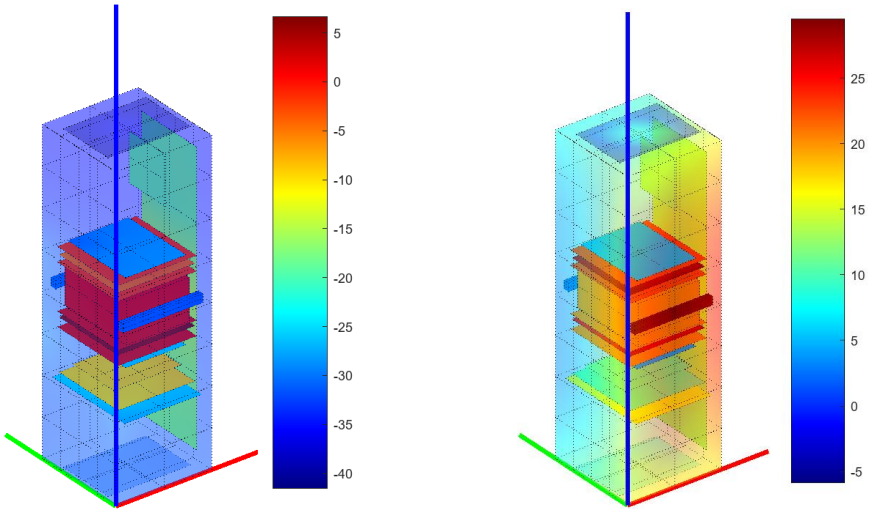


Figure 70: External-Internal view heat map, Cold case (left), Hot case (right).

From the previous heatmaps is clear how the insulation plays a fundamental role in keeping the temperature of the battery above its operative temperature: this was the rationale behind the design choice. By creating a central hot "core" insulated from the rest, the heat produced by the C&DHs is used to keep the battery warm, always above 5°C without ever activating the integrated 4W battery heaters. To favour the heat transfer between the two C&DHs and the battery (which does not dissipate heat on its own) a conductive interface material, a thermal pad, was used to conductively couple these components.

Thanks to these design choices the temperatures of the battery, the C&DHs and the aluminium cases of the C&DHs reach similar values, with low amplitude oscillations. Instead, the bolted interfaces of these components with the primary structure experience high-temperature differences because of the insulating elements used.

4.3.5 SpeiSat Design Optimization

To test the design optimization features of S2T2 several multi-objective optimization runs were performed. It was chosen to focus the analysis on the cold case rather than the hot case, because it was the most challenging one for SpeiSat, since the very first phases of the design. To optimize the TCS of the satellite 3 different degrees of freedom were available:

(1) The optical properties of the structure of the satellite can be changed to a certain extent: the external faces chosen for the optimization were the Z+ and Z- faces of the satellite, where the surface finishes of the aluminium (α_{ext} and ϵ_{ext}) part can be selected with more freedom, the lateral faces instead are more constrained, due to the presence of solar panels and solar cells. For the internal side of the structure, the emissivity ϵ_{int} was optimized for every one of the six faces. The bounds set for these variables are 0.05 (lower bound) and 0.9 (upper bound).

(2) The possibility of adding up to 2 additional conductance links, possibly in the form of thermal straps, was evaluated: the starting item group of the link was set to the following items: primary Structure, DET, Backplane, Sensing Suite, and Secondary Structure elements. The ending group was set to be equal to the starting group; the items were selected based on the ease of applying a thermal strap on them. The bounds for the cross-sectional area of the link were set to $0\ mm^2$ and $100\ mm^2$, while the conductivity of the link entered was $385\ W/m/K$ (copper conductivity).

(3) Heaters with constant power were optimized for the two coldest components, ComSys1 and ComSys2, as well as for the battery, to investigate how much power would be required to keep it near the middle of its operative range. The ComSys were allowed a maximum power of $1\ W$, while the battery's upper bound was set to $4\ W$, to match the integrated heater power of the equipment available during design.

The optimizations were conducted with the three best-performing algorithms implemented in S2T2: RVEA, MOEA/D and RSA. Each optimization was conducted with a maximum of 20000 cost function evaluation, a population size of 200, a time step of $60\ s$ for the post-processing of the optimal solutions and 5 orbits for the transient analysis. To avoid losing solutions in the post-processing the pruning of the transient result exceeding operative temperature was not performed, however, the post-processing was conducted only on the solution on the final Pareto front of each optimization run. After entering all the correct inputs, the optimization took about $400 \div 500\ s$ for each algorithm and $\sim 30\ s$ for each transient analysis performed in the post-processing.

RVEA returned a total of 73 solutions, 48 of which were non-dominated across all algorithms, MOEA/D returned 200 solutions with 173 non-dominated ones and RSA outputted 200 solutions, 53 of which non-dominated. From this

number, it can be seen that every algorithm managed to find a niche of dominant solutions, with MOEA/D covering the largest portion of the final Pareto front. The three objectives to be optimized were the weighted average distance from optimal temperatures (defined as the midpoint between minimum and maximum operative temperatures), the total heater power and the total volume of copper needed to create the conductance links positioned by the optimizers. Figure 71 shows the distribution of solutions in the objective space, comparing the three algorithms. The visualization is 2-dimensional for ease of interpretation: the third objective, the volume of straps material is not plotted.

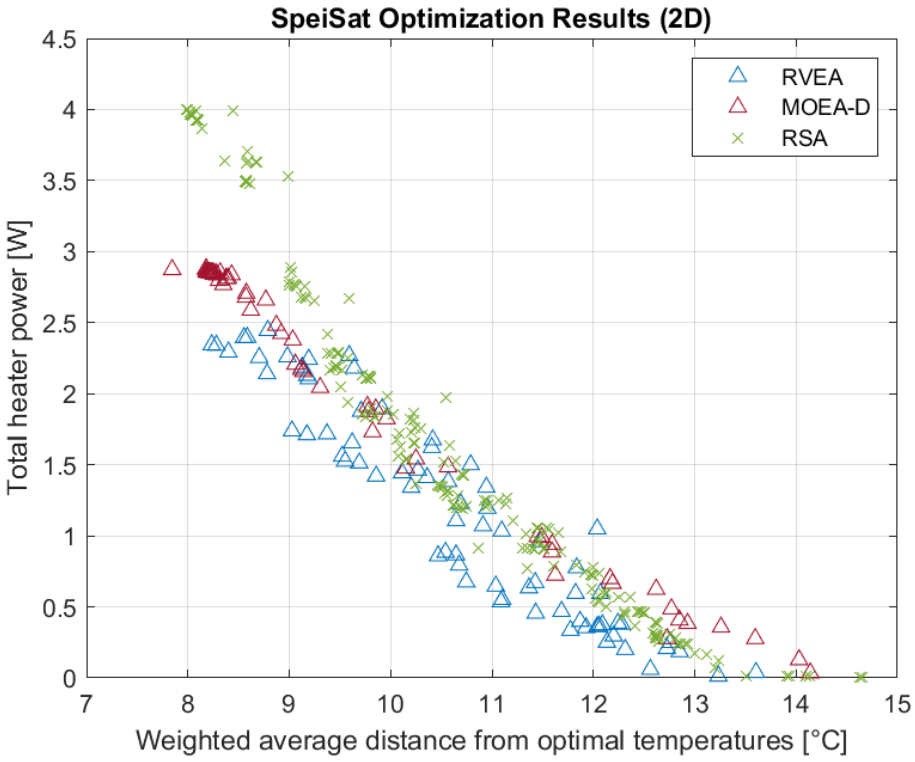


Figure 71: First two objectives (distance from optimal temperatures and total heater power) of the final Pareto fronts generated by RVEA, MOEA/D and RSA in the SpeiSat optimization.

Overall RVEA was confirmed to be the best algorithm for finding a high fraction of non-dominated solutions, however in this optimization the cardinality of the final solution set of RVEA was lower than the other, this could be fixed by tuning the parameters of the algorithms for the specific optimization performed. MOEA/D found the highest number of non-dominated solutions, however, because it does not use reference vectors like the other two algorithms it suffers from a clustering perspective: many solutions are clumped in the top-left portion of the objective space and the general distribution is unbalanced, with many large

holes towards the centre of the objective space. RSA, thanks to its adaptive normalization of the archive and its extensive use of reference vectors managed to arrive at a highly balanced distribution, outperforming the other two algorithms in terms of spacing of the solutions. On the other hand, it showed more difficulty in advancing towards low costs solutions compared to RVEA and MOEA/D and ended with a lower portion of dominated hypervolume.

To compare in a quantitative way the performance of the SpeiSat case study 4 performance metrics were evaluated: NR, IGD, MS and HR. The results are displayed in Table 26 (note that while IGD and HR are better the lower the values are, for the other two, marked with an asterisk, the score should be maximized). The best values for each metric are highlighted in bold text.

Table 26: Performance metrics of the 3 algorithms used in the SpeiSat design optimization.

Algorithm	NR*	IGD	MS*	HR
RVEA	0.1745	0.08764	0.8152	**0.02720**
MOEA/D	**0.6291**	0.1170	0.6758	0.08957
RSA	0.1964	**0.05769**	**0.9210**	0.10684

The quantitative results confirm the observation made from the visual investigation of the objective space: MOEA/D won on the fraction of final non-dominated solutions, RVEA was the best one in terms of Hypervolume and Hypervolume Ratio, while RSA finished on top in terms of spacing and distribution, as witnessed by the MS and IGD metrics.

Two plots of the final non-dominated design points, across all algorithms are given below; in Figure 72 there is a plot with the same axis as before, while in Figure 73 there is a 3-dimensional plot which simultaneously shows the cost of the optimal solution for each of the three objectives.

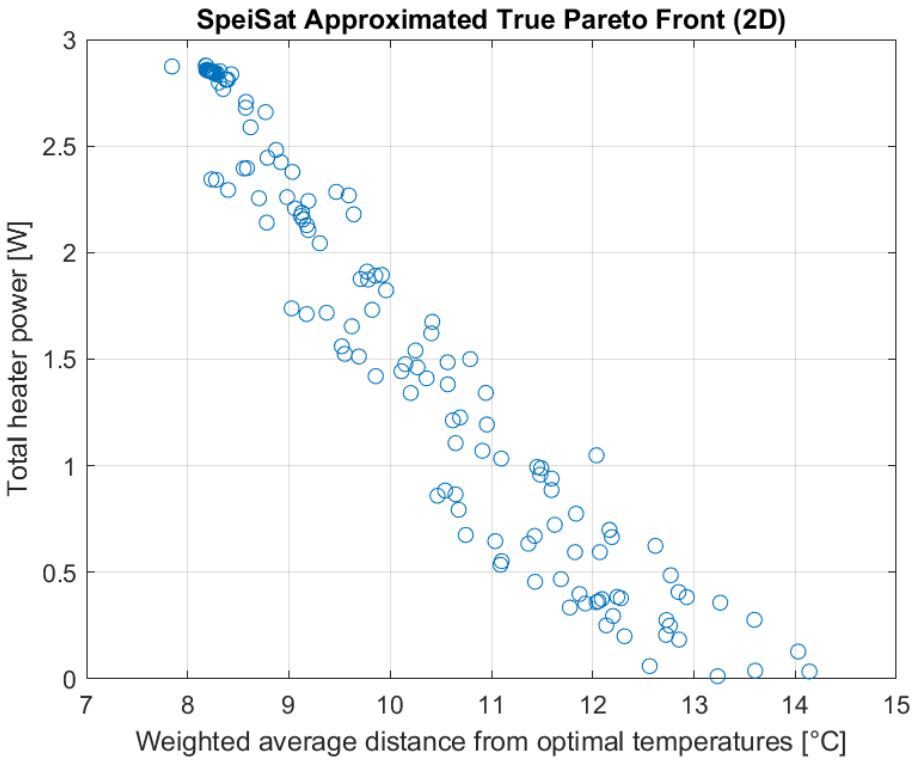


Figure 72: 2-dimensional representation of the approximated true Pareto front for the SpeiSat design optimization. The data points correspond to the non-dominated solutions, across all algorithms.

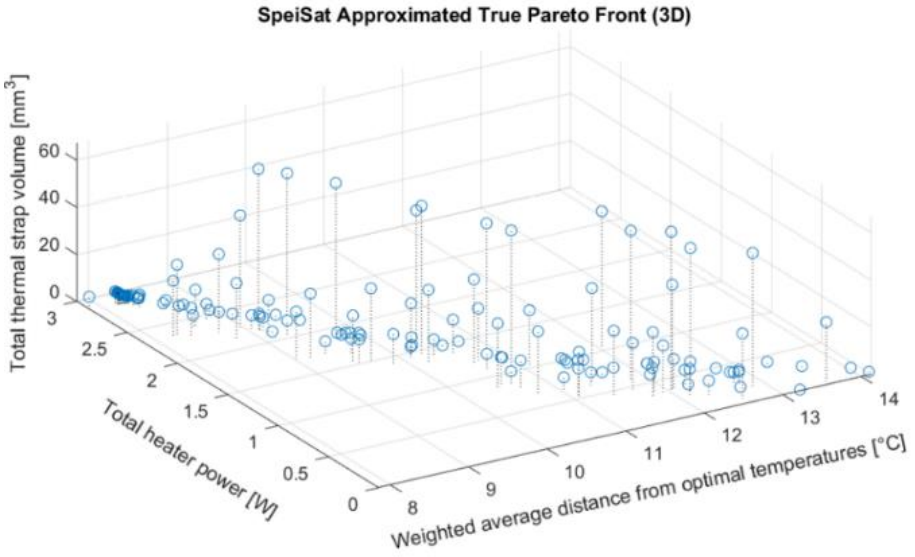


Figure 73: 3-dimensional representation of the approximated true Pareto front for the SpeiSat design optimization. The data points correspond to the non-dominated solutions, across all algorithms.

From the shape of the final Pareto front is clear that the front recalls a planar geometry or a line in the 2d case: this visually shows the effect of competing objectives: lower temperature differences are possible, but they often come with the cost of adding complexity in the system with the installation of thick thermal straps, or at the price of increasing the power dissipated on board by heaters-like elements.

From the observation of the final Pareto front, the final trade-off solution was selected: the rationale was to try and keep the temperature closer to the optimal ones, while also keeping the TCS design simple, which is crucial in the first phases of design. The design point chosen is a solution found by the RVEA algorithm, having cost [10.79, 1.501, 0.04427] for the first, second and third objectives respectively. The optical properties of this solution are listed in Table 27:

Table 27: Optical properties of the final optimized solution chosen for SpeiSat.

Alpha Z-	Alpha Z+	Eps int Y-	Eps int X+	Eps int Y+	Eps int X-	Eps int Z-	Eps int Z+	Eps ext Z-	Eps ext Z+
0.8394	0.8954	0.0506	0.0572	0.2743	0.0540	0.1497	0.1264	0.0519	0.0528

On the external Z faces a material with high α/ϵ is needed, based on the numerical results ($\alpha/\epsilon = 16.17$ for the Z- face and $\alpha/\epsilon = 16.96$ for the Z+ face). The IR emissivity ϵ is desired to be low both on the internal and the external sides of the structure, to retain as much heat as possible inside the satellite, increasing steady-state temperatures. In the real case, a surface finish of SurTec 650 was chosen to satisfy this need, the ratio α/ϵ of this treatment is high, with a BoL value of 6.8, which was determined to be enough to satisfy the requirement of SpeiSat. Regarding the heaters, the solution chosen had the following average powers:

Table 28: Heater power in the final optimized solution chosen for SpeiSat.

Heater avg power [W] (ComSys1)	Heater avg power [W] (ComSys2)	Heater avg power [W] (Battery)
0.0553	0.903	0.542

ComSys1 is almost not heated, while ComSys2 has a high dissipation of almost $1\,W$. In the simulations ComSys2 is the coldest component because it is always oriented towards deep space and does not receive albedo or IR heat from

the Earth, furthermore, the alternation of sunlight and eclipse mean that the temperature oscillations have very high amplitude. The choice to heat this component is thus well motivated: in the next design iterations of SpeiSat, to match this prediction, one of the shunt resistors of the DET was placed on the Z+ face of the satellite, near ComSys2, to compensate for the low temperature. This design choice, however, is not without problems: although the shunt resistors of the DET have a peak power dissipation of around $1\,W$ they are not actively controllable, and their operation depends on the power budget, in fact, they typically operate when the battery state of charge is near 100%. One situation in which they could be useful is at the end of the recharge operative mode of the cold case: after the battery reaches full capacity all the excess power available on board is dumped inside the satellite as heat, which is beneficial for SpeiSat which, on average, suffers more from cold scenarios.

Another reason why this solution from RVEA was chosen is the almost absence of thermal straps, the cross-sectional areas for the two links resulted, in fact, in $0\,mm^2$ and $\sim 1\,mm^2$, which from the thermal point of view are negligible and can be ignored without significant repercussions. The absence of additional straps has also the favourable effect of simplifying the configuration and design of SpeiSat, which is a fundamental aspect when managing complex systems such as the system engineering a spacecraft, and also fits well with the decision of keeping a central hotter core insulated from the rest of the satellite.

4.3.6 SpeiSat Final Design Iteration

The S2T2 software is designed to achieve a level of detail typical of the first two project phases, shown in Figure 74, namely phases A and B. However, since the Spei Satelles project aims to produce a working satellite, and then reach phase E, it was necessary to produce a thermal model that was more accurate and faithful to reality.

To do this, it was necessary to use a tool that would allow a higher level of detail to be achieved. Table 29 lists all the upgrades that were implemented to reach the latest iteration of the SpeiSat thermal model.

Table 29: Difference between S2T2/First iteration and TD/Last iteration models.

Model	S2T2/First iteration	TD/Last iteration
View factors between central body/planet and external-facing surfaces	Computed with the analytical approximation of "Flat plate absorbing and emitting on one side" [35]	Computed with the Monte Carlo Ray Tracing method
S/C self-shadowing	Computed by casting a single ray from every surface element towards the centre of the heat source	Computed with the Monte Carlo Ray Tracing method
Attitude	Nadir-Velocity pointing, constant	Derived from trajectory analysis
Orbit	Keplerian	Derived from trajectory analysis
Structure, solar panels and solar cells model	Box-like structure, with surface-average optical and mass/volume-average thermophysical properties, taking into account primary structure, solar panel PCBs and solar cells	Box-like structure, with deactivated nodes to model holes, rectangular geometries for each solar panel and each solar cell
Case of C&DHs	Simplified representation: only two rectangular geometries for each C&DH case	Detailed representation: two sides of the C&DH case modelled using two box-like geometries with one face open
Secondary structure	Simplified representation: only support elements of the main component are modelled	Detailed representation: many secondary structure features are modelled
ComSys boards	Modelled using only one rectangular geometry	Modelled using two rectangular geometries, one for the "octagonal" board and one for the transceiver board
Deployable antennas	Not modelled	Combination of 3D brick-like geometries and 2D rectangular geometries
Hole for the umbilical access port	Not modelled	Modelled deactivating nodes of one of the solar panels
PCB material	Isotropic material, average properties of the laminate	Laminate material
Edge/centred nodes	Edge nodes for every component	Combination of edge nodes and centred nodes
Number of nodes	Total of 336 nodes	Total of 1273 nodes
Conductors and contactors	Only node-node conductor	Combination of node-node conductor and surface-surface contactor
Heat loads	Constant, time-averaged heat loads	Constant and time-dependent heat loads
Heaters	Not modelled	Battery heater, with proportional law, switch-on and switch-off temperatures

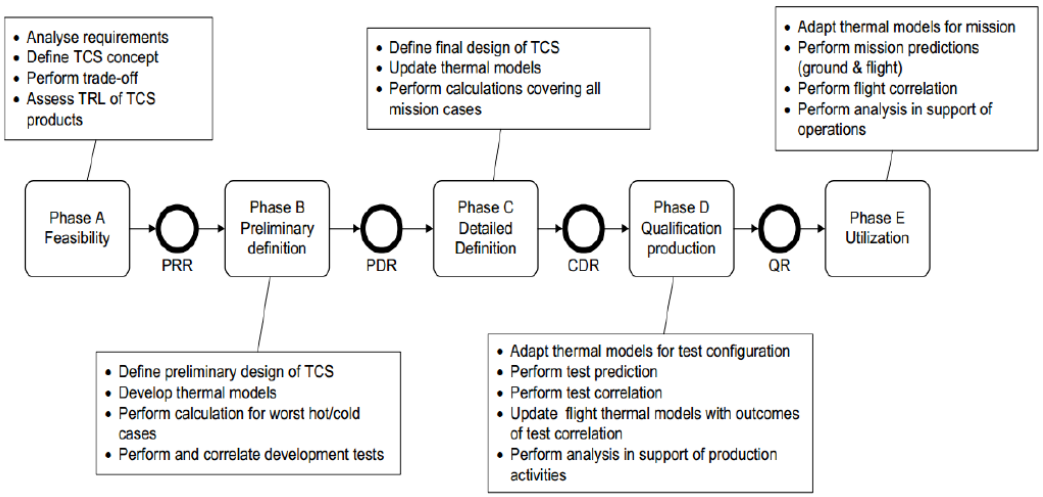


Figure 74: Project phases for TCS Design.

The model shown in Figure 75 was produced. Except for the two ComSys, the other internal components maintained approximately the same geometry as the previous model, except for an increase in the number of nodes, especially regarding the battery model. Two new components, namely the antennas, were introduced, additionally, the model of the external structure was completely revised. For this last iteration primary structure and secondary structure were made as similar as possible to the CAD model, as can be seen in Figure 76. The possibility of deactivating nodes was used to create openings in the structure, thus making the radiative exchange between solar panels and internal components more realistic. This same technique was also used to create the umbilical opening for the access port so that components such as the Backplane could participate in radiative exchange outside of the satellite.

It was also possible to easily create contacts between different surfaces, meaning that conductive exchanges involving all nodes on a surface were created. Another upgrade implemented on the latest model is the ability to create dissipations with a temporal law, to make the heating of some internal components more realistic. TD also offers the opportunity to model heaters within the model, that is, to program certain nodes so that they dissipate a certain amount of power only if their temperature is below a certain threshold. Whereas in S2T2 it is possible to model only Keplerian-type orbits or ideal orbits, TD, on the other hand, allows both orbit and attitude data to be imported from external tools; this involves both the ability to consider disturbances on the orbit but also to consider a more realistic attitude derived from more sophisticated models and analysis.

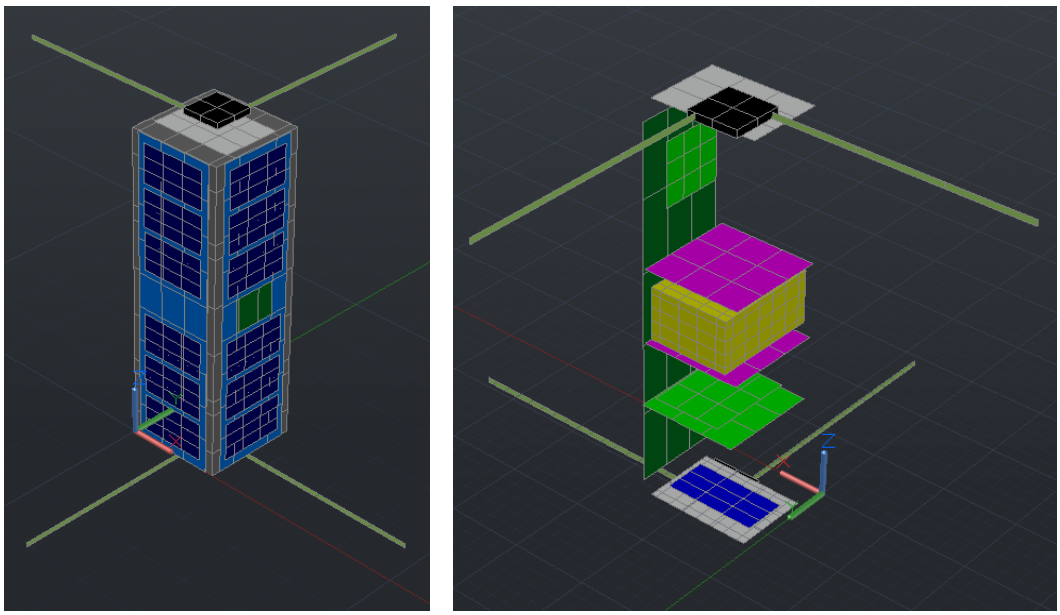


Figure 75: TD model for last iteration, external (left), internal (right)

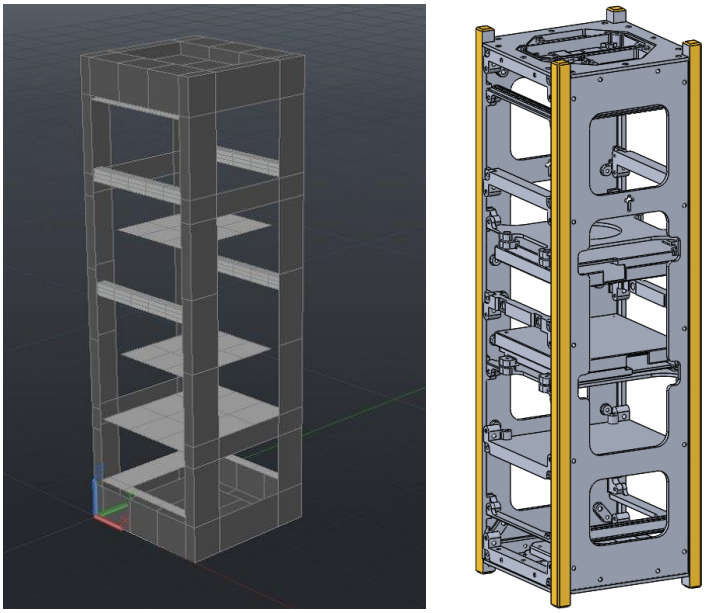


Figure 76: Comparison between the TD model (left) and the primary and the secondary structure CAD model (right).

The results obtained for the hot case and the cold case are shown in the next pages. A margin of 10°C (blue bar) was added to all temperature values obtained from the analyses (green bar) to consider all possible uncertainties in the model [36].

From Figure 77 and Figure 78 it can be seen that all components, both internal and external, fall within their operating ranges considering the margins as well, the only exception being the two ComSys for which considering the margins, the limit of the minor operating temperatures is exceeded. It has not been possible to implement effective thermal control strategies for these components, the reasons are as follows, they are components that mainly face the outside of the satellite, so they dispose a lot of their heat in space. it has not been possible to vary the superficial finishes of these boards, which have a white-coloured solder mask, this greatly limits the absorption of radiation from the outside. In addition, they were only installed on the two smaller external faces namely the +Z and -Z faces, which having a very small surface area still limit the absorption of external radiation. Furthermore, given their remoteness from the other internal components it was not possible to create a thermal connection between them. all these reasons also imply very large temperature fluctuations. it can be observed from the temperature trend graphs, that in contrast to the other components, which have very narrow fluctuations, they have a temperature range in the order of 30°C.

The final temperature trends obtained from TD of the two ComSys, the Sensing Suite, the C&DHs and the battery are visible in Figure 79 and Figure 80, for the cold and hot case, respectively.

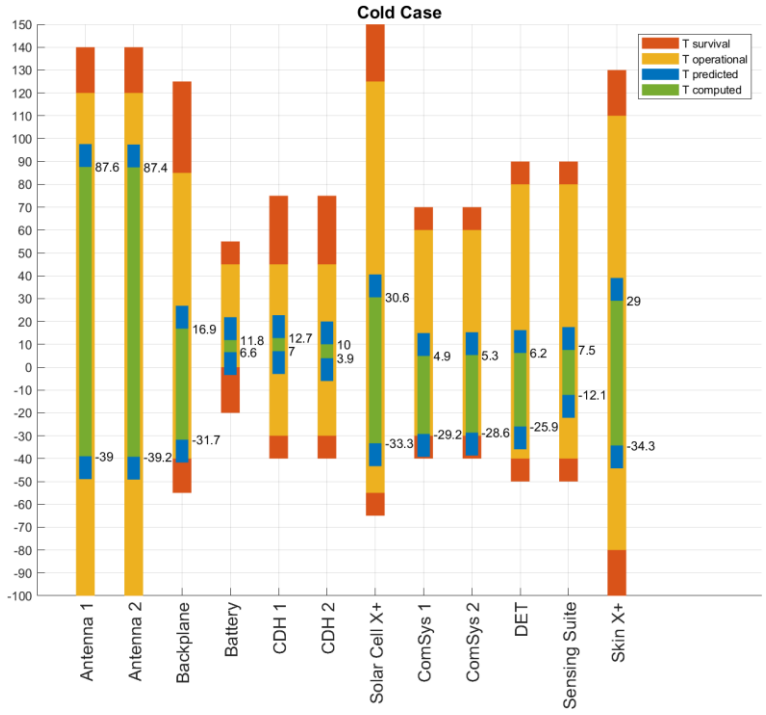


Figure 77: Min Max results, Cold Case.

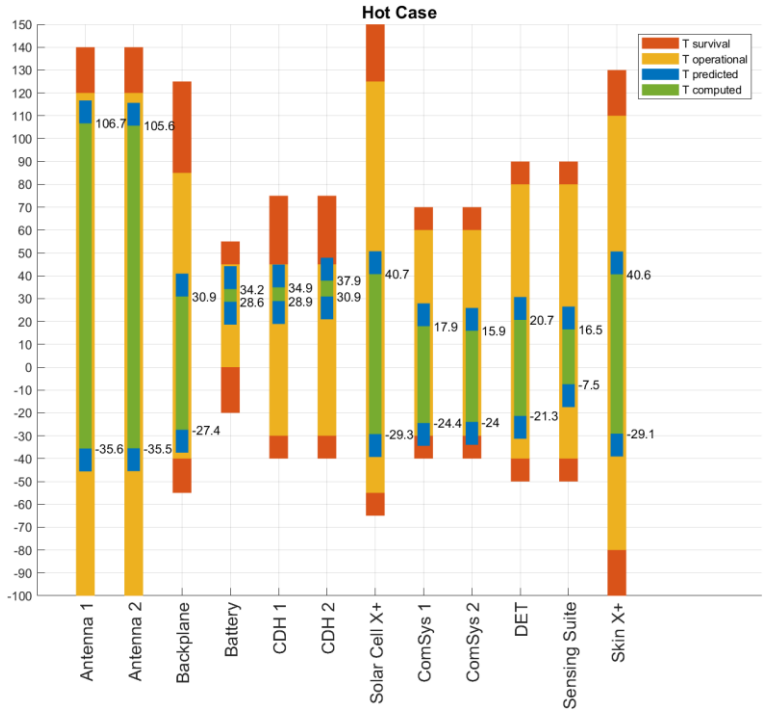


Figure 78: Min Max results, Hot case (right).

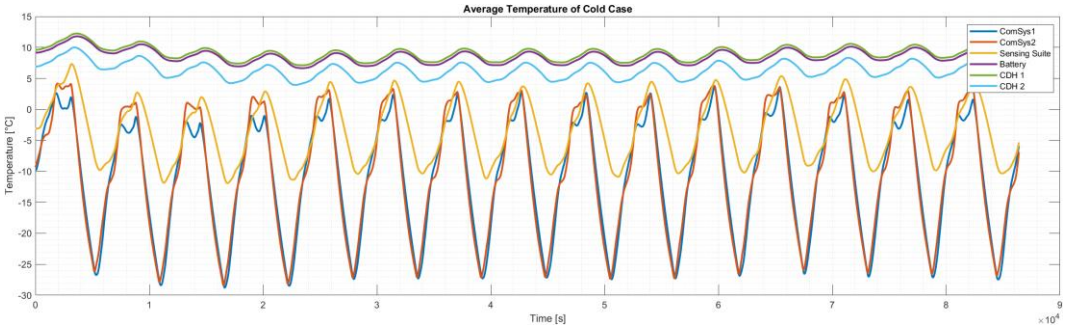


Figure 79: Temperature trends for the Cold Case of SpeiSat.

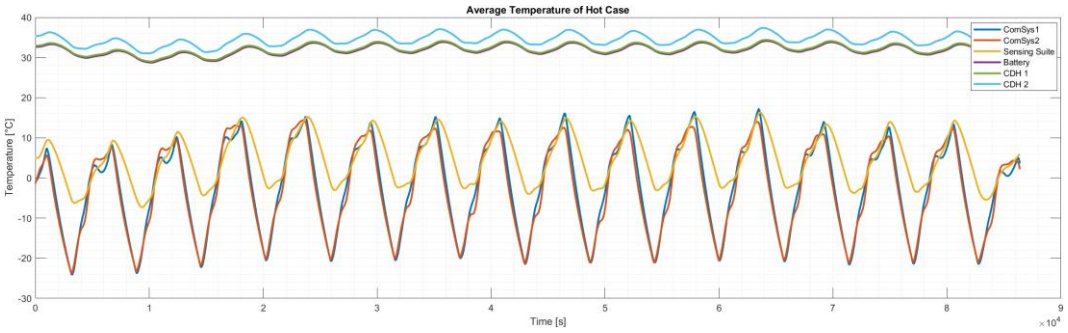


Figure 80: Temperature trends for Hot Case of SpeiSat.

4.3.7 Data from Orbit

During the EOP many telemetries were collected and processed by the Spei Satelles Operations Team. The telemetry data includes the temperatures recorded by the Sensing Suite in the period from the 25th of June 2023 to the 12th of July 2023. Due to the discontinuous nature of the Ground Station operations, downlink requests were sent to SpeiSat on average 1 time a day, only for the passages with a high elevation over the MCC. During low-elevation passages the Ground Stations were kept in listening mode, recording the packets of telemetry data sent by SpeiSat every two minutes. This resulted in the availability of more than 250 data points which, once gathered together, were used to produce the following plots in Figure 81. These graphs are not intended to be temperature trends, because the spacing of the telemetries is not uniform and many data points are clumped near the same time, but rather can be interpreted as scatter plots of the temperatures of the component during different instants of the EOP. Since the date of the cold case used for the numerical simulations is included in the aforementioned period of telemetry availability, these last days of June and the

first ones of July can be assumed as being close to a cold case for SpeiSat. The period, in fact, crosses the moment of minimum environmental heating, which corresponds to the aphelion of the Earth's orbit around the Sun.

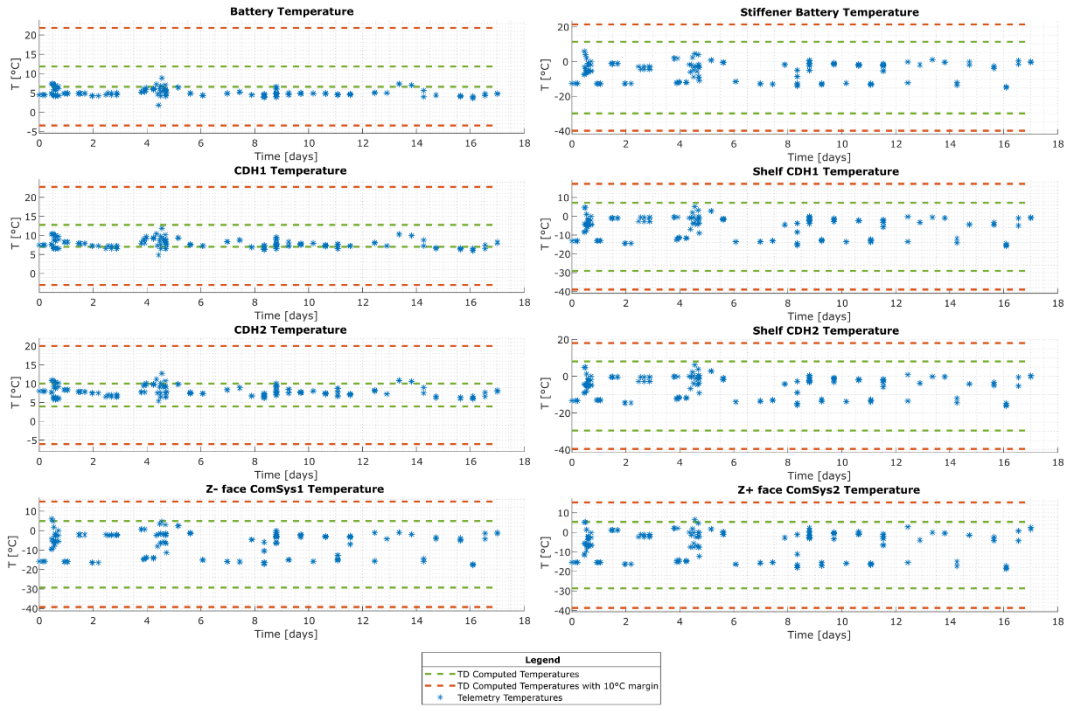


Figure 81: Telemetry data point received from SpeiSat between June 25, 2023, and July 12, 2023.

Although the environment of the first days of July matches the data used for the cold case, the satellite is not in the worst cold operative mode, the Recharge mode. During the EOP the satellite is power-positive, meaning that the total energy consumption on board is on average less than the energy generated by the solar panels: in this condition, the state of charge of the battery never falls below the threshold value, and thus the automating entering in Recharge mode is never triggered. The operative mode of SpeiSat for this period is more similar to the Payload Hot mode because both C&DHs are active and the two buses transmit messages every two minutes, however, the telemetry packets are smaller than the packets needed to transmit mission data of the Payload Hot mode. For these reasons, the operative mode associated with the plots in terms of heat dissipation is intermediate between the Recharge mode and the Payload Hot mode.

In general, the temperatures of every component are inside the dashed lines of the plot, which represent the minimum and maximum temperature computed by

numerical simulations (green dashed lines) with the addition of 10°C margins (red dashed lines), validating the predictions of the earlier design phases. The temperatures of the two ComSys, which were critical in the design phases, present narrower fluctuation compared to the simulations, with most data points located between -15 °C and 5°C, confirming the conservative assumptions made in the definition of the cold and hot case during the development of the numerical models.

Looking at the secondary structure elements, and in particular, those where insulating elements were added to maintain the battery/C&DHs core warm, it is evident a temperature gap across the insulated bolted interfaces. Looking at the three graphs below, in Figure 82, Figure 83 and Figure 84, it can be seen that for most data point this temperature discontinuity resulted in a gap between 5°C and 15°C, which correlates well with the numerical simulations and prove that the design and final integration of the insulating element was performed correctly.

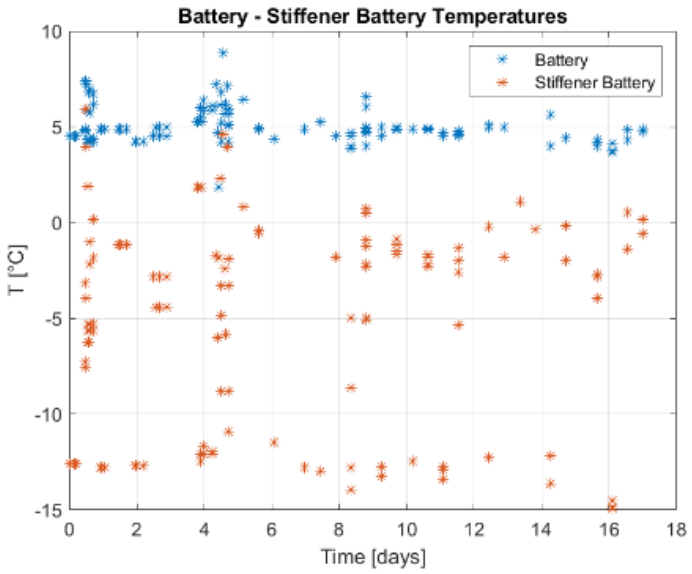


Figure 82: Temperature difference between the battery and its secondary structure support elements.

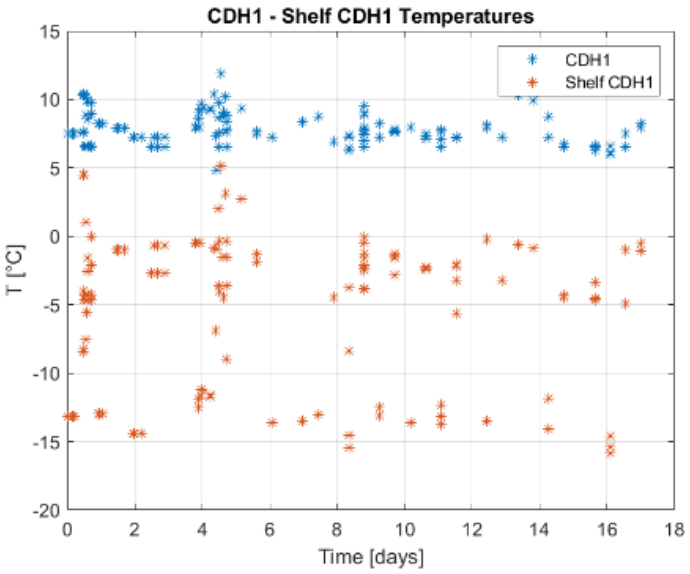


Figure 83: Temperature difference between C&DH1 and its secondary structure support element.

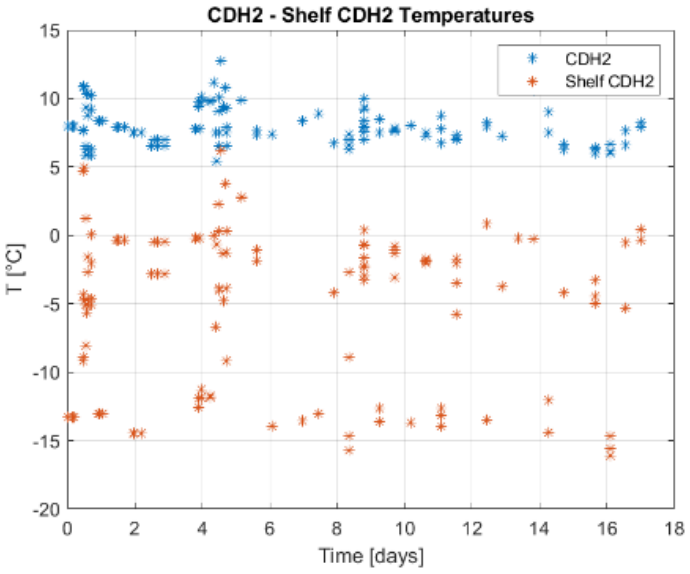


Figure 84: Temperature difference between C&DH2 and its secondary structure support element.

Figure 85, to Figure 88 instead illustrate the data collected by the Sensing Suite on July 13, 2023. The detailed temperature trends are overlayed over the green lines, which represent the items' minimum and maximum temperatures computed in TD for the cold case. Analogously to Figure 82, to Figure 84, the first three plots below show the temperatures of the secondary structure elements.

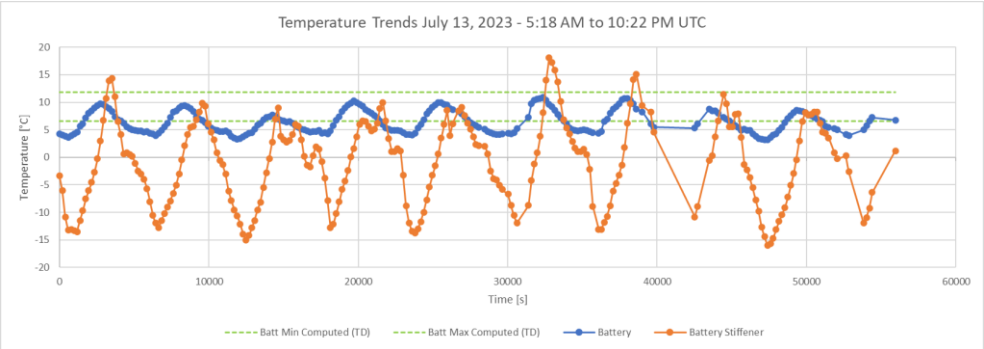


Figure 85: Temperature trends of SpeiSat Battery and Battery Stiffener. Data recorded on July 13, 2023.

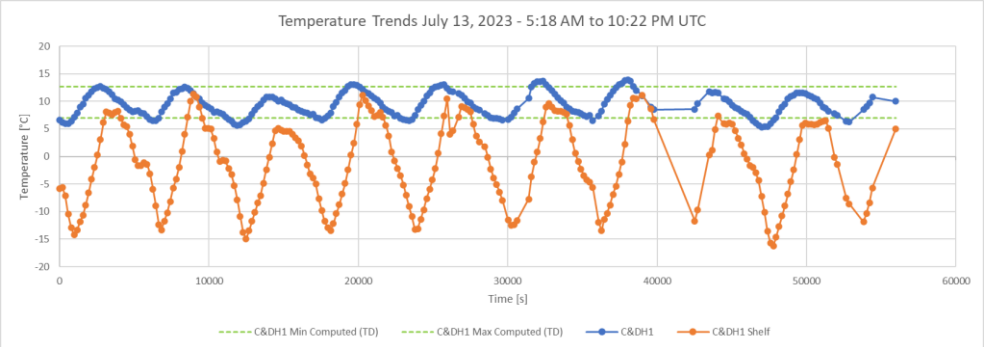


Figure 86: Temperature trends of SpeiSat C&DH1 and C&DH1 Shelf. Data recorded on July 13, 2023.

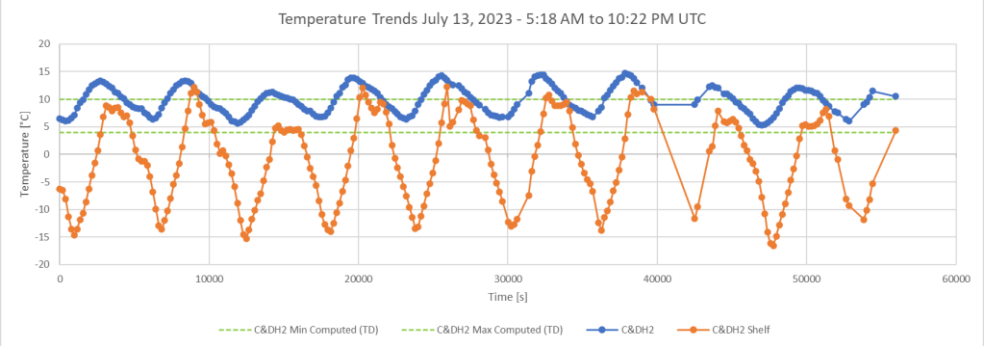


Figure 87: Temperature trends of SpeiSat C&DH2 and C&DH2 Shelf. Data recorded on July 13, 2023.

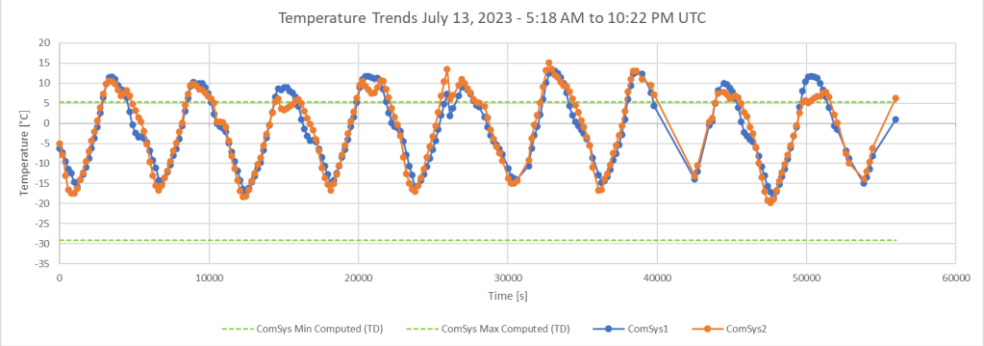


Figure 88: Temperature trends of SpeiSat ComSys1 and ComSys2. Data recorded on July 13, 2023

In the plots, more than 15 hours of temperature data are displayed, with a temporal resolution of about 195 seconds. From the oscillations, the effect of the alternation between eclipse and sunlight is clearly evident, however, the impact is not the same for all the items: the inner, most insulated components predictably present narrower fluctuations, while the external components and the secondary structure show other minor fluctuations that adds up to the main one. The cause of these local, smaller oscillations, which vary from orbit to orbit and that were not present in the simulation, can be traced back to some secondary non-uniform effects such as the variable albedo and IR heating of different regions of the Earth but also on the attitude of the satellite, which cannot be easily predicted by numerical analysis given the passive stabilization system of SpeiSat. Overall the shape of the trends is similar to the ones predicted using numerical simulation, however, the actual minimum and maximum temperature are different, mainly due to attitude discrepancies of the model compared to the real situation and also the different operative mode considered in TD for the cold case.

With more data points like the one plotted on the figures above, and with an approach that considers the temperatures recorded by all temperature sensors of SpeiSat, data like the one presented in these plots can be used to refine the thermal models of the satellite, improving the estimates of the conductance value across the mechanical interfaces (where contactors are simulated), the capacitance of the thermal nodes, and the heat fluxes between the components and from the environment. This technique is usually named *model correlation*.

One future development of the work presented in this thesis and the parallel one [31] is the complete model correlation of the GMM and TMM produced both using S2T2 and TD, improving the understanding of the thermal phenomena both on a qualitative extent and also on a quantitative level, gaining precious data which could be used in every design phase of future space missions.

Chapter 5

Conclusions

The work conducted in this thesis reached the main objective of arriving at a new release of S2T2, improving the existing features and expanding the ones which showed the most important limitations, both from the point of view of computational efficiency and from the usability perspective. S2T2 was also validated in different applications. The optimization algorithms of the Design part of S2T2 were tested and their performances were assessed with state-of-the-art techniques, with satisfactory results. The validation of the new cost function performed through the design optimization of SROC showed how the computational speed was dramatically increased and the availability of multiple optimizers inherently increased the robustness of the solutions generated by the algorithms. Finally, all the new functionalities of the optimization were applied to the real-world case study of the Spei Satelles mission; S2T2 was a fundamental tool in the first phases of development and helped guide the design choices in the right direction. Comparisons with the commercial software Thermal Desktop presented a significant improvement in accuracy over the first version of S2T2, thanks to more precise models and the correction of many bugs. SpeiSat was a precious opportunity to learn in the field about Thermal Control System design and to apply everything that was learned during the Master's Degree Thesis.

S2T2 went through a complete rework, from its core structure to the more specific functionalities. In this chapter some of the improvements not mentioned in the previous ones are briefly reported, then some discussion on the way forward and the possible future developments of the work are presented.

5.1 Code Optimization

Regarding the work done for optimizing the software in general, one of the key aspects was the vectorization of the Monte Carlo Ray Tracing functions, which was one of the most computationally demanding parts of S2T2. Thanks to a cooperative effort with the author of the thesis parallel to this one the time required to generate the view factor between the nodes of the GMM was greatly reduced. This allowed to implement a new version of the Monte Carlo Ray Tracing module, which also considers multiple reflections when casting rays. Details are available in the companion thesis "Develop of a Tool for Thermal Analysis of Small Satellites" [31].

Vectorization of the MATLAB source code was applied in many other places to speed up computation. For example, it was used to update and retrieve properties from the data structures, to compute the average temperatures of every item, to calculate TMM data and also in the graphical processing functions that generate plots and graphs.

Duplicate code sections were eliminated where not necessary and many comments were added in the source code, to facilitate further developments and to tidy up the project folder.

5.2 Utilization Flow Improvements

The previous utilization flow allowed for the user in the first release of S2T2 was effective but simple and did not cope well with input sequences different from the nominal one, with many inconsistencies in case of backtracking or entering already inputted data. To allow the user to make changes on the go to the model, improving the usability of S2T2, the graph of possible utilization paths was significantly expanded and was modified enough to allow more robust backtracking of the user inputs. In Figure 89 the complete graph of all the allowed utilization paths is shown, complete with off-nominal use cases (dotted lines).

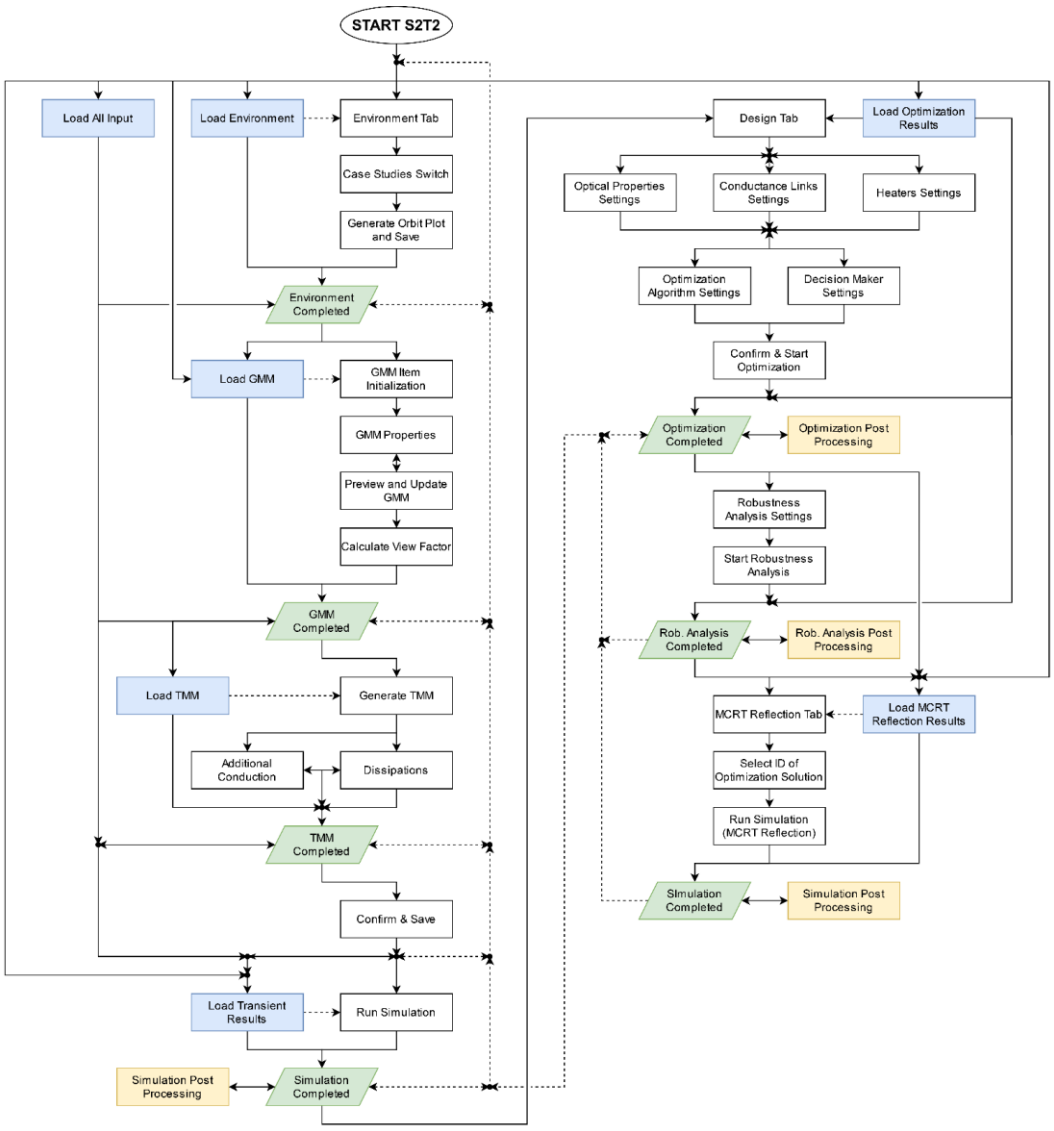


Figure 89: Allowed utilization flow paths of the entire second release of S2T2. Dotted lines represent off-nominal use cases that are correctly managed by the application.

The new structure is organized in milestones, which are represented in Figure 89 by the green parallelograms. Every milestone is associated with a different tab of the new software, namely Environment, GMM, TMM, Simulation, Design, and Monte Carlo Ray Tracing with Reflections. When the user proceeds from a milestone to another the sequence is mostly linear, with some limited backtracking possibilities. When the end of a tab is reached it is then possible to switch tabs and proceed to input new data or review the numbers already entered and make changes following again the flow represented in Figure 89. When a backtracking action is undertaken by the user, the dotted lines must be followed, in the direction specified by the arrows.

All the white rectangle of Figure 89 illustrates the buttons that the user has to press to navigate the tabs and reach the new milestones. The order is important: only the sequence specified by the path of the arrows is allowed. Yellow rectangles instead show that the visualization of graphical results is possible there. Finally, the light blue rectangles illustrate how and where it is possible to import data saved in ".mat" files from previous work sessions. Typically the ".mat" files are created automatically in the root folder of the application upon reaching a milestone.

The new utilization paths allowed during the development and testing of the software to navigate more easily between the input fields and to manage data more freely, saving time that previously was dedicated to running the program again from the start after incorrect user inputs. These improvements will be helpful even for future non-expert users. The elimination of the main code-breaking errors derived from wrong user input was also an important part of this work.

5.3 UI Improvements

The general feeling of the S2T2 UI was maintained from the first version, but different UI improvements were implemented. One of the most important ones is the visualization of the attitude of the spacecraft in the post-processing modules. This allows the user to have visual, clearly interpretable feedback on the data inputted in the Environment tab, reducing the possibility of error.

A grid layout, a feature of the MATLAB App Designer, was set for every panel, button and in general every UI element. This improves the user experience when the window is resized, keeping in reasonable proportion every element, and helps when there are many elements on-screen. On top of that, it allows to make modifications and layout changes more easily for future releases.

The arrangement of panels of the TMM was changed, with the intent of increasing the space dedicated to the spacecraft 3D plot. During development some difficulties in the visualization of large and/or complex models were noted: when many geometries need to be visualized it is important to exploit all the space available for the display of a large plot, increasing the separation between nodes, which are more clearly distinguishable this way. This is especially important when the user needs to check if the additional conduction and dissipations of the

TMM tab are applied to the correct nodes. Figure 90 and Figure 91 show the new UI with two different aspect ratios.

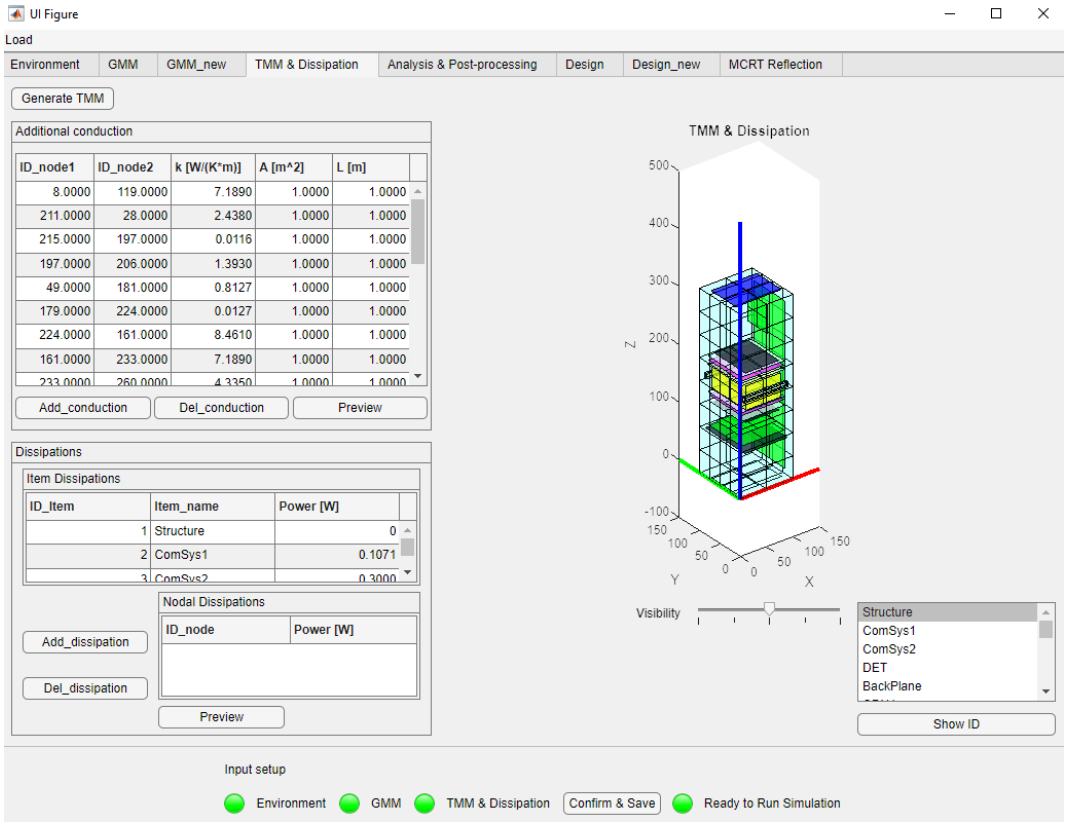


Figure 90: New UI of the TMM tab of S2T2 (default aspect ratio)

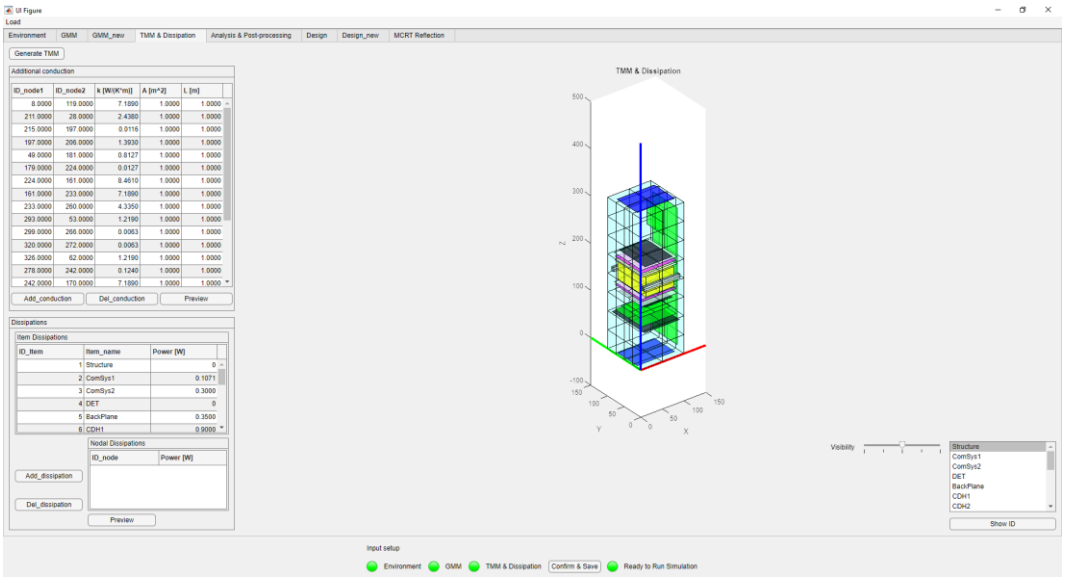


Figure 91: New UI of the TMM tab of S2T2 (full screen 1920x1080 px aspect ratio)

5.4 Future Developments

Even though many aspects of S2T2 were enhanced from release 1 to release 2, a lot of work can still be done to keep improving the software and the UI, to optimize the computations and, most importantly to add new functionalities. In this section are first listed the future works regarding the Design tab of the software, then some more general open points regarding S2T2 are described.

- Multi-objective optimization is a very active field of research. Thus many advanced techniques that could be used to boost the performance of the existing algorithms exist. Some of these methodologies could be introduced in S2T2 and the new optimization algorithms could be tested against the one tried in this work. To give one promising possibility among the many available, an ND-Tree-based update technique could be used for the online update of a Pareto archive [37]. This could apply to most of the algorithms tested in this work since an archive or repository to store the solutions found on the way is a common point of many optimizers.
- The next big step in further increasing the performance of the design optimization is the vectorized evaluation of the cost function. This means that the new solutions are not tested one by one but rather in batches, in a vectorized way, which is the most efficient approach in the MATLAB environment. This would require a rewrite of several sections of the cost

function and also advanced skills in working with high-dimensionality matrices and data structures. Vectorization of the cost function evaluation step of RSA and many of the other optimizers would also be necessary to reach this objective, but the resulting computation times could be lowered by a factor of 10 or more, as shown in Figure 27 of Chapter 3.

- The performance of the algorithms which use tuneable parameters, like RSA, could be boosted with the technique of Hyperparameter optimization, which aims at finding the best parameter for the specific problem faced [10]. The behaviour of the algorithms could thus change and adapt to match the specific criticalities of different problems.
- Other decision-maker algorithms to select the best solution from the Pareto front in addition to the current one based on the weighted sum method could be implemented. ϵ-constraint and goal programming are two possibilities [5].
- New design optimization features could be introduced. More specifically other TCS elements such as cryo-coolers, heat switches, phase-change materials, thermal louvers, heat pipes, variable emittance surfaces, etc. could be modelled and optimized [4].
- Adding new objectives to the cost function, relevant to the new TCS elements mentioned in the previous point, can also be a line of work. Some possibilities are the number of interfaces and thermal gradients, but also, using appropriate models, more high-level metrics such as cost, complexity, etc.
- Since the robustness analysis is the most time-demanding process, filtering out the unwanted solutions (for example the ones with utility value under a certain threshold, or by explicit user definition) before starting the process can help reduce the amount of wasted computations.
- An important aspect that needs further revision is the case of single objective optimization (which presents itself when only the optical properties of the structure are optimized and when only one case study is considered). For single objective optimization, many algorithms with high performances are available both in the MATLAB libraries and in the scientific literature in general.
- The design variable available in S2T2 could be expanded. For example, it could be desirable to optimize the optical properties of other items and not only the structure of the satellite. Another possibility is allowing to specify the starting and ending group for each thermal strap independently.
- If further code optimizations of the cost function are carried out, the possibility of moving from a steady-state design optimization to a full transient optimization could also be evaluated. This would eliminate the need

of performing a post-processing phase after the optimization and would lead to more meaningful optimization objectives, however, the evaluation of the cost of each solution would be very taxing and would probably require some compromises, such as trade-offs on the optimization step.

The future developments regarding in general the entire S2T2 software are many and diverse, below is reported a list of the ones that the author of this thesis deems worthy of interest in the close future of S2T2.

- The spinning attitude, selectable in the Environment tab, needs a heavy rework: in the current state random angular velocities in the range of $0\ rad/s$ to $5\ rad/s$ are assigned to the body axes of the satellite when this option is selected. A more sensible approach would be to give the user the possibility to specify a constant angular velocity or an angular velocity temporal law for each axis of the spacecraft.
- When two case studies are considered simultaneously (hot and cold) there is the need of being able to specify different values for the solar constant, the albedo factor and the Earth IR irradiance because usually, the hot and cold case occurs in two different periods of the year. This could be implemented with three additional "edit field" components in the UI of the Environment tab.
- The possibility of specifying starting temperatures of the transient analysis could be added, instead of always using steady-state temperatures as the initial condition of the simulation.
- New geometries could be added in the GMM tab: some candidates are hollow boxes, cones and truncated cones. The switch to curvilinear element needs to be investigated, to enhance the fidelity of the cylindrical and conical geometries.
- The possibility to "deactivate" some nodes of the model could be introduced, to open the way to implement geometries with holes and apertures more faithfully.
- While the default computation of view factors with Gebhart's method has been greatly optimized, some work could still be done for the computation of view factors using the Monte Carlo Ray Tracing method with reflections. The idea is to apply vectorization techniques similar to what has been done for the calculation that uses Gebhart's method.
- The introduction of time-dependent heat loads could be a great step towards closing the gap between S2T2 and other commercial software. With the same

logic, the introduction of active heaters with user-defined control laws could greatly expand the capabilities of the software.

- The computation of planet-spacecraft view factors could be improved using the Monte Carlo Ray Tracing method, similar to what happens for the node-node radiative exchanges. This would further improve the accuracy of the results, at the cost of a lengthier evaluation of the environmental heat sources. Again, vectorization could be a powerful tool for this implementation.
- The improvement of data management can be explored: having the possibility to export all the satellite data in spreadsheets, *csv* or *json* format could improve the usability of S2T2 in contexts where multiple software are utilized. On the same line of thought, the data of S2T2 could also be ported automatically to other thermal analysis software like Thermal Desktop, using appropriate APIs.
- The UI of S2T2 can be further improved with more buttons and quality-of-life features. For example, more checks for the validity of the inputs could be added to help non-expert users. A help guide could even be added inside the software, to boost the accessibility of S2T2.
- For future source-code developments the utilization of a versioning software such as Git is strongly advised: a repository that is always up to date is the key to a long software life cycle, opening the possibilities of reverting to previous versions, tracking changes and updates and managing multi-developer contributions.

As a final thought, the conclusion of the PhD thesis of the creator of S2T2 is recalled in this quote:

> *The author hopes that S2T2 can help the research community involved in thermal analysis and thermal management system design.*

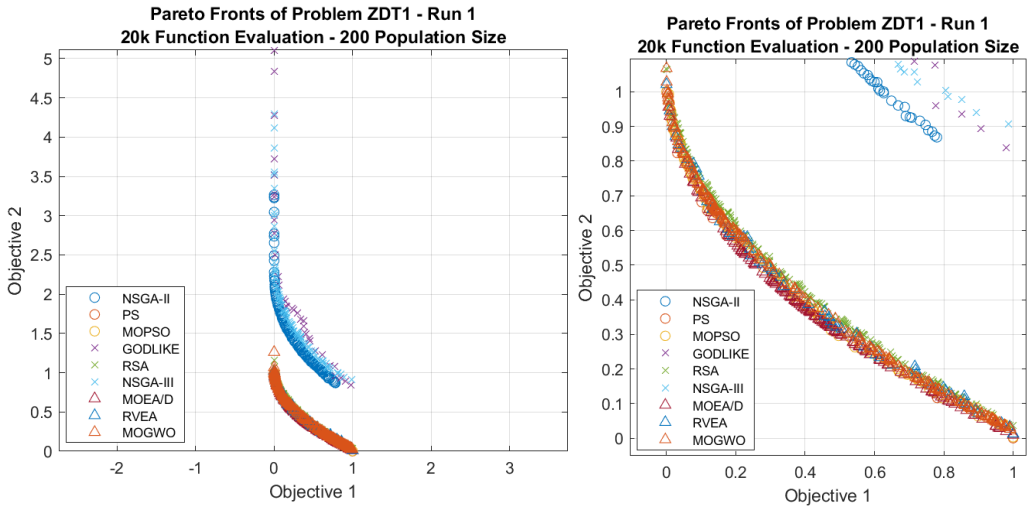
While these words revealed true for what concern the work of this thesis, the author of this document shares the vision of the creator of S2T2 in the sense that this software was much more than an academic exercise on thermal analysis: it allowed to support the design of a real satellite and has been a great learning opportunity in multiple ways. This message to the research community is passed on, in the hope that S2T2 could continue to support research in thermal analysis and thermal control system design.

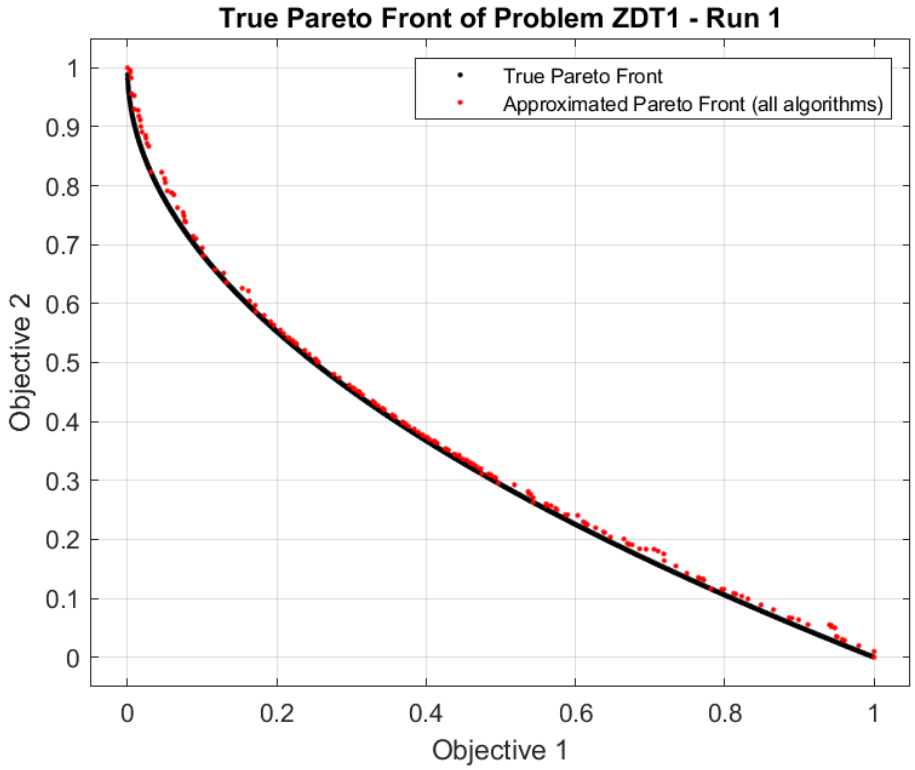
Appendix

6.1 Appendix A - Benchmark Problem Results

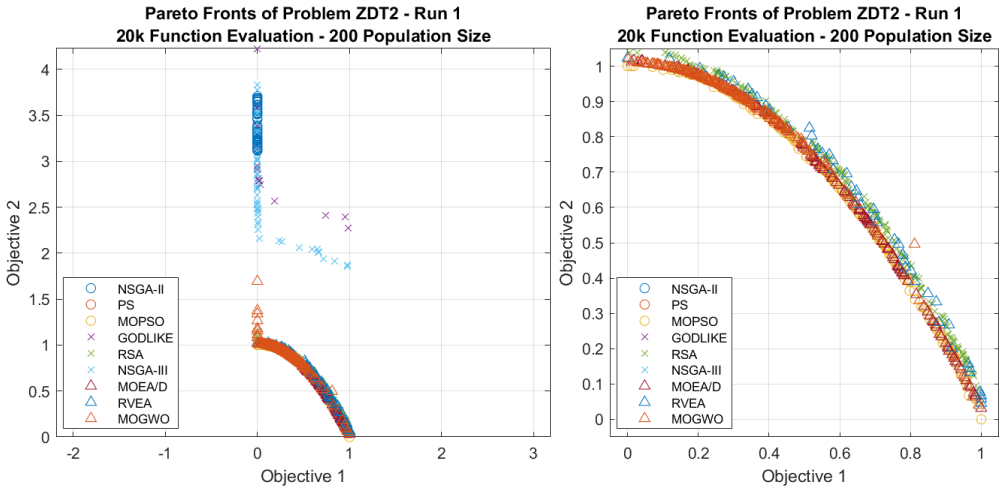
In the following sub-sections of this appendix, the complete results of the benchmark test using the ZDT and DTLZ problems are presented. The first image of every sub-section shows the final optimal Pareto front obtained for every algorithm, with the second one being a zoom of the first one, centred on the area where the true Pareto front is located. The third image shows the real Pareto front (black) overlapped with the approximated Pareto Front (red), obtained by combining all of the non-dominated solutions across all algorithms, to give a visual interpretation of the general difficulty of the problem, across every optimizer. For the DTLZ problems, only the first 2 objectives out of the total 4 are plotted, for ease of interpretation. Note that only the graphical results of the first run out of the 10 performed are shown because the other ones generated similar Pareto fronts and they do not add significant visual information.

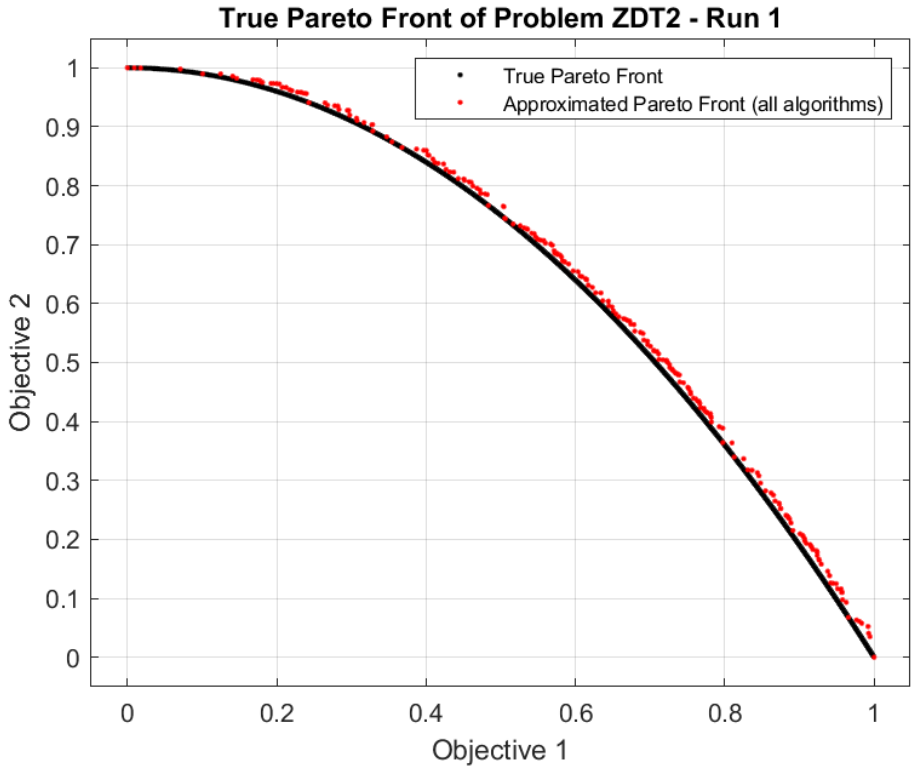
6.1.1 ZDT1


Pareto Fronts of Problem ZDT1 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Pareto Fronts of Problem ZDT1 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO

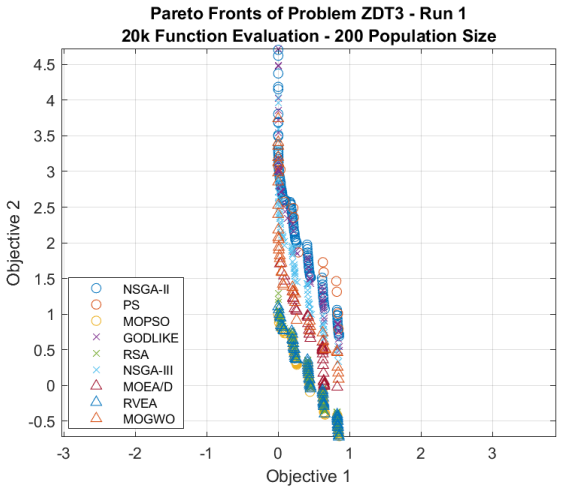

True Pareto Front of Problem ZDT1 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 2
Objective 1

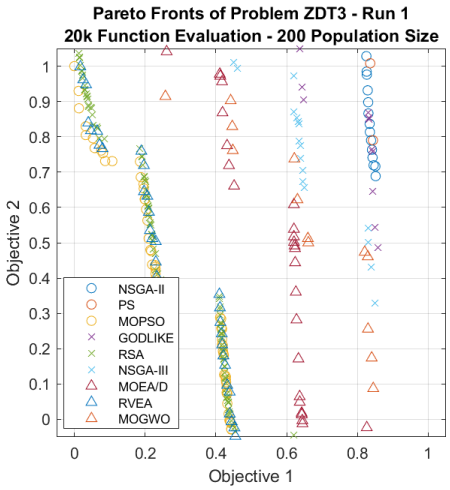
6.1.2 ZDT2

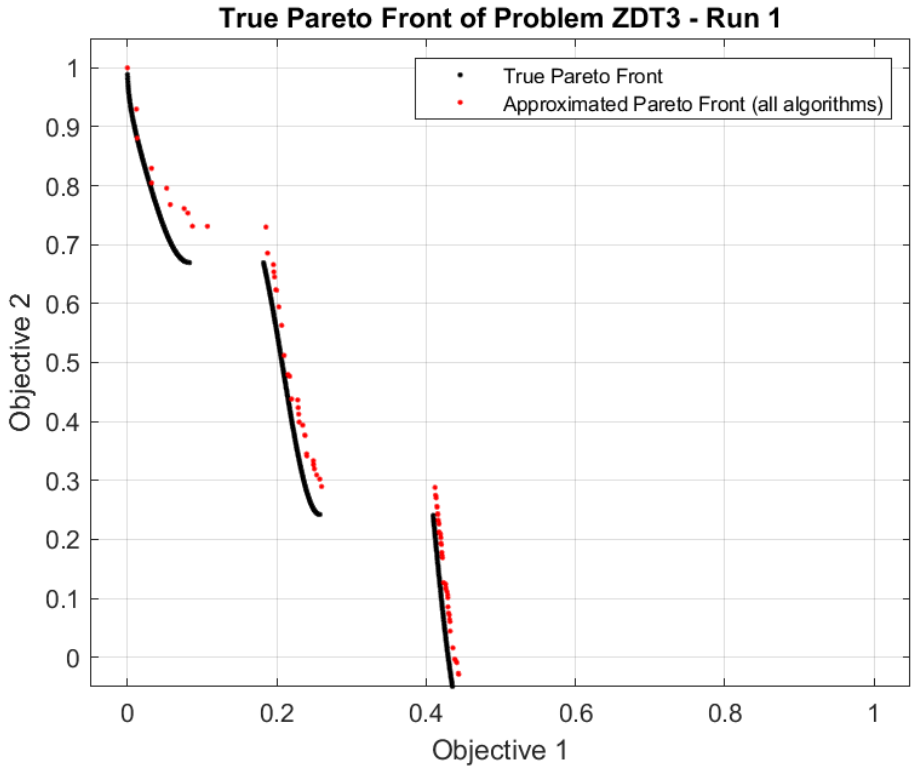

Pareto Fronts of Problem ZDT2 - Run 1
20k Function Evaluation - 200 Population Size
Objective 1
Objective 2
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Pareto Fronts of Problem ZDT2 - Run 1
20k Function Evaluation - 200 Population Size
Objective 1
Objective 2
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


True Pareto Front of Problem ZDT2 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

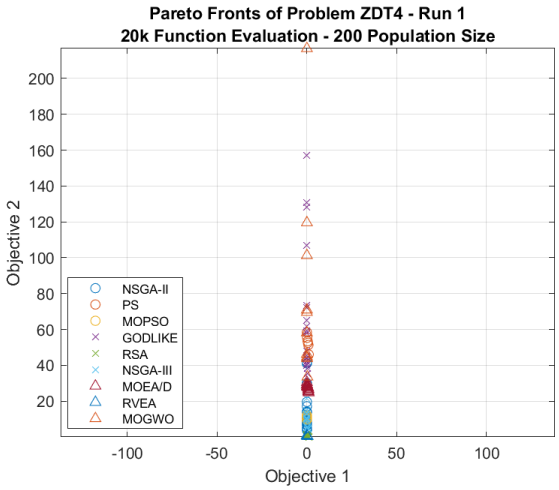
6.1.3 ZDT3

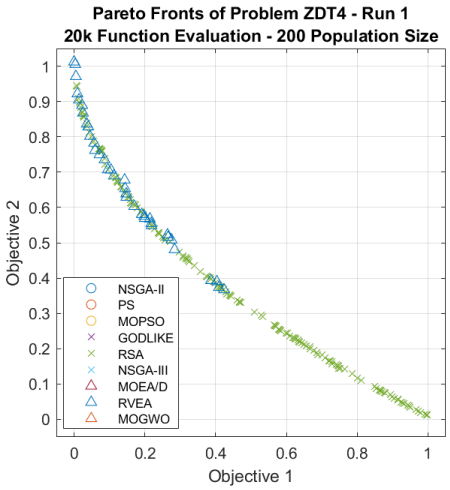

Pareto Fronts of Problem ZDT3 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO

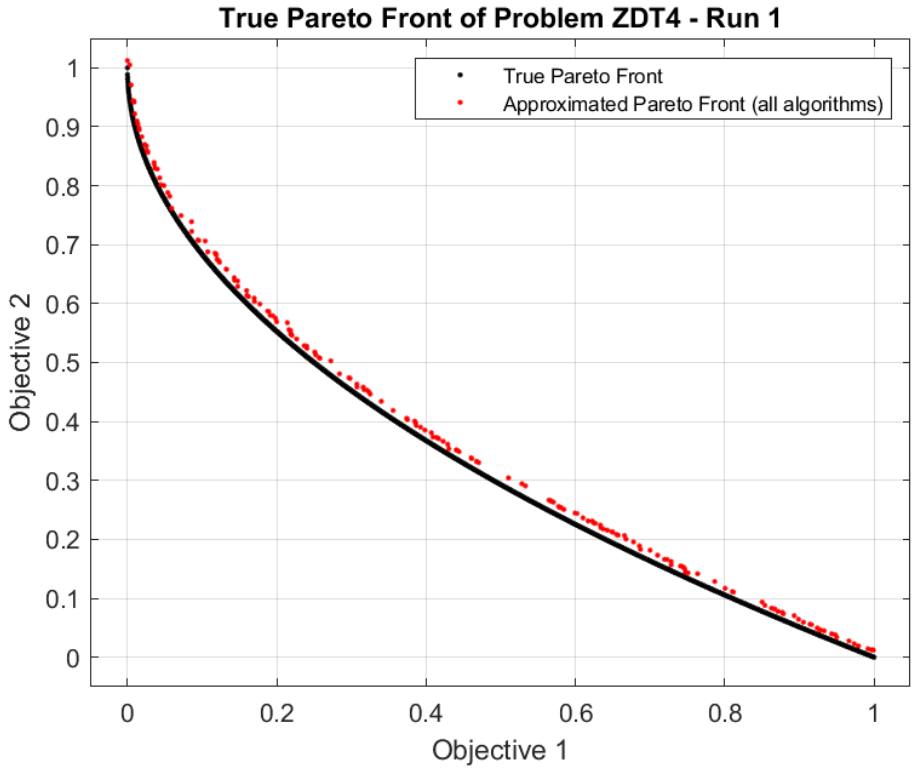

Pareto Fronts of Problem ZDT3 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


True Pareto Front of Problem ZDT3 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 2
Objective 1

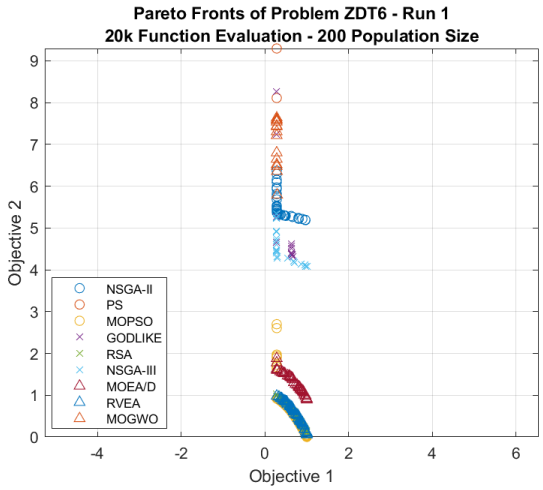
6.1.4 ZDT4

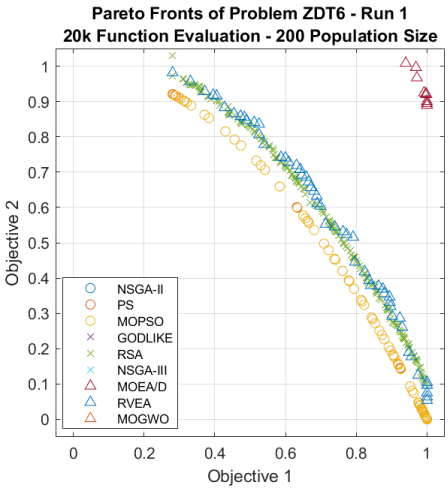

Pareto Fronts of Problem ZDT4 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO

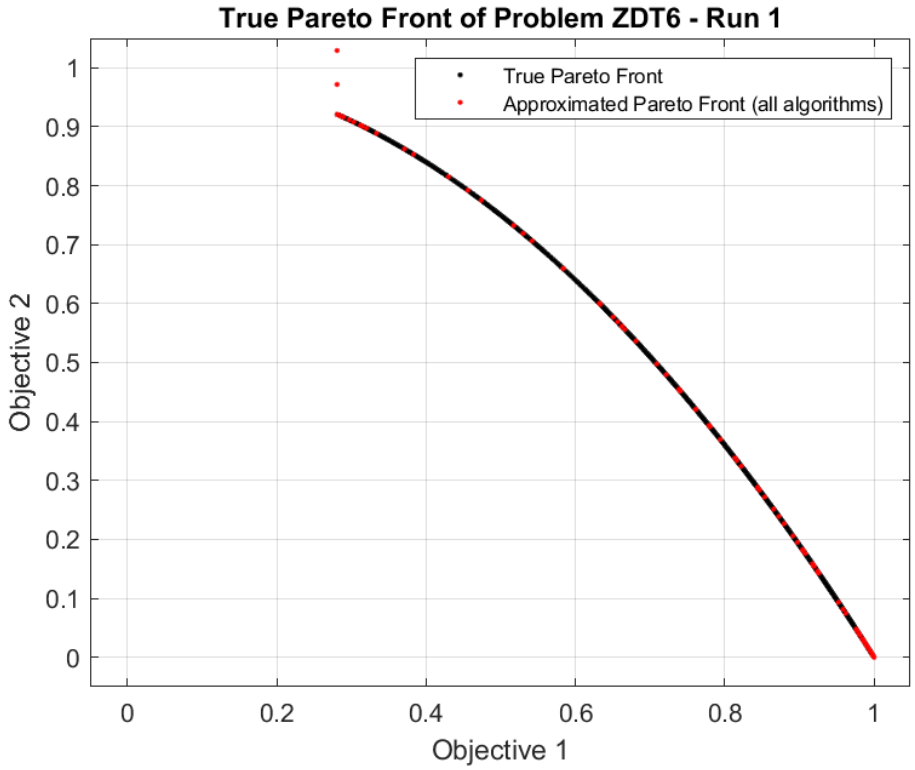

Pareto Fronts of Problem ZDT4 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


True Pareto Front of Problem ZDT4 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 2
Objective 1

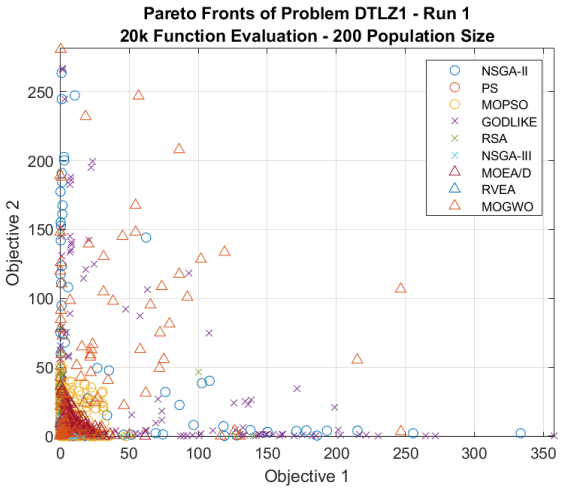
6.1.5 ZDT6

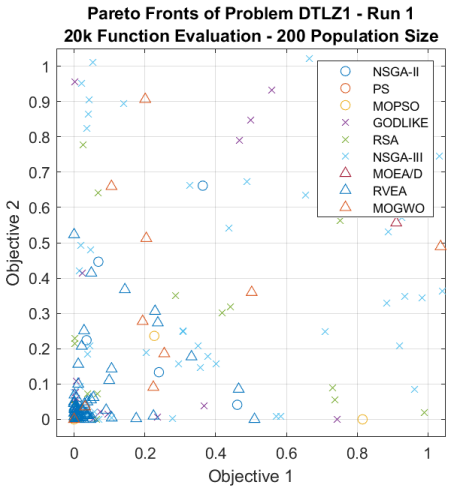

Pareto Fronts of Problem ZDT6 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO

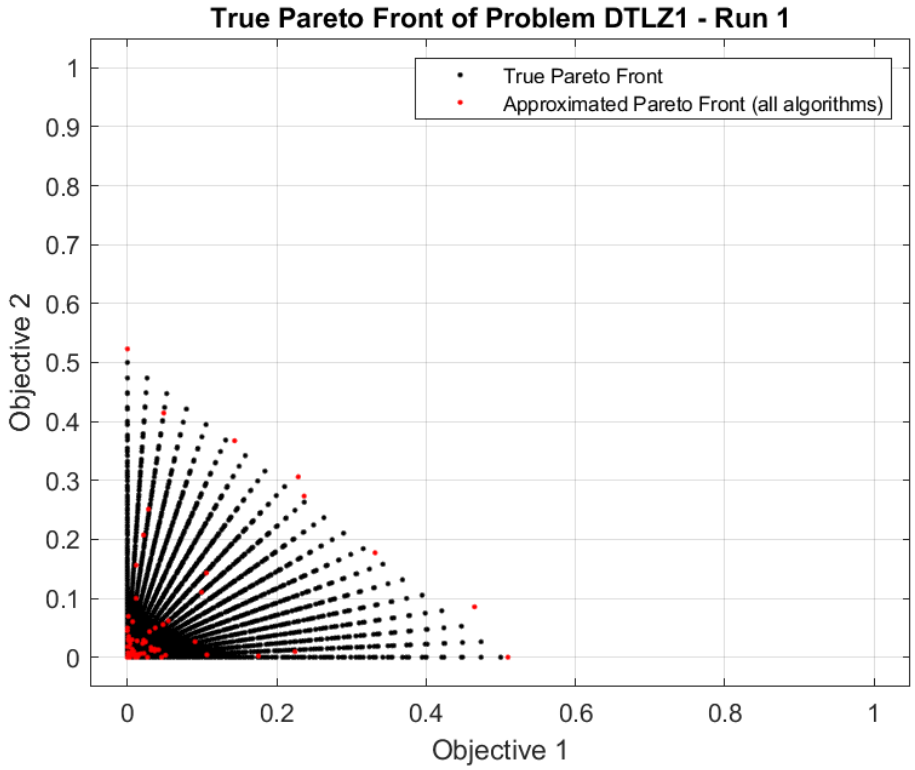

Pareto Fronts of Problem ZDT6 - Run 1
20k Function Evaluation - 200 Population Size
Objective 2
Objective 1
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO


True Pareto Front of Problem ZDT6 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 2
Objective 1

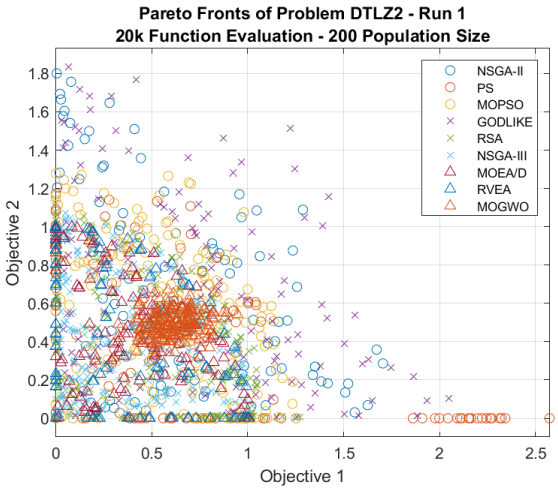
6.1.6 DTLZ1

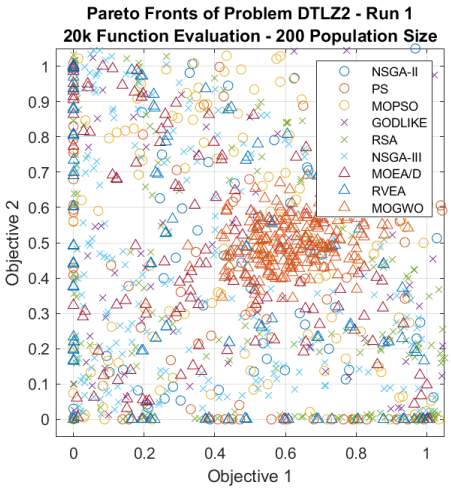

Pareto Fronts of Problem DTLZ1 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2

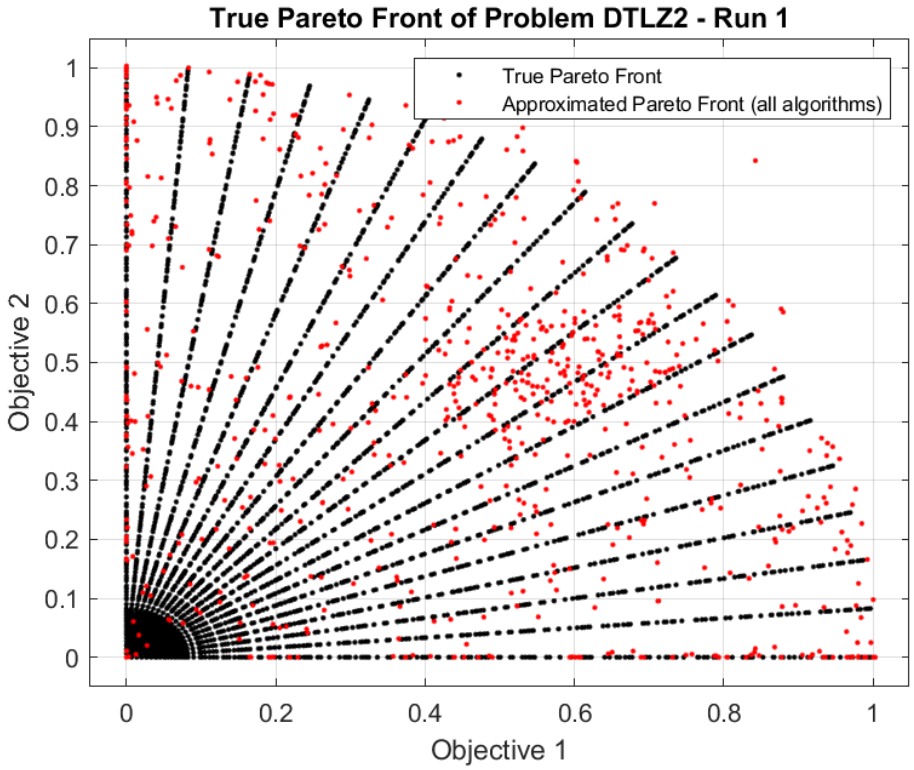

Pareto Fronts of Problem DTLZ1 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ1 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

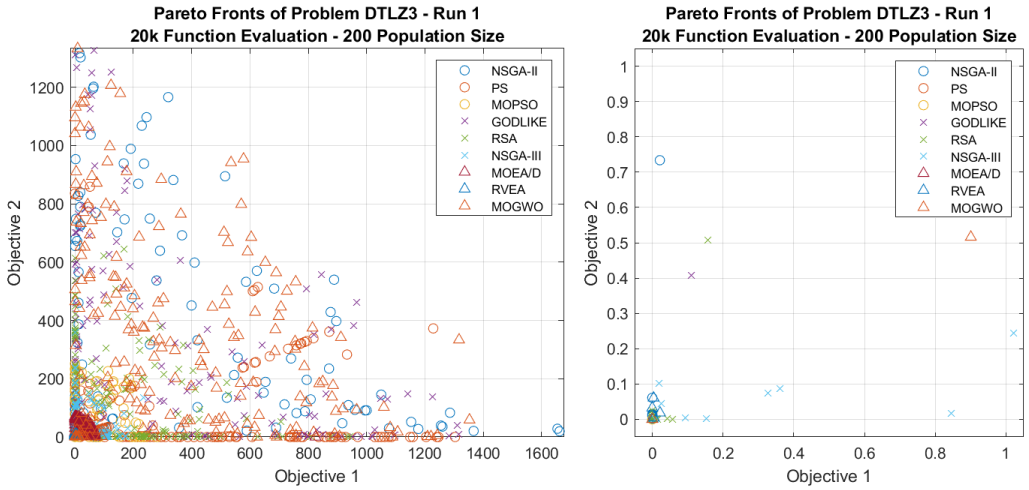
6.1.7 DTLZ2

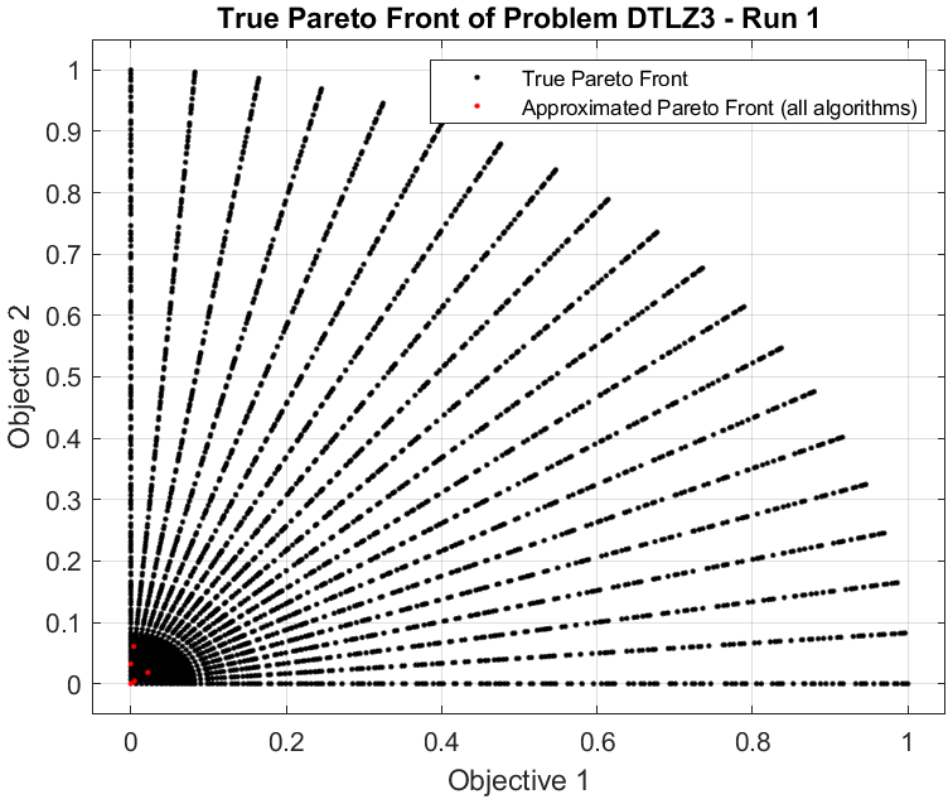

Pareto Fronts of Problem DTLZ2 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


Pareto Fronts of Problem DTLZ2 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2

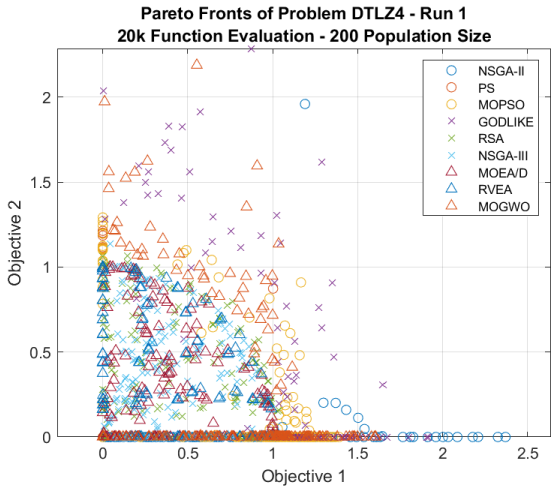

True Pareto Front of Problem DTLZ2 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

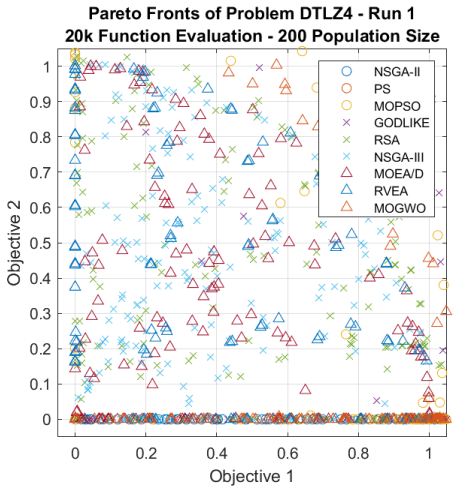
6.1.8 DTLZ3

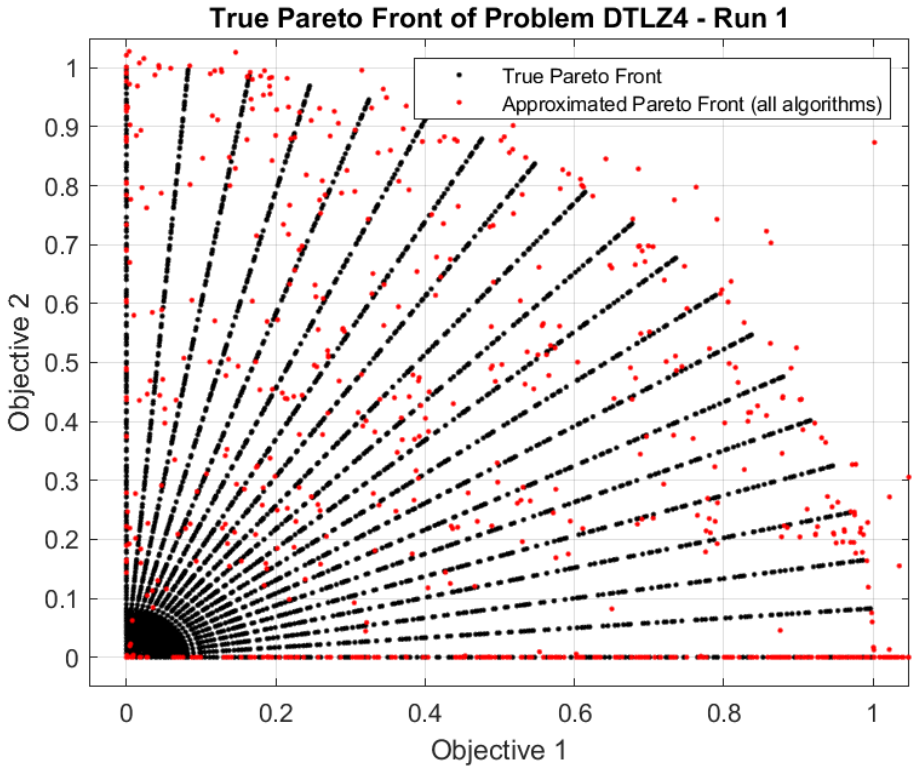

Pareto Fronts of Problem DTLZ3 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2
Pareto Fronts of Problem DTLZ3 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ3 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

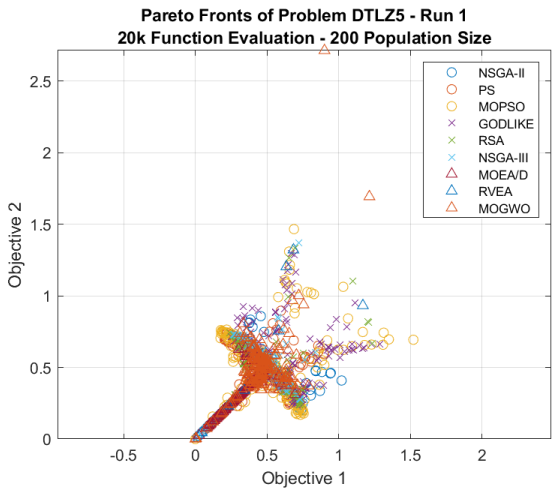
6.1.9 DTLZ4

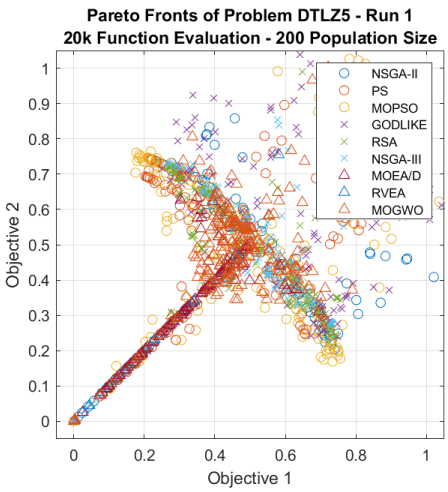

Pareto Fronts of Problem DTLZ4 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2

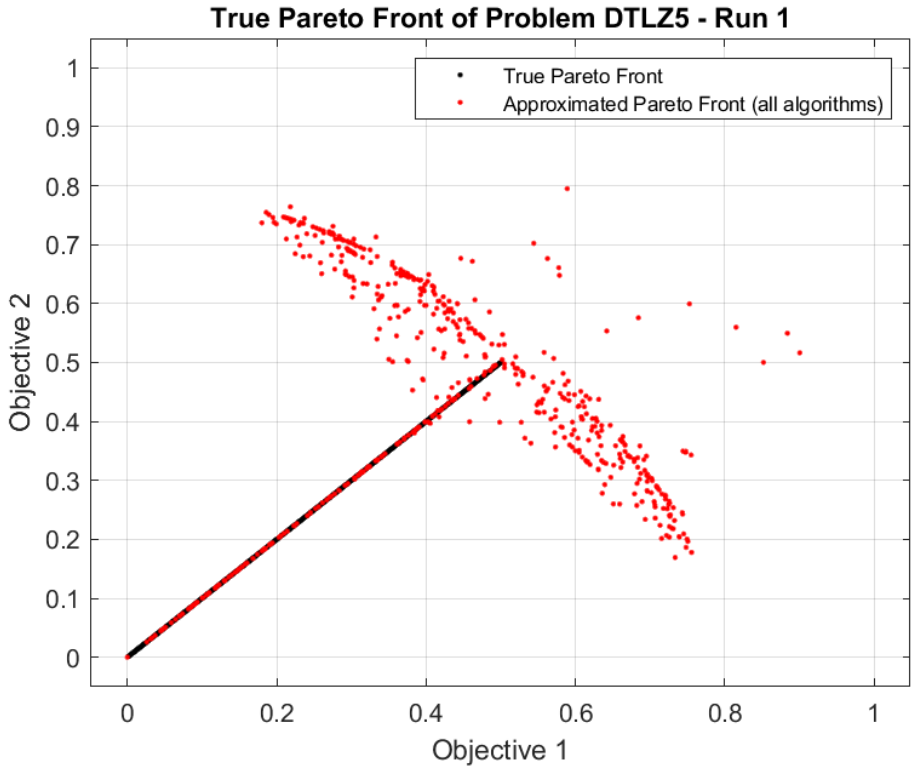

Pareto Fronts of Problem DTLZ4 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ4 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

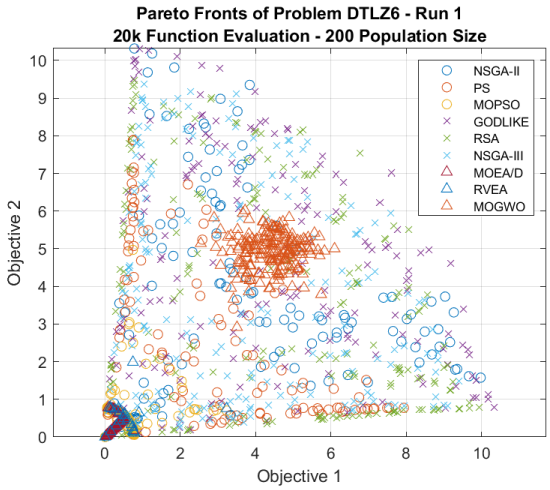
6.1.10DTLZ5

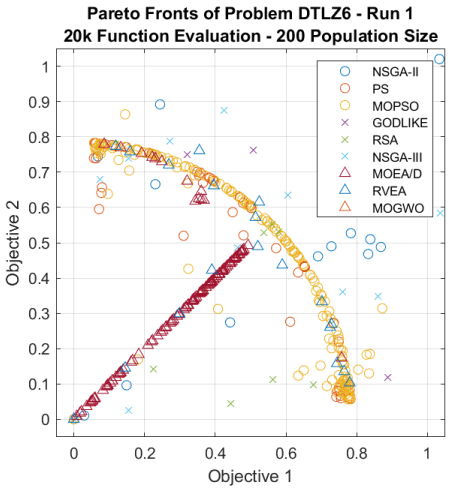

Pareto Fronts of Problem DTLZ5 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


Pareto Fronts of Problem DTLZ5 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ5 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

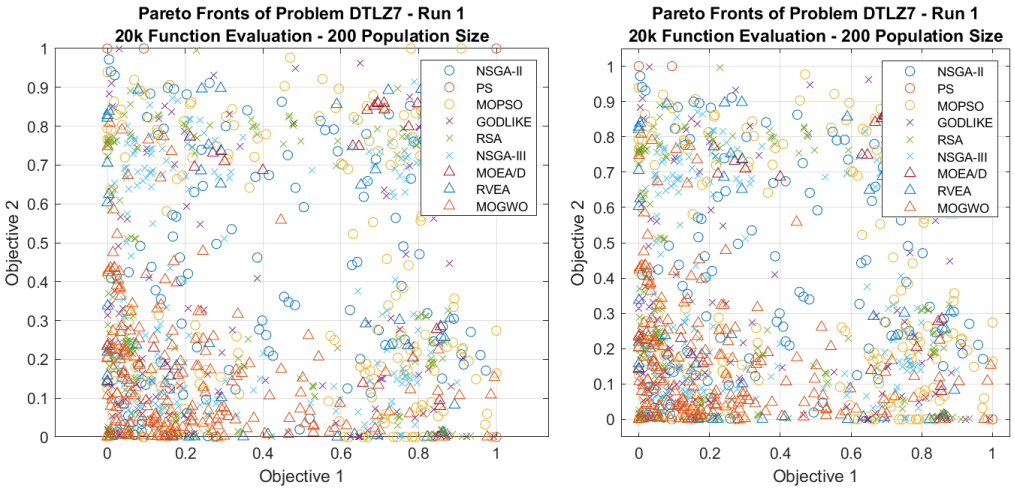
6.1.11DTLZ6

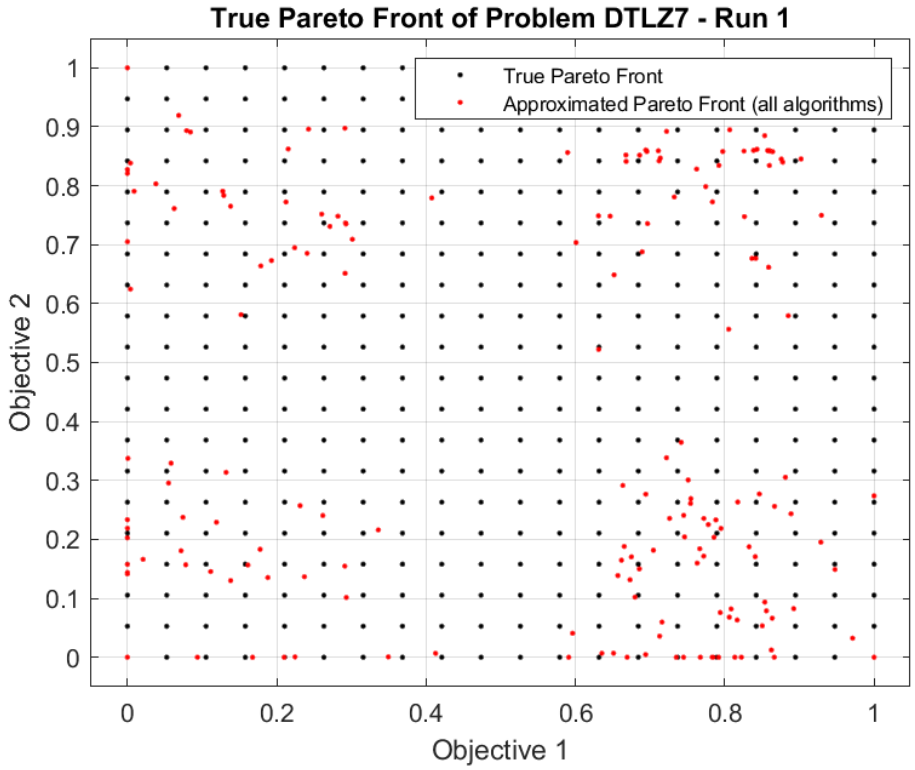

Pareto Fronts of Problem DTLZ6 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


Pareto Fronts of Problem DTLZ6 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ6 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

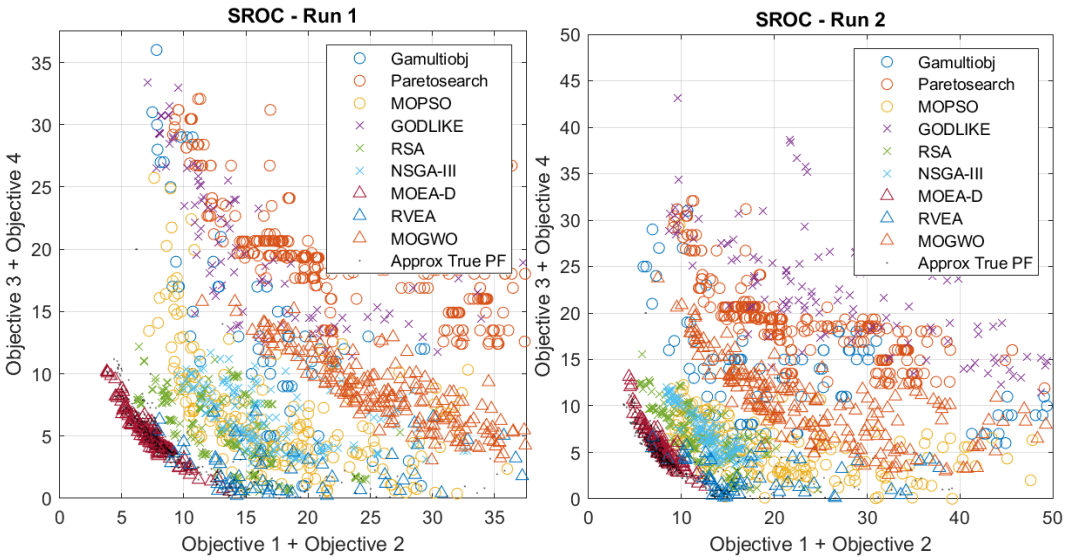
6.1.12 DTLZ7

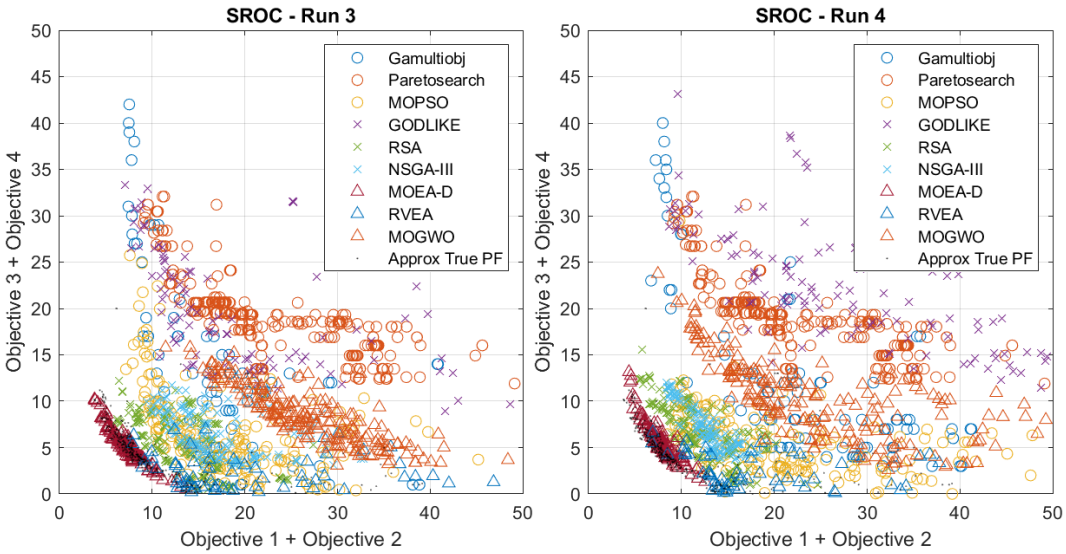

Pareto Fronts of Problem DTLZ7 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2
Pareto Fronts of Problem DTLZ7 - Run 1
20k Function Evaluation - 200 Population Size
NSGA-II
PS
MOPSO
GODLIKE
RSA
NSGA-III
MOEA/D
RVEA
MOGWO
Objective 1
Objective 2


True Pareto Front of Problem DTLZ7 - Run 1
True Pareto Front
Approximated Pareto Front (all algorithms)
Objective 1
Objective 2

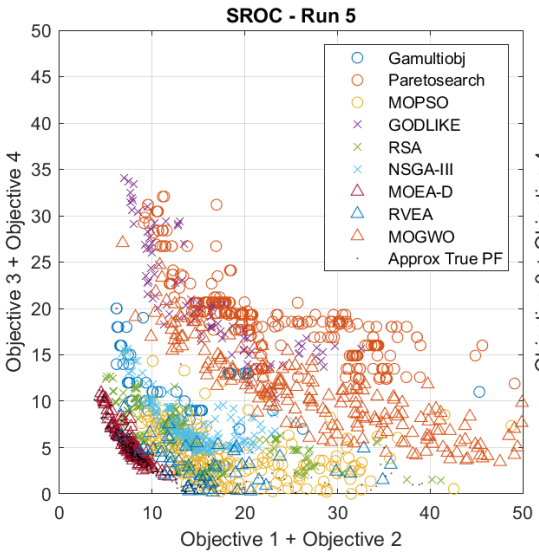
6.2 Appendix B – SROC Design Optimization Results

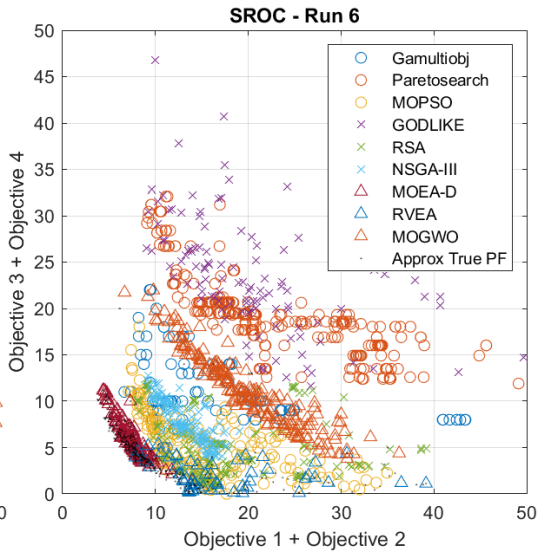
In this appendix, the complete results of the SROC application run are shown. Every image corresponds to a different S2T2 run of the optimization of the thermal design of SROC. In the images, the sum of the first and second costs (temperature objective in the hot case and temperature objective in the cold case) is indicated on the x-axis, while the sum of the third and fourth costs (total heater power in the hot case and total heater power in the cold case) is visible on the y-axis. The results of each algorithm are identified by a different symbol, visible in the legend. The black dots, instead, represents the element of the set of non-dominated solutions, across all algorithms and all executions.

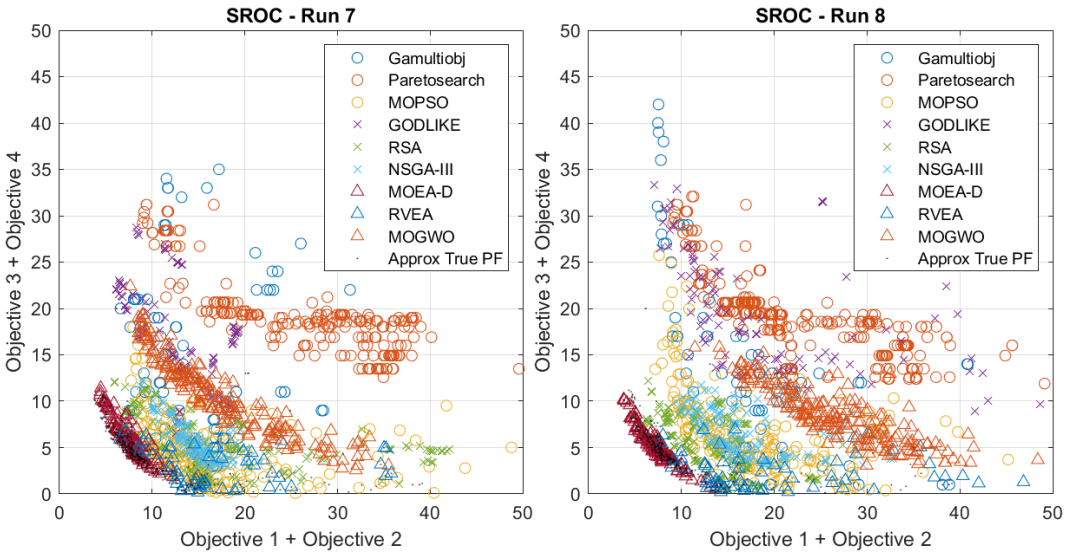


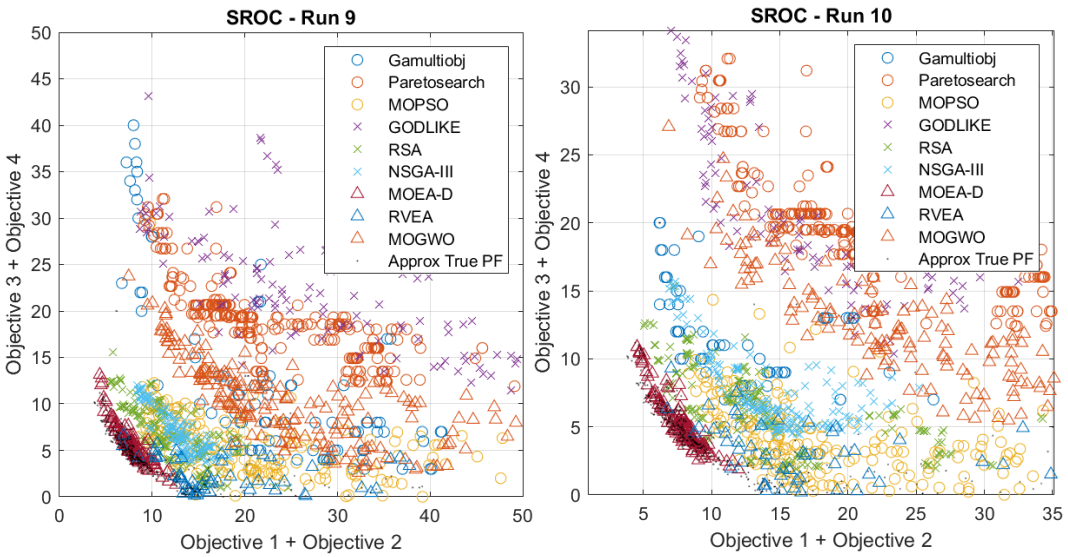

SROC - Run 3
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF

SROC - Run 4
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF


SROC - Run 5
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF


SROC - Run 6
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF


SROC - Run 7
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF


SROC - Run 9
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF

SROC - Run 10
Objective 3 + Objective 4
Objective 1 + Objective 2
Gamultiobj
Paretosearch
MOPSO
GODLIKE
RSA
NSGA-III
MOEA-D
RVEA
MOGWO
Approx True PF

References

[1] E. Mabrouk, «What are SmallSats and CubeSats?,» 7 August 2017. [Online]. Available: https://www.nasa.gov/content/what-are-smallsats-and-cubesats.

[2] BryceTech, «Smallsats by the Numbers,» 2023.

[3] E. Kulu, «Nanosats Database,» 2023. [Online]. Available: https://www.nanosats.eu/.

[4] I. Foster, Small Satellite Thermal Modeling Guide, Albuquerque: Kirtland Air Force Base, 2022.

[5] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.

[6] L. Paulon, «Mathematics in Health Care with Applications,» *Sapienza University Rome, Italy,* 2013.

[7] R. Cheng, Y. Jin, M. Olhofer e B. Sendhoff, «A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization,» *IEEE Transactions on Evolutionary Computationm,* pp. 773-791, 2016.

[8] G. Ridolfi, «Space Systems Conceptual Design, Analysis methods for engineering-team support,» *Politecnico di Torino,* 2013.

[9] K. Amine, «Multiobjective Simulated Annealing: Principles and Algorithm Variants,» *Advances in Operations Research,* 2019.

[10] R. Sengupta e S. Saha, «Reference point based archived many objective simulated annealing,» *Information Sciences, Volume 467,* pp. 725-749, 2018.

[11] S. Bandyopadhyay, S. Saha, U. Maulik e K. Deb, «A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA,» *IEEE Transactions on Evolutionary Computation,* pp. 269-283, 2008.

[12] J. Blank, K. Deb e P. Roy, «Investigating the Normalization Procedure of NSGA-III,» *Evolutionary Multi-Criterion Optimization. EMO 2019. Lecture Notes in Computer Science,* pp. 229-240, 2019.

[13] C. Lin, A. Qing e Q. Feng, «A comparative study of crossover in differential evolution,» *J. Heuristics,* pp. 675-703, 2011.

[14] M. Pant, R. Thangaraj, A. Abraham e C. Grosan, «Differential Evolution with Laplace mutation operator,» *IEEE Congress on Evolutionary Computation,* pp. 2841-2849, 2009.

[15] H. Mohammad, «A DYNAMIC POLYNOMIAL MUTATION FOR EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS,» *International Journal on Artificial Intelligence Tools,* 2012.

[16] X. Ma, «A New Hybrid Evolution Genetic Algorithm with Laplace Crossover and Power Mutation,» *International Conference on Computational Intelligence and Security,* pp. 88-91, 2009.

[17] K. Deb e R. Agrawal, «Simulated Binary Crossover for Continuous Search Space,» *Complex Syst,* 1995.

[18] N. Riquelme-Granada, C. Von Lücken e B. Baran, «Performance metrics in multi-objective optimization,» *Latin American Computing Conference (CLEI),* 2015.

[19] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel e L. Salomon, «Performance indicators in multiobjective optimization,» *European Journal of Operational Research,* pp. pp.397-422, 2020.

[20] G. G. Yen e Z. He, «Performance Metrics Ensemble for Multiobjective Evolutionary Algorithms,» *IEEE Transactions on Evolutionary Computation,* 2012.

[21] L. Bezerra, M. López-Ibáñez e T. Stützle, «An Empirical Assessment of the Properties of Inverted Generational Distance on Multi- and Many-Objective Optimization,» *Lecture Notes in Computer Science,* pp. 31-45, 2017.

[22] B. Yost, S. Weston, G. Benavides, F. Krage, J. Hines, S. Mauro, S. Etchey, K. O'Neill e B. Braun, «State-of-the-Art Small Spacecraft Technology,» *NASA Technical Publication (TP),* 2021.

[23] M. Feathers, Working Effectively with Legacy Code, Prentice Hall, 2004.

[24] D. Calvi, «Thermal analysis and control of small space platforms,» *Politecnico di Torino,* 2021.

[25] S. L. Rickman, «Form Factors , Grey Bodies and Radiation Conductances (Radks),» Thermal and Fluids Analysis Workshop 2012, Pasadena, California, 2012.

[26] A. George e E. Ng, «On the Complexity of Sparse QR and LU Factorization of Finite-Element Matrices,» *SIAM Journal on Scientific and Statistical Computing,* pp. Vol. 9, Issue 5, pp. 849–861, 1988.

[27] D. G. Gilmore, Spacecraft Thermal Control Handbook Volume I : Fundamental Technologies, California, 2002.

[28] «Documentation of mldivide,» 2022. [Online]. Available: https://it.mathworks.com/help/matlab/ref/mldivide.html.

[29] B. Gebhart, «Surface temperature calculations in radiant surroundings of arbitrary complexity-for gray, diffuse radiation,» *Int. J. Heat Mass Transf.,* pp. vol. 3, no. 4, pp. 341–346, 1961.

[30] J. A. Clark e M. E. Korybalsk, «Algebraic methods for the calculation of radiation exchange in an enclosure,» *Wärme- und Stoffübertragung,* pp. vol. 7, no. 1, pp. 31–44, 1974.

[31] F. Lucia, «Develop of a Tool for Thermal Analysis of Small Satellites,» *Politecnico di Torino,* 2023.

[32] E. Zitzler, K. Deb e L. Thiele, «Comparison of Multiobjective Evolutionary Algorithms: Empirical Results,» *Evolutionary computation,* pp. 95-173, 2000.

[33] J. Blank e K. Deb, «pymoo: Multi-Objective Optimization in Python,» *IEEE Access,* pp. 89497-89509, 2020.

[34] K. Deb, L. Thiele, M. Laumanns e E. Zitzler, «Scalable Test Problems for Evolutionary Multiobjective Optimization,» *Evolutionary Multiobjective Optimization. Advanced Information and Knowledge Processing.,* 2005.

[35] ECSS-E-HB-31-01 Part 3A, Noordwijk, The Netherlands: ESA-ESTEC Requirements & Standards Division, 2011.

[36] ECSS-E-HB-31-03 Part 3A, Noordwijk, The Netherlands: ESA-ESTEC Requirements & Standards Division, 2016.

[37] A. Jaszkiewicz e T. Lust, «ND-Tree-based update: a Fast Algorithm for the Dynamic Non-Dominance Problem,» *arXiv,* 2017.