A comparative analysis of algorithms for satellite operations scheduling.

by

Eirini Komninou

Thesis

Submitted to the University of Strathclyde

for the degree of

Doctor of Philosophy

Computer and Information Sciences

April 2020

Declarations

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.
The copyright of this thesis belongs to the author under the terms of the United Kingdom Copyright Acts as qualified by University of Strathclyde Regulation 3.50. Due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Eirini Komninou

Declarations

Published articles containing our contributions:

- *An Integrated System-Operations Approach to the Optimal Design of Small-Scale Satellites, Komninou, E., Vasile, M. & Minisci, E. 11 Oct 2012 UN/JNSS-12-S2*

  An overview of what is included in this thesis' Chapter 3 was presented. A preliminary attempt at performing Multi-Objective optimisation using ACS (optimising operations scheduling, minimising satellite mass by optimising subsystems' design parameters) was also proposed.

- *Optimal dynamic operations scheduling for small-scale satellites. Komninou, E., Vasile, M. & Minisci, E. 1 Oct 2012* (invited for publication to IAA Acta Astronautica)

  A more in-depth description of what comprises this thesis' Chapter 3 was presented, as well as preliminary results on optimising satellite subsystem design parameters.

- *Optimal power harness routing for small-scale satellites. Komninou, E., Vasile, M. & Minisci, E. 3 Oct 2011 IAC-11-C3.1.3*

  An approach to automatically routing satellite cabling i.e. harness was presented, using ACS in a constrained volume representing a small-scale satellite's interior. Experimental results showed that ACS performs well on non-Hamiltonian paths without the use of Local Pheromone update, speeding up computation.

- *Fast evidence-based space system engineering. Vasile, M., Minisci, E., Zuiani, F., Komninou, E. & Wijnands, Q. Oct 2011 IAC-11-D1*

Section “V POWER-TELECOM DESIGN” and subsequent calculations were directly based on this thesis author’s Chapter 3.

- *Approximated computation of belief functions for robust design optimization. Vasile, M., Minisci, E. & Wijnands, Q. 23 Apr 2012*

  Section “C. POWER TELECOM INTEGRATED DESIGN PROBLEM” was taken directly from this thesis author’s early work on satellite subsystems’ modelling (see Chapter 3 and Appendix C)

- *An approach for the robust design of the power systems of small satellites. Vasile, M. & Alicino, S. 29 Sep 2014 IAC-14-C3.4.2*

  Section “II. POWER AND TELECOM MODEL” was taken directly from this thesis author’s early work on satellite subsystems’ modelling (see Chapter 3 and Appendix C). Presented calculations were based on the output of this model.

Abstract

Scheduling is employed in everyday life, ranging from meetings to manufacturing and operations among other activities. One instance of scheduling in a complex real-life setting is space mission operations scheduling, i.e. instructing a satellite to perform fitting tasks during predefined time periods with a varied frequency to achieve its mission goals. Mission operations scheduling is pivotal to the success of any space mission, choreographing every task carefully, accounting for technological and environmental limitations and constraints along with mission goals.

It remains standard practice to this day, to generate operations schedules manually, i.e. to collect requirements from individual stakeholders, collate them into a timeline, compare against feasibility and available satellite resources, and find potential conflicts. Conflict resolution is done by hand, checked by a simulator and uplinked to the satellite weekly. This process is time consuming, bears risks and can be considered sub-optimal. A pertinent question arises: can we automate the process of satellite mission operations scheduling? And if we can, what method should be used to generate the schedules? In an attempt to address this question, a comparison of algorithms was deemed suitable in order to explore their suitability for this particular application.

The problem of mission operations scheduling was initially studied through literature and numerous interviews with experts. A framework was developed to approximate a generic Low Earth Orbit satellite, its environment and its mission requirements. Optimisation algorithms were chosen from different categories such as single-point stochastic without memory (Simulated Annealing, Random Search), multi-point stochastic with memory (Genetic Algorithm, Ant Colony System, Differential Evolution) and were run both with and without Local Search. The aforementioned algorithmic set was initially tuned using a single 89-minute Low Earth Orbit of a scientific mission to Mars. It was then applied to scheduling operations during one high altitude Low Earth Orbit (2.4hrs) of an experimental mission. It was then applied to a realistic test-case inspired by the European Space Agency PROBA-2 mission, comprising a 1 day schedule and subsequently a 7 day schedule – equal to a Short Term Plan as defined by the European Space Agency.

The schedule fitness – corresponding to the Hamming distance between mission requirements and generated schedule – are presented along with the execution time of each run. Algorithmic performance is discussed and put at the disposal of mission operations experts for consideration.

Acknowledgments

Many thanks go to Dr John Levine for his effort, his comments, for sharing and teaching me a new way of thinking. And for showing me how captivating Computer Science can be. Sincere thanks go to my friends and colleagues too, for all their support through health and illness. I was lucky to meet some excellent scientists and people in my Department, who became my good friends.

Amidst major difficulties, I was encouraged to continue my research by Dr Esaú Vicente-Vivas, a pioneering Mexican space engineer, and Prof Ανδρέας Βλησίδης, a forward thinking Greek Informatics Engineer. Both educators passed away before I managed to thank them for their encouragement. Muchísimas gracias Profesor, Σας ευχαριστώ ειλικρινά Δάσκαλε.

Many thanks go to my mentors Dr Damien Anderson, Dr Αρετή Δαμαλά, Dr Mark Dunlop, Dr Jonathan Love, Dr Emma Nicol, Dr Olubukola Oduntan, Dr Eric Ollie, Dr Gareth Owens and Dr Phillip Rodgers (alphabetically) for their invaluable advice and reminding me not to give up. It would be remiss of me not to sincerely thank Prof Imogen Coe – an academic leader – and Prof Carron Shankland, an inspiring Computer Science role model, for sharing invaluable words of wisdom.

Most importantly this thesis would have never been possible without the irreplaceable support of my parents, who encouraged me through the difficult times, who so generously funded me to see through my research degree. Σας ευχαριστώ πολύ, δεν έχω λόγια.

With support and travel funding from the RSE (Royal Society of Edinburgh), I visited the European Space Agency's ESTEC (European Space Technology Centre) (Netherlands) / ESAC (European Space Astronomy Centre) (Spain) / ESOC (European

Space Operations Centre) and EAC (European Astronaut Centre) (Germany), Airbus DS (Defence and Space) (Germany), EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites) (Germany) and four SMEs (Small-Medium Enterprise), namely ClydeSpace, Spire, STAR-Dundee, Bright Ascension. I also spoke to STFC (Science and Technology Facilities Council) RAL (Rutherford Appleton Laboratory) Space.

I would like to extend my thanks to Mr A. Beaton (Airbus Group) for his contributions to the TTC model development. Sincere thanks also go to Mr Joe Zender (ESA PROBA2 / Rosetta), Mr Mauro Casale, Ms Sara de la Fuente, Dr Jonathan McAuliffe (ESA Bepi-Colombo), Ms Laura Tomas, Dr Nora Loiseau, Dr Matthias Ehle (XMM-Newton), Dr Nicola Policella, Dr Simone Fratini, Mr Alessandro Donati (ESA Advanced Mission Concepts and Technologies Office), Dr Mehran Sarkarati, Mr Jonathan Schulster (ESOC), Mr Chris Palfrey (EUMETSAT Meteosat Third Generation), Mr Matthieu Bolland, Ms Hayley Higgings, Dr Chris Stewart (EUMETSAT Eumetsat Polar System), Mr Luis Rodrigues (EUMETSAT), Mr Jeffrey Hendrikse (Airbus DS METOP), Mr Peter Platzer (Spire), Mr Craig Clark (ClydeSpace), Dr Peter Mendham, Dr Mark McCrum (Bright Ascension), Prof Steve Parks (STAR-Dundee), the late Dr Helen Walker (STFC RAL Space UKube-1) for sharing their expertise, information and providing us with valuable and very helpful comments.

Contents

List of Tables

List of Figures

Nomenclature

ACO	Ant Colony Optimisation
ADS	Attitude Determination Subsystem
AI	Artificial Intelligence
AIG	Artificial Intelligence Group
AIMS	APSI Integral Mission Scheduler
AM0	Air Mass Zero
AMCTO	Advanced Mission Concepts and Technologies Office
ANN	Artificial Neural Network
ANOVA	ANalysis of VAriance
APSI	Advanced Planning and Scheduling Initiative
ASCL	Advanced Space Concepts Laboratory
ASPEN	Automated Scheduling and Planning ENvironment
ASPEN-RSSC	ASPEN-Rosetta Science ground segment Scheduling Component
ATS-6	(pplications Technology Satellite-6
BETA	Bioinformatics and Empirical & Theoretical Algorithmics Laboratory
CDH	Command and Data Handling
CMA-ES	Covariance Mix Adaptation-Evolution Strategy
COSEAL	COnfiguration and SElection of ALgorithms
EC	Evolutionary Computation

EGA	Elitism Genetic Algorithm
EIRP	Equivalent Isotropically Radiated Power
EMPRESS	Expert Mission Planning and REplanning Scheduling System
EO	Earth Observation
EPS	Electrical Power Subsystem
ES	Evolutionary Strategy
ESA	European Space Agency
ExoMars	Exobiology on Mars
FPS	Fitness Proportional Selection
GGA	Gender-Based Genetic Algorithm
GRASP	Greedy Randomised Adaptive Search Procedure
GSA	Gravitational Search Algorithm
HGSANM	Hybrid Gravitational Search Algorithm-Nelder Mead
HST	Hubble Space Telescope
IA	Immune Algorithm
ICA	Imperialist Competitive Algorithm
ICRS	Improved Controlled Repeated Random Search
IDRE	Institute for Digital Research and Education
ILS	Iterative Local Search
INTEGRAL	INTErnational Gamma-Ray Astrophysics Laboratory
ISA	Interior Search Algorithm
ISS	International Space Station
ISTC-CNR	Institute of Cognitive Sciences and Technologies – Consiglio Nazionale delle Ricerche
ITAR	International Traffic in Arms Regulations

ITI	Innovation Triangle Initiative
JPL	Jet Propulsion Laboratory
KSC	Kennedy Space Center
LCA	League Championship Algorithm
LEO	Low Earth Orbit
LTP	Long Term Plan
LTP	Long Term Plan
MA	Memetic Algorithm
MCS	Multilevel Coordinate Search
MEX	Mars EXpress
MIP	Mixed Integer Programming
MJD	Modified Julian Date
ML4AAD	Machine Learning for Automated Algorithm Design
MRSH	Multiple-Restart Stochastic Hillclimbing
MTP	Medium Term Plan
NAIF	Navigation and Ancillary Information Facility
NASA	National Aeronautics and Space Administration
NORAD	North American Aerospace Defense Command
O&C	Operations and Checkout
OAT	One-step-At-a-Time
ONERA	Office National d'Etudes et Recherches Aérospatiales
PARR	Planning And Resource Reasoning
PBIL	Population-Based Incremental Learning
PI	Principle Investigator
PROBA-2	Project for On-Board Autonomy-2

PSO	Particle Swarm Optimisation
QAP	Quadratic Assignment Problem
SAA	South Atlantic Anomaly
SAMPLE	Scheduling Algorithm for Mission Planning and Logistics Evaluation
SAT	boolean SATisfiability problem
SFL	Shuffled Frog Leaping
sGA	sequential Genetic Algorithm
SM/PART	Servicing Mission Planning and Replanning Tool
SOC	Science Operations Centre
SRES	Stochastic Ranking Evolutionary Strategy
sSA	sequential Simulated Annealing
STP	Short Term Plan
STScI	Space Telescope Science Institute
SUS	Stochastic Universal Sampling
TCT	Time-Cost Trade-off
TGO	Trace Gas Orbiter
TLE	Two-Line Element set
TS	Tabu Search
TSP	Travelling Salesman Problem
TSP	Travelling Salesman Problem
TTC	Tracking, Telemetry and Commanding
UAV	Unmanned Aerial Vehicle
UBC	University of British Columbia
uES	unconstrained Evolution Strategy
USAF	US Air Force

VEX	VEnus Express
VNS	Variable Neighbourhood Search
XMAS	Xmm Mission Apsi Scheduler
XMM	X-ray Multi Mirror

"For the beginning is thought to be more than half of the whole, and many of the questions we ask are cleared up by it."

– Aristotle, Ethica Nicomachea I.7

1 Introduction

1.1 Space mission operations

Since the dawn of the space era in October 1957, a plethora of space missions (manned or otherwise) have been launched into space. According to NASA (National Aeronautics and Space Administration) more than 8000 missions have been launched [Bell and Grayzeck, 2013] at the time of writing this thesis for Earth observation, meteorology, telecommunications, technology demonstration, commercial applications, manned spaceflight, planetary observation, moon observation and military surveillance. Unmanned satellites are operated from the ground, instructing the spacecraft how to perform every single task at any given moment. Manned spacecraft receive considerable and vital support from the ground too, for the mission flow to be uninterrupted and efficient. Overall, mission operations are pivotal for any mission's success.

Spacecraft design and mission objectives can render mission operations difficult. Limited resources combined with the inability to intervene in case of a malfunction, and intricate orbital manoeuvres form a challenge. While the first unmanned satellite, Sputnik-1, was a simple radio transmitter (undoubtedly a true technological feat of that time), subsequent satellites have become sophisticated observatories, telecommunications relays and scientific laboratories able to analyse or collect and return samples of interstellar material back to Earth. Space mission technological complexity has increased rapidly since 1957, presenting a need for reliability, automation and optimisation of mission operations.

A vital part of mission operations is planning and scheduling. The noun *schedule* according to Collins Dictionary[1] is defined as: *"A plan for carrying out a process or procedure, giving lists of intended events and times"*. Similarly, the scientific definition of scheduling refers to assigning resources to tasks in order to complete all tasks given constraints and requirements [Błażewicz et al., 2007]. Resources can refer to time, cost, personnel, room availability, consumable resource availability such as energy and so on. Tasks map out steps that need to be taken for a process to be completed. They call for different resources, depending on the nature of each task. And, occasionally they exhibit randomness and interdependence, occurring unpredictably while requiring certain tasks to precede others. Process scheduling contains a large set of different problems with some common characteristics such as dependencies, resource constraints, sometimes resource availability uncertainty and so on. Depending on the process aim, scheduling can be optimised for time or production output, taking into account parameters affecting the problem such as physical volume, production and testing time, curing time, temperature restrictions and so on. *In the spacecraft mission operations capacity, scheduling*[2] *translates to utilising spacecraft resources as best as possible to maximise scientific – or commercial – return, given constraints imposed by technological and environmental parameters.* Depending on mission objectives, tasks (frequently called "activities" in satellite operations) are assigned to particular temporal points throughout a mission time-line, provided that resources suffice to execute this schedule. Normally, scientific mission operations scheduling comprises the "science planning" part and the "control" part. Science planning refers to collecting and serving all science observation requests, resolving potential conflicts and respecting mission constraints. Control planning refers to activities moderating the satellite attitude, communications, data flow, power generation, energy storage & consumption and so on.

Based on the traditional mission operations approach, a satellite's operations schedule is normally formed manually for three temporal horizons (trimester, monthly, weekly), where each satellite payload team composes a plan of expected activities for their respective instrument. Operations engineers check all science activity plans

[1] `http://www.collinsdictionary.com/dictionary/english/schedule`

[2] used interchangeably with the term "planning" in space-related bibliography. When presenting our experimental work, we will use the term "scheduling" to agree with Computer Science terminology. When referring to space mission operations we will be using the term "planning" according to space engineering terminology

for potential constraint violations and merge them with control operations, to devise the final operations schedule [Rabideau et al., 2004; Teixeira de Sousa et al., 2006; Wertz and Larson, 1999; Zender, 2012 - 15; Cruzen et al., 2011].
Scheduling a space mission is normally a lengthy, complex process. Numerous iterations between scheduling personnel and geographically dispersed scientific teams may be required to revise a schedule and resolve possible conflicts originating from mission constraints. Distributed teams handling science and control planning need to collaborate seamlessly and iteratively, to define a schedule that covers all scientific aims and technological needs for each part of a mission. A relatively coarse Long Term Plan (LTP) is devised initially, spanning three months of operations. The LTP is then progressively refined to a week-long Short Term Plan (STP), containing a detailed list of all activities to be performed at every time instance. Every two days, the latest STP is further refined before being sent to the satellite for execution. The process is repeated for as long as a mission is active. What makes operations scheduling more challenging and highly iterative is the lack of accurate models per spacecraft, that can be used to test and simulate human generated schedules [Cesta et al., 2009].
As Computer Science grew in the 20th century, with the help of increasingly capable hardware and advancements in operational research, automation and optimisation techniques became more widely available to a diverse set of communities, including space engineering and mission operations.

1.2 Automatic near-optimal mission operations scheduling

A number of optimisation solutions have been attempted as early as 1980, with NASA developing experimental science scheduling systems for the Space Shuttle [Dupnick and Wiggins, 1980a; Chien et al., 1999] and the Hubble Space Telescope (HST) [Johnston and Miller, 1994; Johnson et al., 1993]. Such solutions aimed at tackling increasingly complex and costly operations. For instance the Space Shuttle was designed to be a reusable launch vehicle able to transport large enough payloads that would comprise either independent missions or modules of a larger mission e.g. a space station. Each launch cost on average 1.5 billion 2010 US dollars[3] with NASA aspiring to launch up to 9 missions annually. Thus careful scheduling of each launch year and eliminating redundancies was pivotal to minimise costs.
An instance of complex satellite mission operations is the HST, which shows the

[3]http://www.space.com/11358-nasa-space-shuttle-program-cost-30-years.html

extent of scheduling effort that goes into a space mission. Operated manually since 1990, HST receives approximately 1000 observation proposals annually out of which 200 are accepted. The accepted proposals represent roughly 20000 individual observations that need to be scheduled. According to the Space Telescope Science Institute (STScI)[4] *"scheduling of the viewing time falls to staff at STScI. [...] technicians must schedule each observation down to a fraction of a second. Observation information such as which instrument to use, what filter to use, and how long the exposure should be must be converted into a detailed technical list of second-by-second instructions"*. Operating the HST costs about $98 million annually[5] constituting an intricate and expensive mission.

As computational capability has been rapidly increasing, space agencies and researchers have been experimenting with scheduling systems, allowing them to increase their productivity and scientific output while decreasing experts' workload. Such solutions can offer the spacecraft operations community tools to reduce costs, improve their mission performance and alleviate humans from complex, cumbersome and repetitive tasks. More recent instances include the NASA ASPEN [Fukunaga et al., 1997] framework operating on the basis of Greedy search or Iterative Repair, the ESA (European Space Agency) APSI (Advanced Planning and Scheduling Initiative) [VEGA et al., December 2006 - December 2008] framework employing Greedy search, Tabu search or Genetic Algorithm-based search applied to the European missions VEX (Venus Express) and MEX (Mars Express). Such endeavours show that there is no single best strategy able to solve effectively all problems within the domain of operations scheduling. Therefore, some significant effort is required for analysing each problem and choosing a specific approach [Verfaillie, 2013].

Satellite mission operations scheduling lends itself to large search spaces, since the number of tasks and schedule duration can vary widely, as can schedule time resolution (time-unit grain). While smaller sized problems can be successfully tackled using complete search algorithms used in Constraint Programming for instance, it is practically impossible to apply similar techniques to larger problems. Naturally, practitioners turned towards approximation algorithms to tackle such problems effectively [Policella, 2013].

A typical mission operations schedule contains temporal and resource constraints. Temporal constraints contain target visibility windows, task durations, no overlap between particular tasks, precedences etc. Resource constraints usually refer to on-board memory and available power, acceptable temperature per instrument and

[4]http://hubblesite.org/the_telescope/team_hubble/
[5]http://www.space.com/20799-hubble-space-telescope-23-years.html

total operating duration among others. There are some logical constraints as well, such as that data cannot be downloaded unless relevant observations have been performed or that to perform a particular observation, the appropriate instrument must first be turned ON. On top of constraints considerations, there are features that need to be accounted for such as: observations' priority (weight) based on their importance, probability of observations' success based on parameters such as weather, observation angle affecting quality etc. Finally, observations should be scheduled in a fair manner, not exhibiting selection bias towards one instrument over others. In terms of problem size, an average daily operations schedule contains a few hundred pending user requests, a few thousand candidate observations and a few hundred successful observations to be downloaded [Verfaillie, 2013]. Satellite mission operations scheduling is a complex enough procedure that could benefit from optimisation efforts like the aforementioned. Algorithmically speaking, such scheduling problems are challenging enough, requiring attention to restrictions imposed by the problem itself (e.g. constraints, requirements) and its reduction (e.g. search space, fitness function).
This is where approximation methods (in our case, predominantly metaheuristics) come in useful, for tackling larger scale and possibly less well defined problems in reasonable time [Luke, 2013]. To understand better why metaheuristics are considered a promising tool for this job, let's take a step back for a moment and view optimisation from a more general viewpoint.
Overall the term optimisation refers to the process of finding the best possible solution to a problem. Once we formalise the problem in question, we employ strategies to tackle it. Then we verify our blueprint and computationally validate that it returns a globally optimal solution. Optimisation algorithms are divided into two categories, those searching solutions in a *real-valued* search space and those performing their search in a *discrete* search space. Discrete search spaces are found in combinatorial optimisation problems. A combinatorial problem is considered solved when a solution presents the globally optimum objective function value. Some typical combinatorial problems are the Travelling Salesman Problem (TSP), the QAP (Quadratic Assignment Problem) or scheduling and timetabling problems. Combinatorial optimisation algorithms can be classified in two main categories, namely complete and approximate algorithms. A combinatorial optimisation problem of finite size is guaranteed to be solved to optimality in a bound time by any complete method. One needs to be aware though that when combinatorial problems are NP-hard [Garey and Johnson, 1979] and provided that $P \neq NP$, no polynomial time algorithm exists resulting in exponential computation time in the worst case. In

practice this trait can lead to unrealistic computation times. For that reason, approximate methods have been receiving increasing attention since the 1970s. They offer a good trade-off between global optimality and practically feasible computation time. According to Blum et al. [Blum and Roli, 2003], two main approximate method strategies are normally employed, *constructive methods* and *local search*. Algorithms working constructively begin by generating solutions from scratch, adding one element at a time to the solution, until the solution is considered complete. They normally perform faster than other approximation algorithm variants, but are not always as effective since they construct a solution by choosing locally optimal elements. On the other hand, local search algorithms begin by constructing an initial solution using some strategy e.g. randomly, and then work iteratively towards replacing the solution with a better one in the search neighbourhood. It is also possible to combine both methods, improving our search capability towards finding global optima.

1.3 Thesis contributions

The main research question driving this thesis was: *Can the process of generating satellite mission operations schedules be automated?*

What has this thesis attempted to achieve: To attempt addressing this research question, we focused on four main points.

1. Initially, we enquired into *the possibility to numerically model with sufficient accuracy a generic satellite's state, so that proposed operations can be simulated.* It turns out it is possible to approximate, with satisfactory fidelity, general satellite functions. This includes on-board satellite characteristics such as power consumption, generation and energy storage as well as orbital dynamics such as state vectors (position and velocity per time unit). This allows for simulating a satellite mission to a satisfactory level of detail for our purposes.

2. Following this, a question that arises is *if it is possible to construct an algorithm that can generate satellite operations schedules, based on information received from the aforementioned computational model.* While the problem at hand is a scheduling one, it can be considered idiosyncratic. Therefore, given the lack of prior knowledge of which algorithm will do best or even be suitable,

it is advisable to compare different existing algorithms – as has been demonstrated in Computer Science literature for decades.

3. This leads to a subsequent twofold question: *can more than one algorithm be applied for solving this problem? If so, which algorithm performs better?*. To address this question, a small test case was put together, based on a high orbit LEO satellite. Experimenting with different algorithms showed that more than one algorithm can tackle this problem, with a varying degree of success. Quantitative analysis allowed us to infer which algorithm can generate fitter schedules and how processing time differs between algorithms.

4. Naturally, the final question to ask is *how do the algorithms tested scale up to a realistic satellite operations scheduling test case?* In order to answer this question, the challenging task of collecting information and data from ESA and the industry was undertaken. Extensive discussions with satellite mission operations experts from ESA and the industry took place, allowing us to form a realistic test case scenario based on a real LEO Earth Observation mission. We then applied our algorithmic set on it, to observe if and how operations schedule generation may differ.

What this thesis does not aim to do: This thesis is not aimed at performing a comprehensive comparison of approximation algorithms in general, or for this particular application. Neither does it offer a complete solution towards satellite mission operations scheduling. The idea of fully automating the process of building satellite mission schedules is still work in progress for the space operations community at large. It has been investigated since the early 1980s if not earlier, with prominent examples such as the Space Shuttle and HST being strong cases for investing more effort towards this direction.
More recently, with the advent of small-scale satellite swarms, automating many aspects of satellite operations including the very important scheduling component is becoming a pressing priority, both for space agencies but importantly for private enterprises, which strive to maximise return on investment.

1.4 What renders this problem difficult

By definition a satellite contains constrained continuously changing resources onboard, while serving as an orbiting observatory (science missions) or relay platform (telecommunications). At LEO for instance, a satellite like the ISS (International Space Station) orbits the Earth at approx. 400km[6] above mean sea level, with a speed of approximately 27610km/h (7.67m/s) [Battin, 2008]. Such high speeds combined with the Earth's rotation mean that visibility windows' occurrence and duration for targets of interest continuously change. Similarly, power generation and energy storage or usage vary.

Furthermore, unexpected events occur occasionally. For example, a spontaneously occurring Solar Coronal Mass Ejection heading towards the Earth can incapacitate satellites if not put in safe mode [CCSDS, 2001, 2004] promptly. Recovering satellite functionality afterwards requires a different scheduling sequence depending on possible technical faults and resource availability changes (telecoms, battery and power generator health). Similarly, if such a Solar event does not pose a threat to the satellite it is scientifically useful to observe it, requiring extra observation tasks to be promptly inserted in the mission schedule.

Other unprecedented events such as technical faults, may also require changing operation activities to meet newly imposed constraints e.g. following a solar array connection fault, MEX entered Martian orbit with 30% less power generation capability on average, aside from the expected solar array degradation.

Satellite mission operations comprise a dynamic constrained problem that superficially resembles other well studied scheduling problems like timetabling, but cannot be categorised as one without simplifications. At every time-step, we experience variation in resources. For instance battery energy availability decreases during eclipse periods and power output fluctuates depending on sunlight's angle of incidence on solar arrays combined with satellite controlled spin rate or uncontrolled tumbling.

1.5 Structure of this thesis

The structure of the remaining thesis chapters is as follows: Chapter 2 contains bibliographical information forming the foundations of this thesis. An overview of selected literature on algorithmic comparison attempts for engineering applications is presented, as well as a survey of publications on satellite operations scheduling optimisation. The chapter concludes by summarising relevant bibliography on this

[6]Annual mean ISS orbit height http://heavens-above.com/IssHeight.aspx

subject.
Chapter 3 describes the computational model used in the course of this thesis to test our set of algorithms. A modular system was developed, approximating a generic Earth Observation satellite, its resource capabilities, its subsystems & payloads and target visibility occurrences based on the satellite's orbital parameters. Mission objectives were passed to the framework, generating resource demand and availability at every time-unit. To conclude the chapter, we summarise the structure of this framework that approximates satellite functionality.
Chapter 4 summarises our approach towards formalising the real life engineering problem of satellite mission operations scheduling. A set of scientific and control tasks need to be scheduled during feasible and useful time windows, which frequently change depending on mission requirements or technical and environmental constraints. This chapter then presents a preliminary comparison of the aforementioned set of algorithms on a small scale test case, acting as a preamble of each algorithm's performance on this particular problem expression. We test all algorithms in their regular form and also combined with LS to observe the amount of fitness improvement that is achieved. A summary of findings from this preliminary comparison concludes Chapter 4.
Chapter 5 comprises a real-life inspired test case, illustrating how well can each algorithm scale up. All algorithms are put to the test generating operations for two different temporal horizons, namely 1 day and 7 day long operations schedules with a resolution of 30 seconds per time-step. We note how has algorithmic performance changed, possibly rendering some algorithms more scalable than others. Furthermore, we note that combinatorial explosion restricts the problem size to moderate instances, rendering the problem expression itself less scalable. We finally summarise our findings to conclude this chapter.
Finally chapter 6 summarises our work and touches upon possible future directions.

"An algorithm must be seen to be believed."

– Donald E. Knuth, Computer Scientist

2 Optimising satellite operations scheduling

2.1 Overview of algorithmic applications

According to the Merriam-Webster dictionary, the adjective *heuristic* translates as "using experience to learn and improve". The word derives from the ancient Greek word ευρίσκω which translates into "search". The term *metaheuristic* – first coined by F. Glover in 1986 [Glover and Sörensen, 2015] – contains the prefix *meta-* which translates as "beyond" in the sense of high-level. As Glover and Sörensen define the term in their contribution at the Encyclopedia of Operations Research and Management Science: *"A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms"* [Gass and Fu, 2013]. Being approximate search methods, metaheuristics are designed to find a satisfactory solution within a short enough computational time span, unlike exact methods that guarantee to find the optimal solution within a finite (but often practically impossible) amount of time. Exact methods' computational time increases exponentially with problem size when exploring a NP-hard problem search space [Glover and Sörensen, 2015]. Therefore, metaheuristics are a good choice when a satisfactory solution is required in a relatively short computational time.

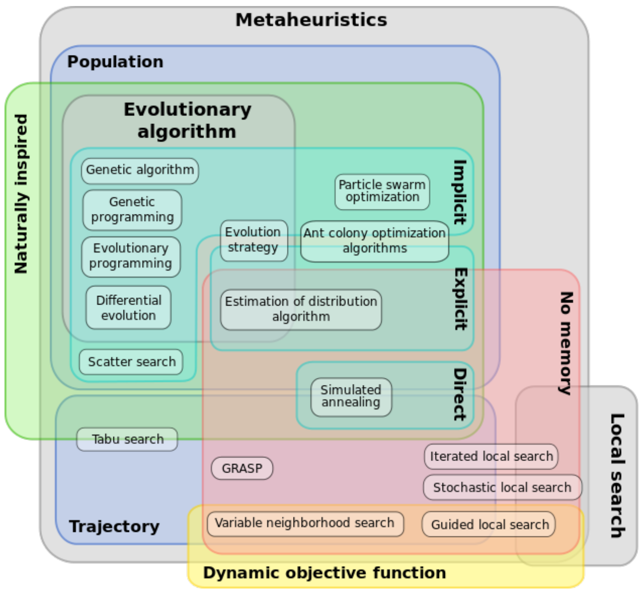


Figure 2.1: Different classifications of metaheuristics, by Johann Dréo and Caner Candan, distributed under a CC BY-SA 3.0 license.

Metaheuristics are classified in different algorithmic families, depending on some key characteristics, offering researchers a number of options to tackle problems effectively. As seen in Fig. 2.1, metaheuristics can be categorised based on their population, process inspiration, memory and other attributes. The main metaheuristics' classification criteria according to Blum et al. [Blum and Roli, 2003] usually refer to the following attributes:

1. Process origin. An algorithm's concept origin (natural or artificial) is one way to classify it. This classification is not always applicable or meaningful due to the presence of hybrid algorithms relying on both natural and artificial strategies. Nature-inspired algorithms draw inspiration from biological processes (e.g. Genetic Algorithm), animal flock or insect dynamics (e.g. Ant Colony System), microorganism behaviour (e.g. Bacteria Foraging Optimisation) and others. Non-nature inspired algorithms rely on semi-random numerical methods (e.g. Tabu Search, Iterative Local Search).

2. Population. Another criterion used for an algorithm's classification is its search agent population. Single point search methods, also known as trajectory methods, produce a single solution at a time describing a trajectory within the search space landscape. Population-based methods scatter a number of search agents around the search space, forming multiple solutions and frequently exchanging information in the process.

3. Objective function usage. Depending on the way an algorithm makes use of the objective function, we can classify algorithms into dynamic and static objective function ones. An objective function is kept “as is” by the algorithm during its search (static), utilising other mechanisms to escape local optima and share information (e.g. Ant Colony System local and global pheromone processes respectively). When experience accumulated during the algorithm’s search is incorporated into the objective function and/or the algorithm attempts to surpass local optima, the objective function is altered rendering it dynamic (e.g. Guided Local Search).

4. Neighbourhood structure. The majority of metaheuristics operate on a single neighbourhood structure, maintaining a fixed fitness landscape topology during an optimisation run. In an attempt to diversify an algorithm’s search, an algorithm can utilise a set of neighbouring structures (e.g. Variable Neighbourhood Search) switching between different fitness landscapes.

5. Memory usage. A further classification criterion used is whether an algorithm makes use of its search history or not. Memory-less algorithms define their next action based on the current state of the search process (e.g. Random Walk). Algorithms with memory utilise past experience, taking into account previous findings before committing to the next step (e.g. Differential Evolution).

Diversity, while useful as a way of providing different vantage points, can also cause confusion in computational intelligence. Since metaheuristics approach a problem stochastically using different strategies, it is safe to assume that not all algorithms could tackle a given problem equally well. Quoting Braun et al. [Braun et al., 2001], whose paper is considered seminal on the subject of algorithmic comparison: *“Selecting the best heuristic to use in a given environment, however, remains a difficult problem, because comparisons are often clouded by different underlying assumptions in the original study of each heuristic.”*.
It is this fact, the difficulty of deciding the best algorithm for a particular problem, that motivated our decision to compare many algorithms for the problem at hand. The scientific community researching satellite mission operations has generated genuinely interesting knowledge, identifying the need and meriting the utilisation of optimisation methods for planning and scheduling. Nevertheless, the very same dilemma arises as to which is the best algorithm to use for a mission operations scheduling application. And the answer to this dilemma has not yet been addressed adequately by the space engineering community, since we find only few mentions of algorithmic comparison [Globus et al., 2003; Barbulescu et al., 2002; Zufferey et al.,

2008; Barbulescu et al., 2004; Wei-Cheng et al., 2005].
Before presenting relevant literature on the subject of satellite mission operations scheduling, we find it interesting to present a general overview of comparative studies on applying optimisation algorithms to various types of engineering problems.

2.2 Description of algorithms compared in this thesis

In this thesis, algorithms from different categories within the framework of metaheuristics mainly, were utilised to address the same discrete optimisation real-life problem. Namely, scheduling the operations taking place on-board a generic small-scale satellite. Alphabetically, the algorithms used were Ant Colony System, Differential Evolution, Genetic Algorithm, Greedy Search, Repeated Random Search and Simulated Annealing. All of them are well known and have been used for solving discrete optimisation problems including scheduling applications in the space domain.

2.2.1 Ant Colony System

Ant Colony System (ACS) [Dorigo and Gambardella, 1997; Dorigo and Stützle, 2004; Bonabeau et al., 1999; Brownlee, 2011] – an improvement over the original Ant System [Dorigo et al., 1996] – is a nature-inspired population metaheuristic with memory, utilising the synergistic example found in insect populations such as ant colonies. First published in the IEEE Transactions on Evolutionary Computation, ACS was applied on the Euclidean TSP (Travelling Salesman Problem), a classic combinatorial optimisation benchmark[1].
In nature ants work collectively, searching for food in a predominantly random fashion and relaying information to the rest of the colony through pheromone deposition. Pheromone evaporates over time, thus less frequently visited paths become less attractive. Shorter paths tend to be visited more frequently. Therefore, pheromone on those paths becomes more prevalent, since it takes less time for an ant to complete a full return trip, rendering it a more attractive route for the ants to follow. It is this basic communications system that allows ant colonies to find the shortest path possible between a food source and the colony nest. Inspired by this process, ACS is a population metaheuristic allowing for a self-organising set of agents to explore the search space in a random way, while exchanging information through

[1]A TSP comprises a set of n cities to be visited exactly once with a minimal cost (shortest path possible). The final path is the most efficient closed one (starts and ends on the same city) [Weisstein, 2013]

pheromone deposition / evaporation and accounting for distance from target in their decision process. Initially, agents perform a random search, with a small probability of exploiting known information (strongest pheromone trail and shortest distance from target). Once a colony has finished its search, the best solution of the current iteration receives pheromone reinforcement inversely proportional to the Euclidean length of this tour, while pheromone found in the rest of the search space decreases by a small amount (pheromone evaporation). Once the simulation stopping criterion is met, a global best solution – which contains the highest concentration of pheromone – is returned as the optimal solution.

2.2.2 Differential Evolution

Differential Evolution (DE) [Storn and Price, 1997; Price et al., 2005; Brownlee, 2011] – originally published in the Journal of Global Optimization – is a population metaheuristic with memory, utilising the concept of weighted vector differences to explore the search space concurrently and converge to a global optimum. It is designed with four main criteria in mind: 1) it is a stochastic direct search method, allowing for non-linear, multimodal, non-differentiable cost functions, 2) it is designed with parallelism in mind, allowing for agents to perform a parallel search within the problem's search space in order to accelerate the convergence to a global optimum, 3) it contains a minimum number of control variables, rendering the technique easier to use, 4) it demonstrates good convergence consistently on independent trials according to Storn et al. [Storn and Price, 1997]. Initially, a population of candidate solution vectors is selected randomly. Every pair of vectors, each representing one possible solution, is recombined (mutation) and a third vector is produced from the weighted difference of each pair of parent vectors. The resulting vector is then mixed with (crossover) another predetermined vector (target vector). If the resulting vector yields better fitness than the target vector, it is selected to replace the target vector in the next iteration. The process is repeated until the simulation's stopping criteria are met, with the result approximating a globally optimal solution.

2.2.3 Genetic Algorithm

Genetic Algorithm (GA) [Mitchell, 1998, 1995; Gendreau and Potvin, 2010; Brownlee, 2011] is a population metaheuristic with memory, applying the concept of adaptation found in evolutionary biology to promote "survival of the fittest". Loosely inspired by natural evolution, GA employs the notion of parents passing their genetic material to their offspring following the process of crossover. Genetic variation happens through random mutation. Typically, in the first generation each parent is represented through a randomly generated solution vector (chromosome). A pair of parents are selected at random and one or multiple pivot points (single locus or multiple loci) are chosen to divide each parent's chromosome. A crossover process then takes place, whereby the first parent donates half of their chromosome information to one offspring individual, with the other half of that offspring's genetic material originating from the second parent. This process generates in total two offspring per pair of parents, containing a combination of their parents' genetic material. A mutation operation then takes place, slightly altering each offspring genetic material with a low probability. The fitness of each offspring individual is calculated and through a selection mechanism, a new generation takes place with new parents being selected to produce the next generation of offspring. Through this mechanism, fitter individuals survive for numerous generations and prevail over the rest of the population, thus providing an approximation of the globally optimal solution.

2.2.4 Local Search

Local Search (LS) [Gendreau and Potvin, 2010; Brownlee, 2011; Luke, 2013] forms the foundation of most methods employing heuristic search. It is a simple strategy working its way fast through large discrete search spaces, sampling in the broader neighbourhood of candidate solutions and refining solutions to approach local optima. Normally, LS commits resources early on, allocating a locally best solution per step, incrementally attempting to form a globally optimal solution. At first, the algorithm generates a full solution using a predefined strategy e.g. random, greedy search etc. Next, it moves to a neighbouring solution and evaluates the effect of this move on the solution fitness. A neighbouring relation is described based on the search space, and it is possible to have more than one neighbourhood (2-opt, 3-opt or more generally k-opt).
This process is performed for a number of iterations, a time limit or until the solution is considered globally optimal (e.g. when a precise mathematical expression of the problem landscape is known).

2.2.5 Repeated Random Search

Random Search [Karnopp, 1963; Gendreau and Potvin, 2010; Brownlee, 2011; Luke, 2013] is a direct search method that does not rely on derivatives. It generates a random candidate solution vector using a uniform probability distribution, performing undirected exploration of the search space.

Random Search can be one shot, comprising a starting solution for another method. E.g. using Random Search, the initial population of a Swarm Intelligence algorithm can be generated. However, it can also be a repeated technique (Repeated Random Search), generating a multitude of solutions and keeping the fittest one. For instance, Repeated Random Search can act as a benchmark compared to other algorithms. Furthermore, on small enough search spaces, it could potentially generate a satisfactory solution.

2.2.6 Simulated Annealing

Simulated Annealing (SA) [Kirkpatrick et al., 1983; Dowsland, 2012; Gendreau and Potvin, 2010; Bertsimas and Tsitsiklis, 1993; Brownlee, 2011; Skiena, 2008] is a metaheuristic inspired by the physical process of annealing, a heat treatment method used in metallurgy to increase material ductility and promote material homogeneity. A metallic sample is heated above a certain temperature threshold and then gradually cooled down in a controlled fashion, to eradicate dislocations, rendering it more malleable.
Initially, a random solution is formed and its fitness is calculated. A neighbouring solution is selected and its fitness is compared to the initial one. If the new solution has superior fitness, it is accepted. Otherwise, the lower fitness solution is accepted with a probability calculated by the Boltzmann distribution[2] that is higher when the system temperature is high, and lowers as the temperature decreases. This means that solutions of lower fitness can be accepted especially during the early stages of optimisation, allowing for escaping local optima and encouraging exploration of the search space.
It is worth noting that prominent researchers such as S. Skiena consider SA to be "the most reliable method to apply in practice" [Skiena, 2008].

2.2.7 Summarising algorithmic control parameters

The main control parameters found in the above algorithmic set are, alphabetically:

ACS: Ant colony size, stopping criterion, exploration vs. exploitation probability, initial pheromone level, pheromone amplification weight, heuristic amplification weight.

[2] $p(\delta\mathcal{E}) = e^{(\frac{-\delta\mathcal{E}}{kT})}$ with $\delta\mathcal{E}$ is an increase in energy, k is the Boltzmann constant and T the current system temperature.

DE: Population size, stopping criterion, crossover probability, amplification factor, exploration strategy.

GA: Population size, stopping criterion, crossover probability, mutation probability, selection strategy, crossover strategy.

LS: Stopping criterion.

Repeated Random Search: Stopping criterion.

SA: Initial temperature, cooling schedule, stopping criterion.

2.3 Applications of metaheuristics on satellite operations

This section contains selected literature on applying metaheuristics to address spacecraft operations scheduling. Publications included focus on unmanned missions, or operations taking place before a manned mission was launched (Space Shuttle servicing). The criteria we applied for including the following literature were a) to gain insights from past applications of metaheuristics which might be useful here, b) to review the state of the art in applying metaheuristics to space operations problems. In the space industry, numerous publications have attempted to support mission operations through the use of Artificial Intelligence using real-life missions as test cases. Existing literature deals with both planning and scheduling in the area of mission operations. While planning refers to the process of synthesising single steps into a coherent plan to achieve a target, scheduling deals with the temporal and resource information related to that plan. It is not common to find pure planning or pure scheduling problems in space applications, they are rarely separable [Fratini, 2013], thus both problems have been studied in an integrated way [R-Moreno et al., 2008; Policella, 2013].
We will therefore continue to refer to our research problem as "scheduling", supported by bibliography that makes reference to both planning and scheduling in space applications. Operations are normally divided into three temporal duration periods; short term, medium term and long term planning, STP, MTP (Medium Term Plan) and LTP (Long Term Plan) respectively [Cesta et al., 2008, 2009, 2011]. Normally the scheduling process starts by devising a high-level LTP corresponding to three or

six months of operations, whereby science targets are decided and allocated to this temporal period. Planning experts then focus on monthly activities representing the mission's MTP and finally devise an STP i.e. a detailed weekly schedule, with a second week scheduled for contingency. Finally, the mission's schedule is further refined and checked every two days to produce the commands that will be sent to the spacecraft for execution [Cesta et al., 2009, 2011].

Two of the most established research groups on advanced satellite operations are ESA's AMCTO (Advanced Mission Concepts and Technologies Office)[3] and NASA's AIG (Artificial Intelligence Group)[4], with the French Aerospace Lab – ONERA (Office National d'Etudes et Recherches Aérospatiales)[5] and the Italian Institute of Cognitive Sciences and Technologies – ISTC-CNR (Institute of Cognitive Sciences and Technologies – Consiglio Nazionale delle Ricerche)[6] making significant contributions to the spacecraft mission operations community. The AMCTO's contribution since 2009 has focused around APSI[7], a research project with a twofold goal. On the one hand, APSI strives to improve the cost-effectiveness and flexibility of mission planning through development of a new software framework that was tested and validated on a number of case studies. On the other hand the project aims to bridge the gap between space mission operations and AI (Artificial Intelligence) according to AMCTO's mission statement.

2.3.1 ESA AMCTO research

ESA AMCTO initially focused on the LTP studying three ESA missions, namely MEX, INTEGRAL and XMM-Newton, with diverse mission scopes. The MEX-LTP, a multi-objective problem in nature, was reduced to a single-objective problem by aggregating various objectives in a common expression preserving the semantics of each individual objective. A tool called MrSPOCK [Cesta et al., 2009, 2011; Steel et al., 2009] was developed, aiming at achieving a higher scientific return by preallocating maintenance tasks optimally [Donati, 2010]. The challenge faced in MEX-LTP was to reduce the time spent iterating between two separate teams, namely the science planning team and the mission planning team, as well as optimising objectives defined according to mission planner needs. The science planning team collects all science requests from Principal Investigators of the MEX mission payloads, compares

[3]http://www.esa.int/Our_Activities/Operations/Overview

[4]http://www-aig.jpl.nasa.gov/

[5]http://www.onera.fr/

[6]http://www.istc.cnr.it/

[7]http://www.esa.int/Our_Activities/Operations/APSI_br_Advanced_Planning_Scheduling_Initiative

them against constraints, iterates with payload teams to resolve any conflicts such as resource violations and passes the final science plan to mission planners. Mission planners in turn merge both science and platform operations into one timeline, check against constraints, communicate any conflicts and try to resolve them successfully before generating the overall mission plan and satellite commands to be uplinked.
MrSPOCK utilises a simple GA as the basis of its solver, with a greedy counterpart acting as a decoder of the binary solution into the common format used throughout the software tool. According to Cesta et al. the choice of algorithm was made based on the multi-objective nature of the problem at hand. While choosing an evolutionary algorithm to tackle a difficult problem agrees with Computer Scientists' intuition, the reasoning behind using GA for MrSPOCK does not provide an objective metric as to why this particular algorithm was used. Since the problem was reduced to a single-objective one and given a carefully thought problem expression, the same single-objective optimisation problem could be tackled using different metaheuristics.
Following MrSPOCK, two more tools were developed by the same team to achieve increased flexibility in astronomical mission science scheduling, constituting the process faster and easier while generating optimal robust schedules. In order to serve the needs of LTP observations for the INTEGRAL (INTErnational Gamma-Ray Astrophysics Laboratory) mission, the AIMS (APSI Integral Mission Scheduler) [Pralet and Verfaillie, 2009] tool was developed. Likewise, in order to serve long-term observational needs of the XMM (X-ray Multi Mirror)-Newton mission, the AIMS legacy was used and enriched accordingly to reflect mission objectives, in a platform named XMAS (Xmm Mission Apsi Scheduler) [Castellini and Lavagna, 2009].
AIMS was designed to schedule observations automatically for the time span of a year normally, meeting observations requests submitted by scientists around the world. Each request could take more than one observation period to be completed and depending on the request observations could be periodic, urgently requested or spread over a period of time. A target allocation committee would normally select observation requests and assign priorities on each request such that the majority of observations are considered complete by the end of a particular observation period. The objective of this process was to perform as much of each observation as possible, as well as feasible requirements and constraints such as the existence of narrow observation windows and non-overlapping observations.
AIMS employed a local search algorithm. A pre-selection of a subset of local moves would initially take place as a means of decreasing the set of possible local moves available per step, which – if considered in full – would render search very slow and

impair global algorithm performance. The pre-selection process combined heuristic and random choices among observation requests to create each subset of possible local moves. A tabu list was used as a mechanism of avoiding local optima, excluding a specific number of previously taken steps. To ensure diversity of possible solutions the algorithm would be restarted after a number of consecutive steps leading to no significant improvement. Pralet and Verfaillie justify the choice of local search in AIMS based on the size of their problem instances which *"precludes the use of complete optimal algorithms, such as systematic tree search"* [Pralet and Verfaillie, 2009].

The term "systematic tree search" can be vague but likely refers to graph traversal, an exhaustive search strategy normally. Pralet et al. correctly underline the preclusion of exhaustive techniques due to problem instance size, arguing that utilising a local search algorithm with tabu search and restart options allows for efficient observation scheduling optimisation. Nevertheless, while local search is a powerful and a well used strategy, it is one of many techniques in the heuristic or approximation algorithm domain designed to tackle combinatorial optimisation problems. Thus choosing local search does not seem to be objectively justified.

Related work supported by AMCTO researchers includes managing multiple Earth observation spacecraft. Spacecraft entities can be seen as multi-agent systems, since they frequently form constellations. Examples of constellations include: i) multiple spacecraft scattered around slightly different orbits to achieve a wider field of view of events/provide telecommunication and navigation capabilities etc., ii) clusters i.e. close formations of spacecraft in order to achieve a particular scientific target for instance ultra high resolution measurements of physical quantities such as gravitational fields, iii) swarms i.e. multiple small spacecraft scattered around slightly different orbits to achieve a wider field of view predominantly used for EO (Earth Observation). The problem of operating those satellites in a satisfactory manner such as to maximise their return, rendering their mission a success, becomes a multi-agent organisation problem. It is therefore intuitive to examine multi-agent techniques that could possibly tackle such a problem automatically and optimally. Iacopino et al. [Iacopino et al., 2011; Iacopino, 2012; Iacopino and Palmer, 2013; Iacopino et al., 2013] studied the multiple spacecraft problem for EO, opting for a stigmergy-based solution found in insect populations such as ants. Utilising the ACO (Ant Colony Optimisation) principles, treating a spacecraft constellation as a self-organised multi-agent system. In their 2011 publication [Iacopino et al., 2011], Iacopino et al. recognise the division of self-organised multi-agent systems in four main categories, namely "*agent cooperation using negotiation paradigm, agent learn-*

ing by means of reinforcement, simple direct interactions or indirect interactions like the stigmergy paradigm". They choose to use the paradigm of simple direct interactions as a promising method, supporting this choice through the claim that it has been demonstrated to achieve complex system behaviours.

While stigmergy is by all means a powerful strategy, proven to solve complex problems very well, it is not the only multi-agent method that has been demonstrated to achieve complex dynamics. Examples such as Differential Evolution and Genetic Algorithm, both well studied powerful multi-agent self-organising systems, can also be considered well suited for the task. It would be interesting to see a comparison in that sort of work, further supporting the researcher's position.

2.3.2 NASA centres' research

The NASA AIG started working on the concept of automated planning and scheduling for space mission operations in 1999, where they first introduced their experimental system ASPEN. The initial goal of ASPEN was to demonstrate that by encoding flight rules, modelling spacecraft and their operations it is possible to generate low-level spacecraft commands corresponding to high-level science and engineering goals. Introducing automation paves the way for smaller operations teams thus cutting down operations costs [Chien and Rabideau, 1997; Rabideau et al., 1999]. ASPEN's solver is based on iterative repair, a technique earlier used by Zweben et al. [Zweben et al., 1993] in 1993 to assist in the coordination of Space Shuttle ground processing which was a constrained complex scheduling problem. Throughout the repair process, the algorithm can alter a schedule using ten possible ways: it can move an activity, add a new activity instance, delete an activity, abstract an activity, make a reservation of an activity, cancel a reservation, connect a temporal constraint, disconnect a constraint or change a parameter value. Deciding which of those methods to use depends on the type of conflict at hand [Rabideau et al., 1999]. That way, an initial schedule containing conflicts or a partially formed schedule can be shaped in such a way that conflicts are resolved given mission requirements and constraints.

Further ASPEN applications include the ASPEN-RSSC (ASPEN-Rosetta Science ground segment Scheduling Component) [Chien et al., 2014] assisting operations of the Rosetta mission that has been studying comet 67P/Churyumov-Gerasimenko since August 2014, as well as ASPEN assisting operations of the Dawn mission that studied asteroid Vesta in 2011 before proceeding to dwarf-planet Ceres in May 2015 [Rabideau et al., 2014].

While ASPEN introduced the concept of conflict resolution in near-real time, allow-

ing for greater flexibility and improved practice in mission operations, there is no mention of optimisation in the process. Thus the problem of achieving the highest possible return from an expensive and rapidly expendable observatory has not been explicitly addressed through this work, allowing enough room for improvement. Spacecraft are very expensive objects built to meet a number of requirements that render a mission successful. Focusing on conflict resolution without explicitly accounting for optimising operations leads to a convenient but not complete solution, allowing for more room for improvement as a means of maximising return on a large investment.
Astronomical applications have been very interesting for researchers in the field of space mission operations scheduling, due to the number of scientific observation requests that need to be scheduled yearly, with each request comprising multiple observations to be completed due to scientific or technical constraints. The STScI has developed and used SPIKE [Johnston and Miller, 1994; Giuliano, 2014, 2013] since 1988, a system initially used for long-term scheduling of the HST and later applied to another six orbital astronomical observatories and three ground-based ones. The SPIKE scheduler is based on multistart stochastic repair, a heuristic repair-based technique incorporating three main steps: *stochastic initial guess, repair, deconflict.* Initial guess is based on the heuristic of "*most-constrained activities first served*" before assigning other activities. A repair heuristic is then applied, moving activities to times where conflict occurrences are minimum. Finally, a deconflict step takes place removing lower priority activities, lower preference or higher number of constraint conflict instances. Any remaining gaps in the processed conflict-free schedule are filled using a best-first search over a pool of unscheduled activities. Bearing in mind that in astronomical observation time boundaries are fixed, the aim is to maximise the quantity and quality of observations within that time-frame. The solution fitness is calculated as the sum of total minimum activity duration plus the total gap time. When rescheduling is required, two mechanisms assist in successfully rescheduling a conflict-free time-line, namely, task locking and conflict-cause analysis. In task locking, selected tasks are kept fixed on the time-line whereas conflict-cause analysis allows the user to force tasks into the schedule and manually resolve conflicts e.g. through moving conflicting tasks in the pool of unscheduled tasks.
SPIKE has proven robust enough to be used for decades in the domain of science operations scheduling for astronomical missions. It provides automation of conflict resolution through a smart heuristic multi-step strategy. Nevertheless, it does not seem to address science operations optimisation. It therefore allows room for im-

provement in order to increase solution quality and make the most of each orbital or ground based observatory.
NASA introduced the Space Shuttle in the late 70s, injecting operational complexity due to mission frequency and ambitious targets set. Space research missions, satellite launch and servicing missions and space station construction missions are some examples of what the Space Shuttle was designed for. One of the earliest instances of optimisation applied to space mission planning and scheduling is SAMPLE (Scheduling Algorithm for Mission Planning and Logistics Evaluation) [Chang and Williams, 1976; Dupnick and Wiggins, 1980b] introduced in 1976, a program that could calculate the minimum of Space Shuttle flights needed to transport a set of specified payloads without expensive redundancies as stated by traditional systems engineering. The solver behind SAMPLE is a simple greedy algorithm, which is a sensible choice for two main reasons: First, at the time SAMPLE was developed, computer processing power and memory availability were significantly restricted. As an example, SAMPLE was implemented on a UNIVAC 1110 with EXEC 8 operating system, constrained to a maximum search space of 100 payloads and 2000 mission combinations due to memory restrictions. Second, at that time more complex metaheuristics were still at their very infancy, but even if they were already well developed it would have been a difficult task to implement them on 70s hardware.
Various planning and scheduling systems were developed for handling Space Shuttle operations, due to the diversity and complexity of its missions scopes. Two interesting examples include a scheduling system for HST servicing missions [Johnson et al., 1993] and a scheduling system for KSC (Kennedy Space Center) payload operations [Pierce, 1987]. Johnson et al. [Johnson et al., 1993] used a system called SM/PART (Servicing Mission Planning and Replanning Tool) originally developed by AlliedSignal Inc. (now Honeywell International Inc.) in 1987 on behalf of NASA. SM/PART uses PARR (Planning And Resource Reasoning) [McLean et al., 1992] as the system's conflict resolution and avoidance engine designed to provide support to scheduling experts. SM/PART introduced automation of the process of building integrated time-lines and command plans for ground personnel working on HST servicing missions. There is no further explanation as to how does the PARR engine works, with both papers consistently using the term "automation" when describing the system's functionality. This leads us to assume that PARR provided a set of heuristics to allow for automated conflict tracking and resolution or avoidance.
Space Shuttle payload installation is divided into horizontal and vertical, with the former taking place in a building further from the launchpad while the latter taking part while the vehicle was on the launchpad. Pierce et al. [Pierce, 1987] presen-

ted EMPRESS (Expert Mission Planning and REplanning Scheduling System) as a solution to assist long range horizontal payload mission planning and scheduling for space shuttle flight payloads at NASA's KSC. Developed by MITRE Corp. in conjunction with NASA KSC personnel in 1986, EMPRESS automatically generated all processes that should be performed in the O&C (Operations and Checkout) building at KSC. EMPRESS determines resource conflicts and allows for interactive conflict resolution as well as maintaining and enforcing constraints when tasks are rescheduled or removed. Unfortunately relevant literature only mentions utilisation of AI powering this expert system, with no further information as to what kind of algorithms are used, search space structure, objective function definition etc.
In order to assist with Space Shuttle ground processing, Zweben et al. [Zweben et al., 1993] introduced the GERRY system used for scheduling each Space Shuttle's inspection, repair and refurbishment. GERRY is based on constraint-based iterative repair which works by generating a complete schedule, with the possibility of flaws being present, and then improving it in an iterative manner given constraints and requirements. At the time of writing, GERRY was used in daily support of Space Shuttle Columbia. The authors empirically show that some problem knowledge can greatly improve convergence speed of an iterative repair-based system, yet too much knowledge hinders performance contrary to human intuition. They conclude that simple random shuffling of tasks violating constraints produces reasonable results on problems of moderate size and difficulty. Lookahead techniques seem to work well on smaller yet difficult problems, nevertheless they are not effective on larger problems. They hope that with the help of machine learning, it will be possible to dynamically switch between different heuristics thus improving solution fitness.

2.3.3 Various groups' research

Earth Observation has also attracted attention from the optimisation community, with particular attention to downlink and ground station scheduling i.e. opportunities for downloading observation data without interfering with the satellite's primary mission milestones or ground station facilities oversubscription. Pralet et al. [Pralet et al., 2012] worked on allocating downlink windows for multiple satellites performing ship surveillance. Surveillance satellite data need to be delivered to users in near-real time for them to be meaningful, therefore timing is very important. Bearing in mind that EO satellites only have contact with any given ground station for 10 minutes on average, it can be challenging to downlink large amounts of data. To make matters more complicated, ground station visibility varies with latitude and satellite inclination, occurring with a frequency of approximately two hours or

more. Furthermore, not all satellite services utilise all ground stations visible at any given point, since contracts need to be put in place and a fee is charged per satellite downlink session. The problem at hand therefore poses the challenges of conflict resolution and downlink window optimisation. The solution proposed by Pralet et al. comprises a mathematical formulation describing the problem, a problem decomposition method for reducing problem complexity and an attempt to optimise the result using MIP (Mixed Integer Programming) methods. The authors mention the possibility of utilising Local Search to solve this problem, but offer no experimental data as a comparison to their original MIP solution. Furthermore, while it is obvious that a great deal of effort has been put towards expressing this scheduling problem as faithfully as possible, this expression will most probably have to be significantly altered for somewhat different observation needs. MIP methods are very powerful but require a faithful detailed representation of any given problem, with the solution being of inferior quality in case the problem expression is not well formed. This leaves room for discussion and consideration of approximation methods, allowing us to generate good solutions from less well defined problems.
In an attempt to address a similar problem, Sun et al. [Sun and Chester, 2010] worked on optimising ground station scheduling using GA. The problem expression is significantly simplified and weight has been given to the design of algorithmic components that allow GA to solve the problem effectively. Multiple fitness functions have been used, each tackling a different objective, aggregating them in a single fitness function expressing all objectives collectively. It is interesting to note that the problem expression is significantly simpler than the one found in Pralet et al. [Pralet et al., 2012], with Sun et al. focusing on the algorithm selection and evolution criteria instead. The advantage to this method is that the problem itself can be less strictly defined, allowing the algorithm to find good solutions through natural evolution despite a less faithful mathematical representation of the problem at hand. Sun at al. have demonstrated that GA (with reasonable customisation) can tackle the ground station scheduling problem effectively, but there is no clear explanation as to they decided to utilise GA in the first place. GA is a well studied technique, allowing people to rely and expand on previous literature, but it is one of many approximation methods that have been studied well enough and shown promising behaviour. It would be interesting to see more than one method being utilised in this work, as a means of not only further exploring the field of metaheuristics but also comparing the effect of the same metrics (e.g. fitness function) in solving the same problem through different techniques.

2.3.4 Attempts of historical interest

Historically speaking, mathematical optimisation has existed for approximately four centuries with Newton, Fermat and their contemporaries manually optimising analytical expressions. Computationally speaking, optimisation has been a much younger discipline spanning a few decades. In space mission planning and scheduling, the introduction of new technologies was driven by the increasingly demanding operational needs of space programs. The following literature is included out of historical rather than contemporary interest, presenting attempts on utilising earlier AI technology for benefiting operations scheduling applications.
Based on the above overview, it is clear that a diverse set of approaches has been utilised to solve a common problem encountered in mission operations, namely scheduling operations on the ground or on-board spacecraft. Some instances of systems developed to tackle similar problems in the domain of satellite mission operations can be found in Table 2.1. Approaching a constrained complex problem from different angles has the advantage of discovering different challenges, problem traits and possible solutions to address this problem. It also creates many parallel strands of research though, which do not always contribute to each other. Different teams build different frameworks that perform similar functions and researchers partially repeat previous work under slightly different assumptions. In an attempt to make a significant contribution to the current body of knowledge, research work is supplemented by inventions such as new languages and new data formats which do come with the cost of an increased learning curve, increased system complexity, lack of inter compatibility and result in a lack of standardisation among others. Furthermore, the community of space operations seems to lack a common problem definition that can act as a benchmark to be utilised by researchers in the field, rendering experiments heterogeneous, making their comparison very challenging.

Table 2.1: Non-exhaustive list of timeline based systems [Chien, 2013]

Tool	Year	Availability
APSI	2009	N/A
ASPEN	1999	Available for external licensing but not export
EUROPA	2007	NASA Open Source Agreement
GMV flexplan	2006	Proprietary
Mexar2	2005	N/A
MUSE	2009	N/A
Pinta/Plato	2000	N/A
SKeyP	2009	N/A
SPIFe	2011	N/A
SPIKE	1988	STScI source licence option

During this research, we discovered a number of different frameworks developed by experienced teams working for ESA and NASA, some of them included in Table 2.1. However we were unable to gain access to any of them to use as a foundation of our research. For instance both AIG and AMCTO have implemented their own framework approximating satellite functions, based on non-publicly available data provided by NASA and ESA missions respectively. None of those frameworks is available for release as open-source (AIG software is available as open-source only to American nationals, pending request, with ITAR (International Traffic in Arms Regulations)[8] rules applying), freeware or otherwise though, despite both organisations being public institutions supported by American and European tax money respectively. Apart from our overview on the algorithmic content and scientific contribution of those publications, we conclude that sharing a widely accepted platform or benchmark representative of the problem at hand will greatly assist toward enriching current bibliography and building on top of previous research to generate better solutions in shorter time frames. Contemporary hardware technology and scientific knowledge allow us to achieve automation and optimisation of mission operations, enabling human experts to focus on innovative, conceptual and creative thinking instead of repetitive problems that can be tackled through technology. For that reason, implementing novel technology in current mission operations should be a mainstream activity instead of experimental research work. And that can only be achieved with collaborative work under similar assumptions with a common aim.

[8]https://www.pmddtc.state.gov/regulations_laws/itar.html

2.4 Tuning algorithm parameters

A pertinent question that arises before using an algorithm for experimentation or production, is how does a practitioner set up control parameters, to ensure good algorithmic performance. Literature mentioning algorithmic parameter tuning contains articles on manually tuning particular algorithms e.g. ACO [Levine and Ducatelle, 2004] or proposing a framework for automated parameter tuning such as ParamILS[9] applied to a particular category of problems like SAT (SATisfiability problem) among others.
Manual parameter tuning has been a common practice in empirical studies, testing each parameter's influence one at a time while keeping the rest fixed (sensitivity analysis) [Santner et al., 2003]. Repeating this process to gather a statistical sample per setup, one can infer a satisfactory parameter set that allows their tuned algorithm to perform well [Morris, 1991; Barr et al., 1995]. This is the method that was applied for tuning our algorithmic set, selecting sensible initial values from a range around the values recommended by each method's authors and running each parameter setup 10 times. The parameter setup with the highest fitness median was selected for performing our scheduling experiments. More details can be found in Chapter 4.
Interestingly, Arcuri et al. [Arcuri and Fraser, 2013] in their engineering-oriented article conclude: *"Using default values is a reasonable and justified choice, whereas parameter tuning is a long and expensive process that might or might not pay off in the end."*, considering the balance between tuning effort and algorithmic performance improvement in a production or research setting.
It is also interesting to see how automated algorithm parameter tuning has been established as a distinct research subject. Selected bibliography follows, for future reference.
The University of Freiburg ML4AAD (Machine Learning for Automated Algorithm Design) group leads research efforts on this topic, following pioneering work[9] from the UBC (University of British Columbia) BETA (Bioinformatics and Empirical & Theoretical Algorithmics Laboratory) in the late 2000s summarised in [Hoos, 2012]. ML4AAD recognise that algorithm parameter tuning is an optimisation problem in itself, publishing a culmination of their experience on the subject [Eggensperger et al., 2017] in mid-2017. Starting from mid-2015, researchers on the topic of algorithm configuration and selection have formed the COSEAL (COnfiguration and SElection of ALgorithms) international research group[10], establishing algorithmic

[9] http://www.cs.ubc.ca/labs/beta/Projects/ParamILS
[10] http://www.coseal.net/

parameter configuration and selection as a distinct research domain.
Through ML4AAD work, we find a review of algorithm configuration literature[11], with the first reference on algorithm configuration dating back to 2002 [Birattari et al., 2002a] configuring metaheuristics for solving the TSP and subsequently influential work published by Hutter et al. [Hutter et al., 2009] in 2009 using a SAT problem as their test case. Bibliography mentions GGA (Gender-Based Genetic Algorithm) [Ansótegui et al., 2009], a variant of GA published in 2009, shown to outperform the Hutter et al. ParamILS method on solving the SAT. In 2010, a software framework for configuring multi-objective ACO algorithms [López-Ibáñez and Stützle, 2010] was proposed for this specific family of algorithms.
A second method of algorithmic configuration was proposed by the ML4AAD in 2011 [Hutter et al., 2011], publicly releasing beta code in 2013 and rewriting it in late 2016[12], tuning a local search algorithm and tree search for solving the SAT and MIP and some AI Planning instances. In 2013, an attempt to optimise planning systems [Vallati et al., 2013] was published using international planning competition instances[13]. The main research focus of ML4AAD and COSEA shifted towards Machine Learning ever since, see for instance [Eggensperger et al., 2014; Hutter et al., 2013, 2014; Feurer et al., 2014; Falkner et al., 2015; Lindauer and Hutter, 2017].
Outwith the aforementioned research groups, we find proposed approaches pertaining to Evolutionary Algorithm tuning such as Sequential Parameter Optimisation [Bartz-Beielstein et al., 2005] applied to PSO or tuning Evolutionary Algorithm [Lobo et al., 2007; Eiben and Smit]. Researchers have also studied the effect of using continuous optimisation algorithms to tune swarm intelligence algorithms like ACO and PSO [Yuan et al., 2012], testing the concept of using an algorithm to tune itself [Yang et al., 2013] and proposing a generic parameter sweep method [Pinel et al., 2012].
Overall we notice that in the last few years, more researchers have recognised the challenge algorithm parameter tuning poses, proposing solutions that potentially increase performance while decreasing configuration effort, time and complexity. There is still a lot of work to be done on this front though, to establish such parameter tuning methods as a de facto standard.

[11]http://aclib.net/acbib/
[12]https://github.com/automl/SMAC3/releases/tag/0.1.0
[13]http://www.plg.inf.uc3m.es/ipc2011-learning/

2.5 Comparative studies on optimising real-life problems

For decades computational optimisation has been researched and applied in numerous disciplines, attempting to solve difficult real-life and theoretical problems. We hereby present in chronological order a collection of relevant selected literature on comparison of optimisation techniques applied in practical problems.

Some of the earliest literature examples attempting to compare optimisation methods come from M. J. Box [Box, 1965, 1966]. In his articles Box compares different algorithms applied to practical constrained optimisation problems. The choice of optimisation algorithms is driven by availability, bearing in mind that in the 1960s software development was considerably more cumbersome than today, with the author comparing Rosenbrock's optimisation method [Rosenbrock, 1960b] with a constrained optimisation version of Simplex method [Wright, 2010] from Spendley at al. [Spendley et al., 1962] in his 1965 [Box, 1965] paper. Box concludes that modifications of parameters for Rosenbrock's, Complex and Simplex methods respectively do not yield significant improvements. He observes that Rosenbrock's method is significantly better at solving unconstrained optimisation problems. Finally the author notes that gradient methods are more efficient than direct-search ones for solving unconstrained problems and expresses the hope that comparison of recently emerged optimisation techniques will take place in the near future.

In 1966, Box publishes another comparison article [Box, 1966] whereby eight unconstrained optimisation algorithms are put to the test, solving highly non-linear problems of up to twenty independent variables. The author extends this work by considering constrained optimisation (only inequality constraints considered) which can be formed such that no explicit constraints appear and thus can be tackled by unconstrained optimisation strategies. Four of the methods studied in Box's 1966 article are direct-search methods not requiring calculation of derivatives, two of them are gradient-based and the final two relying on *"the residuals for each equation instead of merely the sum of their squares"* as mentioned by the author.

Fast forward thirty years, we find another NASA-funded comparison of iterative and evolutionary optimisation algorithms by S. Baluja [Baluja, 1995]. Seven iterative and evolutionary-based methods were used to solve problems from six different classes in an empirical comparison of optimisation heuristics. Quoting the author "One of the goals of this study is to use the algorithms with as little problem-specific knowledge as possible" as problem-specific knowledge is not always readily available in the space domain. Iterative methods could tackle problems with little

problem-specific knowledge availability. A main aim of Baluja's 1995 empirical study was to prove this point. The methods compared in [Baluja, 1995] are three variants of MRSH (Multiple-Restart Stochastic Hillclimbing), two variants of PBIL (Population-Based Incremental Learning) and two variants of GA. Twenty seven problem variants were solved with each of the aforementioned methods, from the problem classes of Travelling Salesman Problem, job-shop scheduling, knapsack, bin packing, neural network weight optimisation and numerical function optimisation. Problems were encoded either in standard binary code or in Gray code[14] and algorithm control parameters were selected empirically. The comparison proves PBIL to be superior to the other two algorithmic categories in 22 out of 27 problems.
In the late 1990s Chun et al. [Chun et al., 1998] performed a comparison of three heuristic algorithms on a number of numerical problems, aiming at taking an informed decision before attempting to optimise the design of a surface permanent magnet synchronous motor. A modified version of IA (Immune Algorithm) (MIA), GA and ES (Evolutionary Strategy) were the algorithms of choice, solving five well studied numerical problems before utilising the best performing one to an engineering application. The problems used for testing algorithmic accuracy and capability are a sphere[15] model, a Rosenbrock function[16] , a step function, a sinc[17] function and a multimodal function[18]. Chun et al. utilising computer science to the benefit of engineering design, correctly proceeded to testing before using, choosing a set of powerful and well rounded benchmarks. They found MIA more able to tackle complex shape problems such as sinc or multimodal functions. ES proved more able in tackling smooth shaped functions such as Rosenbrock's. Considering that the aim of that study is the optimal design of a permanent magnet synchronous motor, the authors chose MIA for their design since the motor's efficiency curve presents complexity similar to a sinc function. Chun et al. conclude that MIA performs better in most cases except from smooth shaped functions, thus it is safe to consider it capable of tackling a wider range of complex optimisation problems.

[14]Also known as "reflected binary code", Gray code was patented by Frank Gray in 1947, representing a binary system where two successive values differ by only one bit. It was originally used to prevent false output from electro-mechanical switches.

[15]A convex, smooth, quadratic function comprising an excellent test for algorithm convergence.

[16]A non-convex function used as a benchmark for optimisation algorithms. It's expressed by $f(x, y) = (\alpha - x)^2 + \beta(y - x^2)^2$. The global minimum lies within a long, narrow parabolic-shaped flat valley, making it easy to trace the valley but difficult to converge to a global optimum.

[17]Unnormalised sinc i.e. *sinus cardinalis* is defined as $sinc(x) = \frac{sin(x)}{x}$. The normalised version of the function $sinc(x) = \frac{sin(\pi x)}{(\pi x)}$ is the Fourier transform of the rectangular function with no scaling. It is used in reconstructing signals from uniformly spaced samples of that signal.

[18]Multimodal functions contain multiple good solutions or global optima. Such functions can usually be solved successfully by Evolutionary Algorithms

The original inventors of PSO (Particle Swarm Optimisation) Eberhart et al. [Eberhart and Shi, 1998] published an article in 1998 comparing GA with PSO. The authors – coming from an Engineering background – state that the main goals of this comparison are to provide an insight into each algorithm's working and suggest ways that could improve performance through interchanging features between the two paradigms. This work stimulated a discussion as to how can elements of GA be incorporated to PSO successfully, enhancing the algorithm's efficiency and effectiveness. Inspired by this discussion, a new PSO feature has been adopted and used routinely by the authors – namely inertia weight – demonstrating the merit of hybridising implementations to improve algorithmic convergence and performance. The authors encourage the continued study and improvement of evolutionary computation methods, utilising benchmarks and importantly practical applications to compare between methods.

It is interesting to see that in 1998 a number of application-oriented comparison articles were published, with GA being a prominent algorithm in all those comparisons. Sexton et al. [Sexton et al., 1998] performed a comparison of GA and backpropagation convergence to optimise an ANN (Artificial Neural Network). The authors mention the propensity of their contemporaries to utilise local search algorithms such as backpropagation to train their ANNs, which may lead to entrapment in local optima and inconsistent or unpredictable performance. Sexton et al. first support this claim through experimental results and then demonstrate the merit of using a global optimisation algorithm. A Monte Carlo comparison on seven test functions was performed, with the GA demonstrating a significantly lower average error than backpropagation. The authors conclude that a well designed objective function utilised by a global optimisation algorithm can achieve a good trade-off, leading to a balanced ANN providing a more robust solution.

In 2001 Braun et al. [Braun et al., 2001] published seminal work on the subject of algorithmic comparison on load-balancing heterogeneous distributed computing systems, the predecessor of modern cloud computing. The authors looked into how eleven heuristics perform on optimally mapping (matching and scheduling) large groups of tasks onto machines comprising a distributed heterogeneous computing environment. This study presented a basis for comparison and understanding under which conditions do some techniques outperform others from the set of algorithms used in this article. Furthermore, the authors add that their study can be used as a starting point for choosing heuristics to be applied in different scenarios.

In 2003 another comparison is published, this time from bioinformatics researchers. Moles et al. [Moles et al., 2003] recognise the difficulty of parameter estimation of nonlinear dynamic biochemical systems, known as inverse problem. Frequently, such problems turn out to be ill-conditioned and multimodal, hindering convergence of gradient-based methods. For that reason, a nonlinear biochemical dynamic model is used as benchmark to test seven global optimisation methods from the deterministic and adaptive stochastic domain. Namely, the methods used are MATLAB's GBLSOLVE [Holmström, 1999] an iterative constrained deterministic optimisation algorithm, MCS (Multilevel Coordinate Search) [Huyer and Neumaier, 1999] a deterministic optimisation algorithm incorporating global search with local search enhancement, ICRS (Improved Controlled Repeated Random Search) [Banga and Long, 1987] a sequential adaptive random search method, DE [Storn and Price, 1997] a population-based stochastic optimisation method, uES (unconstrained Evolution Strategy) based on Schwefel's work [Schwefel, 1995] comprising an evolutionary optimisation method, SRES (Stochastic Ranking Evolutionary Strategy) [Runarsson and Yao, 2000] an evolutionary optimisation method incorporating stochastic ranking based on bubble-sort and finally CMA-ES (Covariance Mix Adaptation-Evolution Strategy) [Hansen and Ostermeier, 1997] an evolutionary algorithm including an intermediate recombination step through derandomised covariance matrix adaptation. The benchmark used calls for the estimation of 36 kinetic parameters of a nonlinear biochemical dynamic model that describes the variation of metabolite concentrations over time. Moles et al. conclude that out of the aforementioned methods, Evolutionary Strategy-based algorithms demonstrate the best results with SRES being the best performing technique and uES following closely. Based on their findings, they make the informed assumption that evolutionary algorithms can be the most promising stochastic optimisation method especially when tackling large problems.

In their extensive 2003 article Blum et al. [Blum and Roli, 2003] made an interesting conceptual comparison of metaheuristics considered most important at the time, recognising the merit of applying combinatorial optimisation in the industrial world. They do not clarify though the selection criteria used to define which algorithms were considered important at the time. The authors examined EC (Evolutionary Computation), ACO, SA, TS (Tabu Search), ILS (Iterative Local Search), VNS (Variable Neighbourhood Search) and GRASP (Greedy Randomised Adaptive Search Procedure). The aforementioned algorithms employ the generic principles of intensification and diversification, expressed somewhat differently depending on the particular algorithmic strategy used. Intensification and diversification are the main drivers behind most metaheuristics, allowing the algorithms to focus on prospective good solutions and yet explore other candidate solutions too as a means of escaping local optima and converging closer to a global optimum. In this work, Blum et al. attempt to put intensification and diversification components in relation with each other, with algorithmic components being characterised by the functions they depend on e.g. objective function, randomisation, and how they affect the algorithms' search. They mention the possibility of identifying "subtasks" within an algorithm's search in their future work, where some algorithmic strategies exhibit better performance than others. The authors conclude that such knowledge will enable them to produce hybrid metaheuristic algorithms with a considerable performance improvement over pure metaheuristic methods, as seen in aspects of real life too.

Wetter et al. [Wetter and Wright, 2004] performed a comparison of different types of optimisation algorithms – deterministic and stochastic, gradient-based and non hill-climbing – for solving six different expressions of a building energy analysis problem. Namely the optimisation techniques used in this comparative study were: Hooke-Jeeves, two versions of Nelder-Mead simplex algorithm, Genetic Algorithm, Particle Swarm Optimisation, a hybrid particle swarm Hooke-Jeeves algorithm and the Armijo gradient algorithm. The authors do not mention the reasons behind choosing those particular algorithms, nevertheless their choice covers a wide range of techniques which leads us to believe that they attempted to cover as much of the optimisation spectrum as possible. Two main simulation models were used, one of them posing discontinuities in 2% of the fitness landscape while the other one has a continuous fitness landscape, allowing for a demonstration of each algorithm's restrictions based on the underlying strategy used. As expected, hill-climbing based algorithms such as Armijo or Nelder-Mead failed to converge close to or exactly at a global optimum, underlying the importance of knowing the fitness landscape upfront before utilising such techniques. Wetter et al. conclude that the best convergence was achieved using hybrid methods, namely the hybrid particle swarm and Hooke-Jeeves algorithm that allows for tackling discontinuities. They also add that whenever designers have enough margins to accept less accurate solutions, utilising a simple GA is a good tradeoff to make. Finally, they do not recommend the usage of Nelder-Mead or Armijo gradient algorithms due to their poor performance on those problems.

Another comparison of evolutionary-based optimisation algorithms with industrial practitioners in mind is published in 2005 by Elbeltagi et al. [Elbeltagi et al., 2005] without explicitly stating what was the aim of this comparison. GA, MA (Memetic Algorithm), PSO, ACO (the authors do not specify which particular ACO algorithm they use) and SFL (Shuffled Frog Leaping) were used to solve two continuous and

one discrete optimisation problem. F8 [19] also referred to as Griewank's function and F10[20] comprise the continuous cases solved, whereas a TCT (Time-Cost Trade-off) construction management problem including 18 activities is used as a discrete benchmark. The authors initially set up algorithm parameters as proposed in relevant literature. They then performed a small scale parameter sweep, monitoring how each parameter change affects every algorithm's performance before deciding the best setup per algorithm. Elbeltagi et al. conclude that PSO was the best performing algorithm overall, producing the best solution and maintaining a high success rate while being the second fastest algorithm in processing time.
Motivated by another engineering problem Panda et al. [Panda and Padhy, 2008] perform a comparison of GA and PSO, using a flexible AC transmission system controller design as their test case. The authors attribute their choice of algorithms on popularity and compare algorithmic performance to conclude on the effectiveness of each technique. Panda et al. infer that both algorithms can be successfully used for optimising a transmission system controller, with PSO demonstrating better performance (convergence achieved in fewer generations) despite the algorithm posing a higher computational time. Finally they underline the importance of choosing control parameters and objective function expression appropriately to ensure a successful optimisation run.
In 2011, we see another algorithmic comparison study for an engineering application. Namely, Shi et al. [Shi et al., 2011] compared EGA (Elitism Genetic Algorithm) with PSO for distributed power generation planning, without detailing the motivation behind this choice. However, in their article they did mention that in previous relevant studies, GA was used and EGA performed better in comparison. They concluded that EGA performed somewhat better than PSO. Optimising the location and unit capacity within distributed generation micro-grids comprises a multi-objective problem, accounting for the cost of investment and power losses. The objectives were to optimise the feeder network and the siting and sizing of distributed generation, based on wind and photovoltaic power generation whose capacity and position are uncertain.
One more comparison between GA and PSO for an engineering application was per-

[19] A scalable, non-linear objective function expressed as: $f(x_i|_{i=1,N}) = 1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N}(cos(x_i/\sqrt{i})$. The first term (sum of fractions) creates a parabolic shape whereas the latter (product of cosine) creates waves over the parabolic surface.

[20] A non-linear multi-variable function, F10 is expressed as: $f10(x,y) = (x^2+y^2)^{0.25} \cdot [sin^2(50(x^2+y^2)^{0.1}) + 1]$.

formed by Roberge et al. [Roberge et al., 2013] in 2013. The aim was to calculate feasible near-optimal trajectories for fixed-wing UAVs (Unmanned Aerial Vehicle) in a complicated 3D environment, while taking into consideration the dynamic characteristics of the aircraft. A multi-objective function was utilised, incorporating integral flight trajectory characteristics. Length, altitude, danger zones to be avoided, power required per trajectory, trajectories leading to collision, fuel required per trajectory and smooth circular trajectory arcs were all taken into consideration when generating a new path. Performing quantitative analysis, the authors found that GA produces superior UAV flight trajectories over PSO generated ones.
Hamdy et al. [Hamdy et al.] compared a number of stochastic optimisation algorithms for generating an optimal integrated design of a zero energy building. Seven multi-objective approximation algorithms were selected on the basis of popularity according to past literature on the topic. Namely, the algorithms used in this study were: a controlled non-dominated sorting Genetic Algorithm with a passive archive (an NSGA-II variant), a Multi-Objective Particle Swarm Optimisation (MOPSO), a two-phase optimisation GA (PR_GA), an epsilon dominance Multi-Objective Evolutionary Algorithm (evMOGA), an Elitist Non-dominated Sorting Evolution Strategy (ENSES), a multi-objective Differential Evolution algorithm (spMODE-II), and a Multi-Objective Dragonfly Algorithm (MODA). The objectives to be minimised were the building's primary energy consumption and life-cycle cost. The authors conclude that PR_GA consistently exhibits better performance, based on the combination of all the criteria used (execution time, convergence to benchmark optimal, diversity of solutions in Pareto set, number of Pareto optimal solutions, contribution of optimal solutions in Pareto front).
More recently, we find one more algorithmic comparison in the realm of applied sciences. Karagöz et al. [Karagöz and Yildiz, 2017] looked into optimising thin-walled tube structures, to improve car crash performance by simulating structure geometry and forming effects. The optimisation objectives comprised maximising energy absorption and minimising structure weight. Nine algorithms were compared, chosen on the basis of their recent appearance in relevant literature. In particular, PSO, Cuckoo search, Gravitational Search Algorithm (GSA), Hybrid Gravitational Search Algorithm-Nelder Mead (HGSANM), League Championship Algorithm (LCA), Firefly algorithm, Bat algorithm, Interior Search Algorithm (ISA) and Imperialist Competitive Algorithm (ICA) were used. The authors conclude that HGSANM generated the most lightweight and potent structure.
Overall, we find a relatively sparse yet interesting corpus of comparisons of algorithms' implementations, parametrised for real world applications. This can be

considered a positive step towards utilising an appropriate algorithm for each individual application.

2.6 Statistical analysis

In quantitative research, while examining two samples it is usual to perform a statistical hypothesis test (for a useful overview, see Appendix G) for assessing if populations' central tendencies are different. The usual null hypothesis H_0 is that the samples at hand share the same mean (for normally distributed data, otherwise median) $\mu_1 = \mu_2 = \mu_3 \cdots = \mu_n$ and the alternative hypothesis H_1 being that at least one mean (or median) is different. The variables to be analysed are typically classified as independent or dependent. An independent variable (also known as a predictor, explanatory, or exposure variable) is one that we think may cause a change in a dependent variable (also known as an outcome or response variable) [McDonald, 2014].

Usually tests assume normality of distribution, and examine sample means to reject or not the hypothesis that all samples share the same mean. A frequently used statistical analysis of two dependent/matched samples of one independent variable is the paired t-test for normally distributed samples. Two ordinal or interval samples can usually be statistically analysed using the non-parametric equivalent of the t-test, namely the Wilcoxon signed ranks test that does not assume normality of samples' distribution [National Institute of Standards and Technology, 2008; McDonald, 2014].

When examining more than two normally distributed samples, ANOVA (Analysis of Variance) [Daniel, 1990; Corder and Foreman, 2011] (also known as one-way ANOVA when examining samples of one independent variable) is used. If the samples examined are not normally distributed, non-parametric ANOVA tests are employed. One method frequently utilised to test two or more independent samples is the Kruskal-Wallis test [Kruskal and Wallis, 1952]. When multiple test attempts are applied (repeated measures), the consistency of the test outcome is usually detected using the Friedman test [Kanji, 2006; Marshall and Marquier, 2014; Lund Research Ltd., 2012].

The Friedman test is the non-parametric alternative to the one-way ANOVA with repeated measures, i.e. detecting differences among multiple test attempts. It is employed for testing for differences between samples when the measured dependent variable is ordinal, investigating the significance of the differences in response for K treatments applied to n subjects. Before using this test, the practitioner needs to check if four assumptions are met. Namely, a) one sample is measured on three or more different occasions, b) each sample is randomly selected from the population, c) our dependent variable should be measured at the ordinal or continuous level, d) samples do not need to be normally distributed [Lund Research Ltd., 2012]. Since the aforementioned assumptions are met, such a non-parametric one-criterion ANOVA test can allow us to infer whether there is an overall significant difference between at least one of the algorithms we employed and the rest of our algorithmic set.

Once a statistical analysis of more than two samples has been performed and a significant difference ($p < 0.05$) has been found, a post hoc analysis is required to indicate which samples are significantly different from one another. A wide range of multiple comparison tests are available, with a basic rule being that we should apply one test only in our experiment [Kim, 2015; Keppel and Wickens, 2004]. Nemenyi [Nemenyi, 1963] is a robust and less conservative post-hoc test used by default in the R software environment, following testing for statistical significance with the Friedman test [Marshall and Marquier, 2014]. It performs pairwise multiple comparisons, to indicate which pairs do present a significant difference in the metric we are measuring (here, schedule fitness).

2.7 Conclusions

In this chapter, we presented literature on stochastic algorithm applications, as well as relevant literature on the subject of optimal satellite mission operations scheduling and algorithmic parameter tuning. A set of algorithms was chosen from different categories such as natural selection or swarm intelligence, with or without memory etc. based on their popularity or absence in relevant literature on satellite mission operations scheduling. Interestingly enough, literature on mission operations scheduling dates back to the late 1980s [Dupnick and Wiggins, 1980a] despite the lack of ample processing power at the time. This fact underlines the need for optimisation and automation of satellite operations scheduling due to the increasingly complex mission requirements and satellite technology.

Diverse approaches have been used throughout the last 30 years, with stochastic algorithms being widely applied during the last two decades. It was clear from early on that each mission varies in operational and technological requirements, making it difficult to analytically describe a mission and apply exact optimisation methods. Therefore, approximation methods started being used not long after they emerged.

Literature shows that different problem sub-sets within the set of mission operations scheduling were attempted to be solved, such as long term astronomical observations, short term Earth observation, short and long term planetary observation on Mars, Venus and other celestial bodies.
Numerous research teams have applied stochastic algorithms in real problems in space mission operations with good levels of success, nevertheless the reasoning behind using one algorithm over another is not clear or strongly supported in the majority of related literature. We therefore find a gap in relevant literature, where practitioners only utilise a single algorithm for solving this important real world problem. Inspired by other multi-algorithm comparison studies, we hereby apply various algorithms for tackling the problem of satellite mission operations scheduling.
In the next chapter we will describe our satellite mission operations framework structure, designed to approximate a satellite mission including its orbital characteristics per time unit, its mission requirements over time, systems and payloads comprising our generic Earth Observation satellite, and how all the framework generated information is passed to a given optimisation algorithm. A space mission is dynamic, rendering the satellite mission operations scheduling problem challenging due to little a-priori knowledge available.

"Simplicity does not precede complexity, but follows it."

– Alan Perlis, Computer Scientist

3
Satellite Mission Operations Subsystem

In this chapter, we describe the motivation behind modelling satellites and how we approximated a typical small-scale satellite for experimentation.

3.1 The history of this project

This research project's initial aim was to perform robust design optimisation of space missions. The general idea pertained to viewing a satellite's subsystems and its mission aims as a complete entity whose design characteristics and mission operations can be optimised for multiple objectives. Namely, the physical design objective was to minimise the satellite's power consumption by minimising its individual subsystems' respective power requirements. This could be achieved by tuning all subsystem parameters that can be controlled by a human designer (design parameters). Environmental parameters could also be somewhat tuned, by changing for instance the satellite's orbit and attitude, respecting mission constraints.
At the same time, the mission would be considered optimal if the maximum of feasible mission operations aims was met. In order to accomplish this, mission operations scheduling had to be introduced as an optimisation objective, maximising the intersection between mission requirements and operations scheduling. To

achieve this, aside from modelling the satellite's hardware, orbit and environment, mission goals had to be set and translated into requirements. Required targets' visibility had to be propagated for the time span of interest, power generation had to be calculated for every time unit and power consumption was computed based on mission requirements.
In 2013, this research project's focus converged to automated optimal mission operations scheduling. Below, we present the software modules developed for the purposes of experimentation, attempting to maximise feasible mission operations by optimising a mission's operations scheduling component.

3.2 Modelling satellites

In general, an engineering system could be modelled in different ways, depending on information availability and system complexity. Space technology is usually closely tied to Defence applications, with stricter regulations constricting the amount of information and data that can be shared publicly.
Ideally, in order to achieve space mission operations optimality, a high fidelity satellite model should be developed to reflect the hardware, software and environment of the mission at hand. While propagating the mission's orbit is possible, it is not straightforward to faithfully model the hardware itself or the dynamic space environment. With the exception of CubeSats[1], satellites are bespoke, varying in volume, weight and technology. On top of that, data is securely held in company premises and are normally not shared with external parties. It would therefore be considered expensive and impractical to attempt to model every mission in detail, unless there is a strong scientific motive and respective funding to support such a project.
Naturally, one might ask if it would be possible to build a general satellite model that can represent and simulate an entire category of satellites. Again, such a project would lead to insurmountable obstacles, with the most prominent one being that satellite design is usually based on recommendations but not standards. Attempting to generalise a satellite category can lead to quite a wide range, thus a less useful categorisation. For instance, mini satellites weigh between 100kg and 500kg (classification is done on the basis of mass). Space systems engineering handbooks attempt to introduce a general picture of how satellite subsystems vary, based on publicly available past mission data sheets and specifications. Such references are used both for education purposes and preliminary design of future missions. However, relying

[1]CubeSats comprise a nanosatellite of standard weight and dimensions, called "U" for Unit. A 1U CubeSat measures 10cm x 10cm x 10cm and weighs less than 1.33kg. Larger CubeSats can be 1U, 2U, 3U or 6U, with their maximum weight being 1.33kg per U [NASA, 2017].

on such information acts as a high-level generalisation and approximation of a wide range of past missions. Therefore, attempting to develop a general satellite model with high fidelity is a very challenging task.

A sensible compromise is to combine space systems engineering experience, with publicly available information drawn from a real mission of interest. That way, an approximation of a satellite can be developed, being realistic enough for experimentation while remaining feasible to implement within a constrained time frame and budget. Space systems engineering information is employed to represent the core functionalities and expected environmental conditions of a type of satellite e.g. a scientific LEO mission. Then, some technical documentation and reports – which have recently started becoming available in space agencies' information repositories – can be used to complement space systems engineering references. Importantly, speaking with mission operations experts is integral for gaining a better understanding on particular missions and receive specific mission information.

This combined approach is the method we applied, in order to approximate an Earth Observation LEO mission that is inspired by the European PROBA-2 (Project for On-Board Autonomy-2) mission. At first, information was gathered from various space systems engineering sources and technical publications documents and a set of generic satellite subsystem approximation models were implemented. This piece of work contributed to publications [Vasile et al., 2011; Komninou et al., 2012a,b; Vasile et al., 2012] and subsequently was awarded an ESA ITI (Innovation Triangle Initiative)[2] grant. Appendix C contains the latest draft available to the author, of the relevant Stage A ESA ITI report. Then, we established communication with the PROBA-2 Science Operations team, who provided us with a set of high level requirements on science planning automation. Those requisites reflected the manual science planning performed at the Royal Observatory of Belgium, which is the Science Operations Centre (SOC) for the PROBA-2 mission. Appendix B gives an overview of PROBA-2 mission requirements, forming the basis of our experimental test cases. Had our collaboration with PROBA-2 experts continued, our software framework would be further developed to simulate the PROBA-2 satellite's functionality, mission and environment.

Utilising relevant literature and personal communications with space engineers, we developed modular software approximating main functionalities that characterise a small-scale Earth Observation satellite. The main systems included in this satellite model are a TTC (Tracking, Telemetry and Commanding) system, an ADS (Attitude Determination Subsystem), a CDH (Command and Data Handling) system,

[2]https://iti.esa.int/iti/index.jsp

an EPS (Electrical Power Subsystem), two cameras and three generic instruments. The model is deterministic, a reasonable assumption to make due to the reliability of satellites.

3.2.1 Modelling the satellite's orbit

When approximating a satellite mission, naturally the first step is to model its state for a time span of interest. Propagating its orbit allows us to infer integral mission characteristics that need to be used later on. Examples include targets' visibility or environmental parameters like incident solar radiation over time. Such attributes set the foundations on which the mission approximation will be based on. Calculating for instance a target's visibility opportunities and length per orbital pass, allows us to compute this task's contribution to the mission's power requirement for this time period. Likewise, computing when the satellite will pass above a magnetic field anomaly, will allow for including the appropriate timing and type of action to be taken in the mission's schedule.

To be able to deduce terrestrial targets' visibility windows for a continuously moving artificial satellite orbiting Earth, ESA recommended us to use the observation geometry system being developed by and used at NASA and ESA, named NAIF (Navigation and Ancillary Information Facility) SPICE[3]. SPICE calculates among others a satellite's state (its position and velocity) at any given moment, its field of view, position of the Sun and other stellar/planetary objects of interest.

At first SPICE reads a set of kernel files describing the satellite's orientation, ephemeris[4], each system's or payload's field of view and so on. Its output returns visibility occurrences (time and duration) of targets of interests for a given discrete time horizon with a chosen time resolution.

Each system accepts a number of inputs relevant to its functionality and outputs the main metric, power consumption. This output is used for calculating scheduling

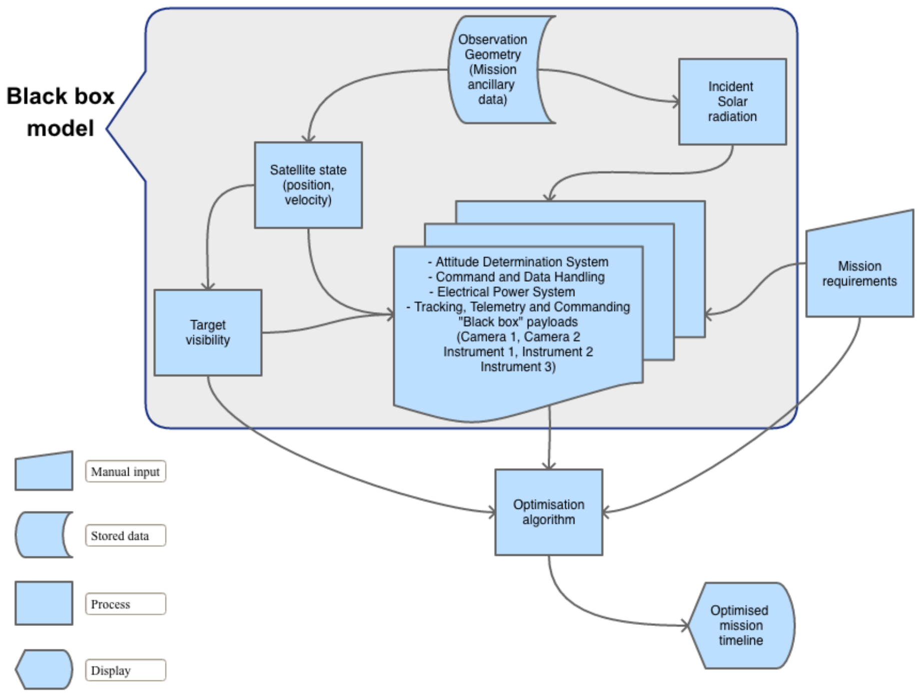
[3]An acronym of the five main types of information contained in a typical data file named "kernel", namely:
S- Spacecraft ephemeris, given as a function of time.
P- Planet, satellite, comet, or asteroid ephemerides, or more generally, location of any target body, given as a function of time.
I- Instrument description kernel, containing descriptive data peculiar to a particular scientific instrument, such as field-of-view size, shape and orientation parameters.
C- Pointing kernel, containing a transformation, traditionally called the "C-matrix" which provides time-tagged pointing (orientation) angles for a spacecraft bus or a spacecraft structure upon which science instruments are mounted. A C-kernel may also include angular rate data for that structure.
E- Events kernel, summarizing mission activities - both planned and unanticipated.

[4]An ephemeris gives the positions of naturally occurring astronomical objects as well as artificial satellites in a point of reference at a given time

capacity based on individual system and payload power consumptions. A set of predefined inputs was selected based on literature, leading to approximating small-scale Earth Observation satellite functionalities and thus individual systems' power consumption.

3.2.2 Overall framework structure

The system is modular as shown in Fig. 3.1, with components working in synergy to produce information about the mission status throughout the time frame in question. Mission information generated is then passed to the optimiser for schedule generation. The optimiser module searches for the best solution possible given the mission requirements and resource constraints set, using either a time-constrained or an iteration-constrained search. It returns a numerical result in the form of a binary matrix and a Gantt-like chart visual representation including power availability and tasks scheduled throughout every time step as seen in Figure 3.2.



Black box model
Observation Geometry (Mission ancillary data)
Incident Solar radiation
Satellite state (position, velocity)
Target visibility
- Attitude Determination System
- Command and Data Handling
- Electrical Power System
- Tracking, Telemetry and Commanding
"Black box" payloads (Camera 1, Camera 2 Instrument 1, Instrument 2 Instrument 3)
Mission requirements
Optimisation algorithm
Optimised mission timeline
Manual input
Stored data
Process
Display

Figure 3.1: Subsystem structure

In particular, this experimental framework comprises the following modules (author attribution and license given when appropriate):

- An Observation Geometry Subsystem for Planetary Science Missions is used for calculating the satellite orbit and lighting conditions[5]. Gregorian calendar dates are converted to MJD (Modified Julian Date)[6]

- A target visibility module is used for calculating passages through all targets of interest in mission[5]

- A set of satellite models approximating electric characteristics of the ADS, CDH, EPS, TTC[7]

- A set of payload *"black box"* models approximating satellite payloads' power consumption

- An optimisation algorithm[8] forming an optimised schedule based on mission requirements and constraints

- A mission timeline module outputting a binary matrix (1 = system/instrument ON, 0 = system/instrument OFF) and a Gantt-like chart visualisation of the resulting schedule

3.2.3 The optimisation problem

The combination of irregular satellite scientific operation requirements with the continual orbital motion of a satellite renders this optimisation problem dynamic. Scientific operation requirements change as new PI (Principle Investigator) requests are frequently submitted at varying times to the Mission Manager. Combined with the fact that power generation depends on orbital characteristics, satellite attitude, power generator health among others renders the overall problem description fluid, presenting a changing landscape per time unit.
As an example of how science operations comprise a dynamic problem instance, in Appendix D we present the first month[9] of MEX's real mission data following

[5]Author: NASA JPL under the NASA Open Source Agreement

[6]Author: LuxSpace sàrl under the BSD 2-Clause "Simplified" License. MJD is a more convenient form of the Julian date (JD), in which the zero point is 1858 November 17. The Modified Julian Date was introduced by Smithsonian Astrophysical Observatory in 1957. It is defined as JD - 2400000.5. The Julian Date represents an integer illustrating a whole solar day count starting from 12.00 Greenwich Mean Time. JD number 0 is assigned to 12.00 January 1st 4713 B.C Julian calendar (November 24th 4714 B.C Gregorian calendar) [United States Naval Observatory, 2008].

[7]TTC authored in part by A. Beaton, used with permission

[8]Author attribution given in Chapter 4

[9]During that month it was discovered that Mars' South Polar ice cap contains water. http://www.esa.int/Our_Activities/Space_Science/Mars_Express/Mars_Express_sees_its_first_water_scientific_results

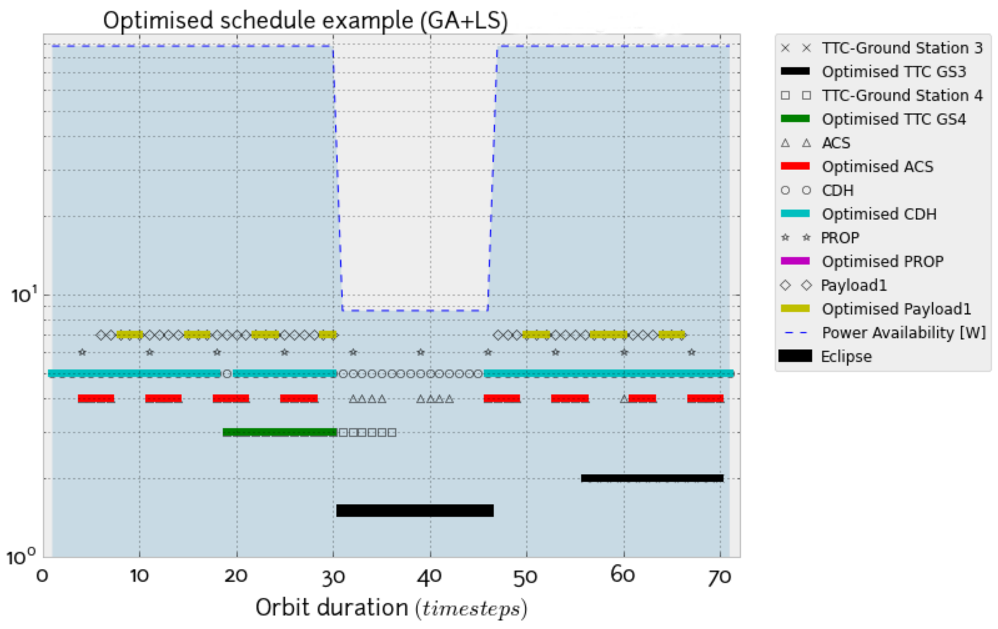


Figure 3.2: Example of an optimised 2.4 hour schedule Gantt-like chart

insertion into Martian orbit. The Earth and Sun distances with respect to the spacecraft change over time, just like Solar eclipse and Earth occultation[10] occurrences and durations do. The Earth's distance and occultation duration from Mars affect telecommunications' power consumption (reflected in our TTC model) and visibility windows that direct downlink operations' scheduling (reflected in our mission requirements module). Likewise Solar distance and eclipse durations affect the spacecraft's power generation[11] (reflected in our EPS model), usage and energy storage[12] profiles. Combined with irregularly dispersed scientific observations, it is evident that scheduling a scientific space mission is constrained and dynamic.

3.2.4 Geometry and target visibility

In order to calculate the mission's orbital characteristics per time-step (an example shown in Fig. 3.3), the SPICE [Acton, 1996] toolkit is used applying up-to-date

[10]To conceal (an apparently smaller celestial body) from view by passing or being in front of it e.g.
the Moon occults Mars during daylight on March 22. https://en.oxforddictionaries.com/definition/occult

[11]Electric power is the rate of transfer of electrical energy (J/s) within an electric circuit, per time unit.

[12]Battery energy capacity is the total energy available when the battery is discharged at a certain discharge current, calculated by multiplying the discharge power (W) by the discharge time (h). https://web.mit.edu/evt/info.html

NORAD (North American Aerospace Defense Command) elements[13]. SPICE comprises a set of individual software packages that read through mission kernels and generate satellite state vectors, target visibility occurrences, satellite orientation, satellite systems' and payloads' field of view among others. Such information allows mission planners to forecast the next available opportunities for scientific observation, data download and housekeeping operations among others.

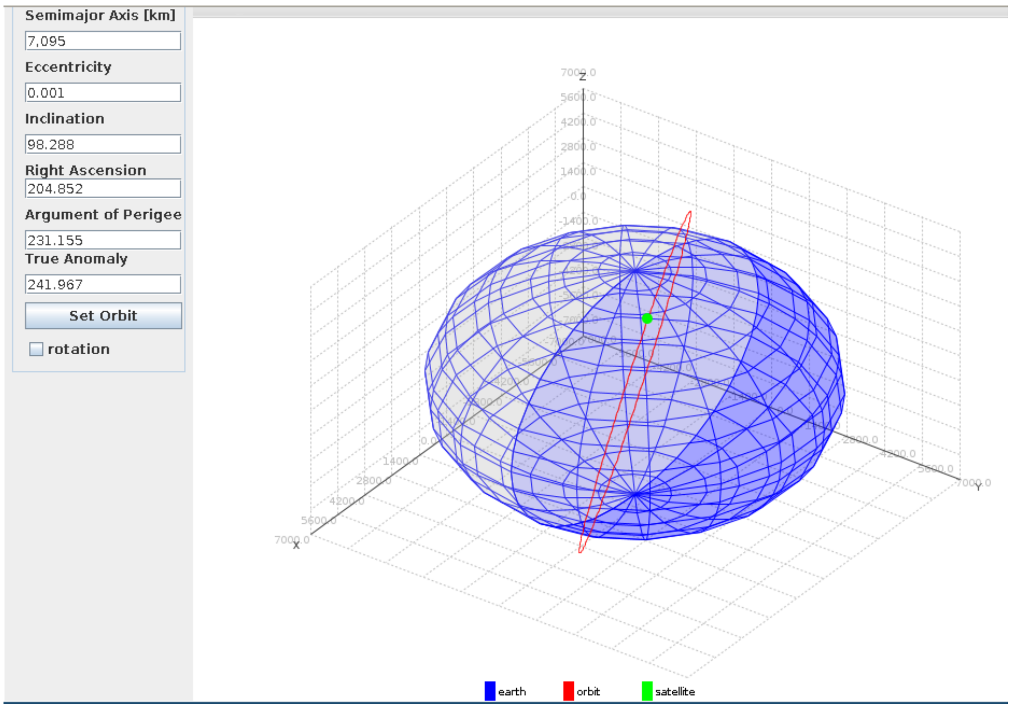

Semimajor Axis [km]: 7,095
Eccentricity: 0.001
Inclination: 98.288
Right Ascension: 204.852
Argument of Perigee: 231.155
True Anomaly: 241.967

Figure 3.3: An average PROBA-2 orbit, visualised using JAT Orbit Viewer [Gaylor et al., 2002]

SPICE normally accepts generic kernels[14] as inputs i.e. data files that apply to a wide range of missions. Mission-specific kernels[15] are also used for particular mission calculations. Generic kernels include:
The planetary, satellite, comet and asteroid ephemeris files produced by the NASA JPL (Jet Propulsion Laboratory) are known as SPKs (SPice Kernels). SPKs allow for computing the position of the satellite in question and the natural bodies surrounding it. It calculates ephemeris data (position and velocity per time instant)

[13]NORAD elements are found at: http://www.celestrak.com/NORAD/elements/engineering.txt

[14]Generic kernels can be downloaded from: https://naif.jpl.nasa.gov/naif/data_generic.html

[15]PROBA-2 kernels available at: http://proba2.sidc.be/aux/data/spice/kernels/

from the mission's ancillary files. This allows for the prediction of the satellite's position and velocity at any given time instant as well as the position of the celestial body it orbits e.g. the Earth and any other bodies of interest like the Sun.
The leapseconds kernel (LSK) is used to adjust mission time such that calculation time is close to the mean solar time[16]. Such occasional adjustments are necessary due to the Earth's rotation presenting irregularities creating a differential between atomic clock time and Earth rotation.
The planetary constants kernel (text PCK) contains constants used to model the orientation, size and shape of the Sun, planets, natural satellites, and selected comets and asteroids. The orientation models express the direction of the pole and location of the prime meridian of a body as a function of time. Likewise, the high-precision lunar orientation kernel (binary PCK) contains information to model the orientation, size and shape of the Moon.
The Earth station topocentric locations (SPK) and reference frames (FK) kernels include topographic information of every target of interest. Targets of interest are decided based on mission aims. Ground stations, morphological features and other targets of interest are defined based on their geodesic coordinates according to the World Geodetic Subsystem 84 [National Imagery and Mapping Agency, 2000], the same coordinate system used by the Global Positioning Subsystem. SPICE utilises that information to calculate the frequency and duration of each target's visibility. Detailed documentation and tutorials for using the SPICE toolkit are found at `https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/`.

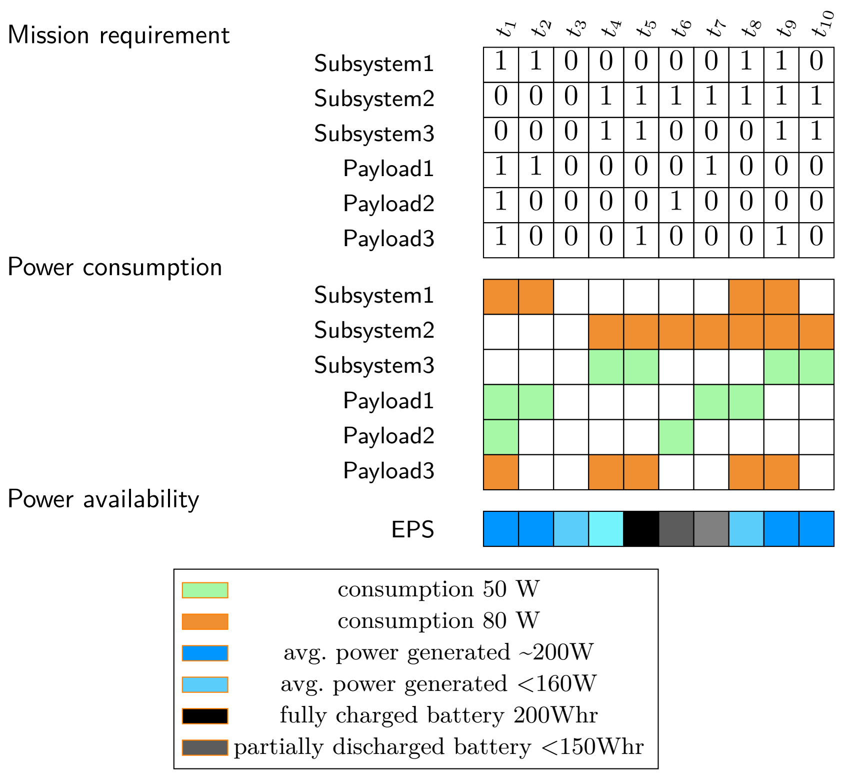
3.2.5 Representing mission requirements in our framework

Based on the mission scope, a set of requirements are decided and a number of constraints inevitably arise. The aim of the mission imposes specific requirements such as target observation, telecommunication through specific ground stations, environmental constraints like magnetic field anomalies and so on. Since satellites are highly resource- / mission- / environment-constrained systems by definition, limited resources such as power availability, battery energy storage and satellite orientation (directly influencing power generation) per time-step impose hard constraints over the mission.
Such constraints are incorporated into mission requirements, to ensure compliance with technical specifications. For instance, a requirement may request a measure-

[16]Solar time is the measure of time based on the Sun's position in the sky. Apparent solar time refers to sundial time while mean solar time refers to atomic clock time

ment such as multispectral Earth imaging to be periodically taken with specific cadence e.g. 2 minutes until further notice. This task should not be executed when the satellite passes through the SAA, due to increased ionising radiationinterfering with equipment and risking damage. Thus, during this passage, the original requirement like the one seen in Fig.3.4 is amended to abide by that technical constraint. In this Figure, we see a binary matrix representing the mission's requirement for 3 subsystems and 3 payloads, spanning 10 time steps ($t_1 \cdots t_{10}$). Here, 1 signifies that a subsystem or payload will execute some task during that particular time-step. 0 indicates that there is no need for the subsystem or instrument to operate during that time. We then see a power consumption matrix, which reflects each subsystem's or payload's power consumption whenever this is in operation. We then find the satellite's power availability, generated and stored by the EPS, which acts as our hard constraint. Soft constraints are considered those where a satellite subsystem/payload operates when not required, provided that the power constraint is not violated. Capturing requirements and constraints in such a way, allows for them to be easily used by our framework.
The framework accepts requirements as vectors containing a start time, duration and frequency in MJD format for each system or payload on-board. This forms a landscape encompassing all currently required tasks, shaping what an ideal mission operations schedule would look like. Each task comes with a vector of information regarding its power consumption per time unit of operation.

To present a realistic enough test case, we spoke to science mission operations experts from ESA, EUMETSAT, STFC and private companies, as well as satellite systems engineers from ADS, who kindly offered us their insights. Based on information from different missions (see Appendices B and C) and inspired by PROBA-2 [Science Center (P2SC), 2009; Kramer, 2002; Zender, 2012 - 15], we built a set of requirements in our formulation reflecting mission operations corresponding to a generic PROBA-2-like Earth Observation mission. It should be noted that time count starts from the exact time of the launch as per SPICE convention, thus all time intervals occur precisely after launch like SPICE documentation suggests. For instance, monthly calibration for 30 minutes occurs exactly a month after launch for 30 minutes, daily observations twice a day occur +11 hours after launch for their predefined duration and again +23 hours later. In particular the requirements this mission has to abide by can be found below.
Aside from including traditional satellite infrastructure, such as attitude determination, data processing, power handling, telecommunications, cameras etc. we have also included 4 generic models. These models represent experimental equipment for


Mission requirement
Power consumption
Power availability
EPS
consumption 50 W
consumption 80 W
avg. power generated ~200W
avg. power generated <160W
fully charged battery 200Whr
partially discharged battery <150Whr

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Subsystem1	1	1	0	0	0	0	0	1	1	0
Subsystem2	0	0	0	1	1	1	1	1	1	1
Subsystem3	0	0	0	1	1	0	0	0	1	1
Payload1	1	1	0	0	0	0	1	0	0	0
Payload2	1	0	0	0	0	1	0	0	0	0
Payload3	1	0	0	0	1	0	0	0	1	0

Subsystem1	🟧	🟧						🟧	🟧	
Subsystem2				🟧	🟧	🟧	🟧	🟧	🟧	🟧
Subsystem3				🟩	🟩				🟩	🟩
Payload1	🟩	🟩					🟩	🟩		
Payload2	🟩					🟩				
Payload3	🟧			🟧	🟧			🟧	🟧	

Figure 3.4: Generic mission requirements example, taken from our framework

engineering or scientific applications. Namely, Subsystem 1 can be an experimental propulsion or attitude control system like a solar sail, an inflatable structure, a new generation momentum wheel and so on. Instruments 1 - 3 can represent experiments such as radiation measurement, magnetosphere mapping, in-orbit astronomical observations among others.

Launch date:

- `2009 11 02 01:20:00.000 UTC`

Simulation start - end interval: (randomly chosen)

- Start: `2013 MAR 13 00:00:00.001 UTC`

- End: `2013 MAR 19 23:59:59.999 UTC`

Subsystem requirements:

ADC:

- Determine and control satellite attitude every 3.5 minutes In reality, this may vary irregularly depending on space weather and satellite pointing requirements, however this attitude variation has not been modelled.

CDH:

- Consider data handling always on. (in reality, engineers distinguish between idle, peak consumption and nominal. Due to the lack of concrete data, we didn't model this load variation.)

EPS:

- Generate power per time unit during daytime, provide energy during eclipse while decreasing stored battery energy according to power consumption per time-step.

TTC:

- Downlink data whenever visibility window for REDU[17] or SvalSat[18] ground station spans more than 9 minutes. Visibility windows are derived using SPICE for a chosen operations' timespan based on orbital characteristics and dates of interest.

Subsystem 1:

- Subsystem 1 operates 5 hours per day at the start of each day.

Payload requirements:

Camera 1:

- Image acquisition every 2minutes

- Camera reboot and software reload once a week for 20 minutes

- Camera calibration every two weeks for 50 minutes

[17]http://www.esa.int/Our_Activities/Operations/Estrack/Redu_station
[18]https://www.ksat.no/en/

- No calibration performed when flying over the SAA (South Atlantic Anomaly)[19]

Camera 1 performs periodic tasks usually, apart from the SAA constraint which occurs at a predictable but not always regular interval.

Camera 2:

- Camera calibration every two weeks for 540 minutes
- From November to February annually, acquire data once daily
- Unit calibration monthly for 270 minutes
- Occultation observation campaigns may be scheduled when predicted to occur (occurs irregularly)
- Solar flare observation campaigns may be scheduled at any point that the probability for a high-energy flare is higher than a predefined threshold. This is modelled as observations in our mission requirements, assuming knowledge of solar flare events.

Instrument 1:

- Experiment 1A observation twice a day for 480 minutes
- Experiment 1B observation twice a day for 360 minutes
- Experiment 1C observation twice a day for 480 minutes

Instrument 2:

- Experiment 2A observation three times a day for 360 minutes
- Experiment 2B observation twice a day for 300 minutes
- Experiment 2C observation three times a day for 240 minutes
- Experiment 2D observation twice a day for 540 minutes
- Experiment 2E observation twice a day for 540 minutes

[19]The South Atlantic Anomaly refers to a region over the South Atlantic ocean whereby the inner Van Allen belt extends as low as 200km above mean sea level, causing electric interference to subystems and sensors as well as affecting humans' optic nerves. Van Allen belts comprise energetic charged particles (the inner belt is dominated by protons, the outer one by electrons mainly) deriving from solar wind or cosmic radiation, held in place due to the Earth's magnetic field

Instrument 3:

- Experiment 3A observation twice a day for 270 minutes
- Experiment 3B observation three times a day for 300 minutes
- Experiment 3C observation twice a day for 420 minutes
- Experiment 3D observation twice a day for 420 minutes

The above requirements were derived from personal communication [Zender, 2012 - 15], multiple on site visits and a technical note we were entrusted with from PROBA-2 mission operations engineers (see Appendix B). Requirements were complemented with prior knowledge, experience and classified documentation from the ESA ESEO satellite that we had access to while taking part in the ESEO mission[20].
ADS and Subsystem 1 exhibit periodicity in operation, and CDH a constant behaviour in this model, as we did not have adequate data to approximate real-life behaviour. However ADS presents variable power consumption subject to orbital velocity and radius from the planet's centre of mass. EPS and TTC present variation in their operation that is related to both the satellite's orbital characteristics and mission requirements as seen in Tables F.3 and F.4.
Instruments 1 - 3 contain periodic tasks spanning longer periods of time due to the nature of each observation e.g. ADS-B[21] monitoring, AIS[22] monitoring, astronomical observations and others.
Further to observations, mission requirements contain housekeeping tasks too such as data downlink, attitude control etc. An important housekeeping task is to downlink as much stored observation data as possible to minimise information loss. Thus, every time the satellite passes above an available ground station, it downlinks data and receives new commands as long as the current visibility window spans longer than 9 minutes, to allow for meaningful communications. Visibility windows occur at different times depending on the orbit type, altitude, inclination and so on.
Determining and correcting the satellite attitude is also important, to enable instruments on-board to aim at or away from a particular target with good accuracy. For instance, aiming at a geological characteristic requires precision especially for narrow angle cameras. Likewise, aiming instruments away from the Sun to avoid

[20]ESA ESEO was launched on Dec 3rd 2018 https://directory.eoportal.org/web/eoportal/satellite-missions/e/eseo

[21]Automatic Dependent Surveillance - Broadcast (ADS-B) is a technology that allows aircraft to determine their position through satellite navigation and periodically transmit their bearing, altitude, identification code and others, to manage airspace efficiently

[22]Automatic Identification Subsystem (AIS) is a technology used to track and identify ships, supplementing marine radar information, to achieve collision avoidance in water transport

damaging their sensors also requires relative accuracy. Attitude determination is performed every 3.5 minutes, providing a good level of control over the satellite's orientation. Computer processing and power generation are always activated since they form the basis of all satellite functions.

In all three test cases we employed throuthout this thesis, this resulted in an array of n tasks and T timeslots, containing $n \cdot T$ binary values. The small test case spanning 2.4 hours per orbit, was inspired by the ESA technology demonstration mission LISA Pathfinder's orbit injection sequence[23] and ESA ESEO's Phase B1 hardware design. Each time-step was 120 seconds long, resulting in $T = 72$ time-steps with $n = 6$ tasks to schedule. The larger 1-day and 7-days long (STP) cases were inspired by the ESA technology demonstration mission PROBA-2's mission profile, i.e. its orbit, mission requirements and hardware. Both larger schedules utilised 30 second long time-steps, resulting in $T = 2880$ and $T = 20160$ time-steps respectively. The 1- and 7-days cases contained $n = 25$ tasks to schedule, with the 7-days case amounting to 504000 array elements $(25 \cdot 20160)$.

The problem consists of two parts: i) determining whether or not the mission requirements are feasible, given the aforementioned constraints, ii) if the mission requirements are infeasible, suggesting an alternative schedule that presents as much of the original mission requirements as possible.

3.2.6 Subsystem and payload models

High-level models were developed and used to approximate satellite functions and instrument representation to derive power consumption and availability per time instant. Each model accepts specific input values reflecting the satellite's technical specifications and environment, and all models output an estimate of their power consumption. The EPS outputs the power generated per time instant as well as the battery energy capacity.

Subsystem models[24] require two types of variables, *design* and *environmental* as seen in Appendix C. The former correspond to numerical values engineers can decide on (within feasible intervals given technical requirements and constraints) whereas the latter correspond to numerical values occurring from a satellite's orbit, ephemeris, planetary environment e.g. magnetic field and others.

Most subsystem models were based on system engineering tables authored by NASA, USAF (US Air Force) and industry engineers. Those tables were derived from pre-

[23]See eoPortal's LISA Pathfinder entry, Figure 18 and Appendix C

[24]Satellite subystem models' code is available at `https://www.bitbucket.org/satschedopt/satschedopt/Satsystems/`

vious missions' statistical information published in technical reports & engineering books, found in this thesis' bibliography, that are used in the space industry and academia. Performance range has been derived statistically from past missions of different sizes and scopes, representing a heterogeneous and realistic sample per subsystem equipment type. Such information is utilised by space agencies and companies during the first stages of mission design (Phase A & B) to perform approximate budget calculations of satellite subsystem characteristics.

Validation Validation of subsystems was performed predominantly through real mission data and empirically, ensuring that inputs and outputs correspond to realistic approximations of ESA LISA Pathfinder[25]. On the one hand tabular data were used from engineering literature, combined with ESA ESEO design data and LISA Pathfinder publications and ESA ESEO design documentation. On the other hand experience was drawn from engineering experts at ESA in close collaboration with Prof Massimiliano Vasile and Dr Edmondo Minisci of Mechanical & Aerospace Department at Strathclyde who closely monitored, steered the development of those subsystem models and validated their output.
Tabular data were linearly spaced within each metric's specified range and linear interpolation was used to calculate a subsystem's power consumption (or generation in the EPS case) corresponding to design inputs specified by the user.
As mentioned above, the subsystems included in this generic Earth Observation satellite are the following:

Attitude Determination The ADS was modelled mainly based on bibliographical system engineering data included in Table 3.1. A typical ADS comprises a number of sensors contributing to attitude determination, with calculation capability taking place either in the satellite's main processing unit or a separate ADS processing unit. If the subsystem contains attitude control too, passive or active control modules such as magnetotorquers or thrusters are included in budget calculations.

[25] http://sci.esa.int/lisa-pathfinder/

Table 3.1: ADS: Estimating power in early phases of a project [Wertz and Larson, 1999]

Sensor	**Typical Performance Range**	**Mass (kg)**	**Power (W)**
Internal Measurement Unit	Gyro drift rate = $0.003 deg/hr$ to $1 deg/hr$ Linearity = 1 to $5e^{-6}\ g/g^2$ over $20 - 60g$	1 to 15	10 to 200
Sun Sensors	Accuracy = $0.005 deg$ to $3 deg$	0.1 to 2	0 to 3
Star Sensors	Attitude accuracy = 1 *arc sec* to 1 *arc min* $0.0003 deg$ to $0.01 deg$	2 to 5	5 to 20
Horizon Sensors	Attitude accuracy $0.1 deg$ to $1 deg$	1 to 4	5 to 10
Magnetometer	Attitude accuracy = $0.5 deg$ to $3 deg$	0.3 to 1.2	< 1

The ADS we used contains two sensors of each type mentioned in Table 3.1, with performance characteristics stated in Table F.1. Quantities depending on orbit or other environmental characteristics, are denoted as variables. For attitude determination, the subsystem contains a pair of Star sensors, Horizon sensors, Magnetometers, Gyroscopes and Accelerometers.

Command and Data Handling The CDH design is based on system engineering data included in Table 3.2.

Table 3.2: Early phase estimation of power [Brown, 2002]

Equipment	**Mass (kg)**	**Power (W)**
Equipment relatively independent of data rates Telecommunication interface Data-flow control Signal conditioning Command processing Spacecraft clock	30	20
Computer	2	10
Science data processor	15	2 + 1W/instrument
Engineering data processor	10	5
Data storage	0.25kg/Gbit	1W/Gbit

To ensure robustness, our model's CDH subsystem comprises two processing units corresponding to a total of two processors, two telecommunication interfaces, data flow control units, signal conditioning modules and so on. Two data recorder units are included in the subsystem, with a total storage of 12 *Gbits*. The setup parameters are appended in Table F.2.

Electrical Power Subsystem Information on EPS design is more widely available [Patel, 2004; Hyder, 2000], allowing us to design the model of this subsystem using both statistical data and analytical expressions. A typical Earth Observation EPS comprises a power generator and an energy storage unit, along with some basic robust logic unit to ensure efficient power generation and safe energy storage and distribution.

Power generation: We chose solar arrays for our power generator, which is the standard choice for all inner solar system missions. At first the average total power requirement needs to be calculated using Eq. 3.1.

$$P_{total} = \frac{\left(\frac{P_e\ t_e}{X_e}\right) + \left(\frac{P_d\ t_d}{X_d}\right)}{t_d} [W] \tag{3.1}$$

where P_e, t_e and X_e refer to the average power demand during eclipse, the average duration of each eclipse and the average energy transmission efficiency during an eclipse respectively. Likewise, P_d, t_d and X_d refer to the daylight counterparts

of the above.
Once the P_{total} has been calculated, the designer has to choose the type of solar cells to utilise for the satellite power generator. Solar cell efficiency depends on solar cell material chemistry, which also affects characteristics like voltage and current output as well as degradation over time. Figure A.1 in Appendix A depicts the state of the art solar cell technology. For our subsystem we chose six commonly used solar cell types found below. In our model, solar cell characteristics' values are linearly interpolated.

Table 3.3: Solar cell types and characteristics [Wertz and Larson, 1999]

	CdTe	**p c-Si**	**u c-Si**	**3j GaAs**	**conc. 3j GaAs**	**multij**
Voltage (V)	0.740	0.530	0.430	2.290	2.680	2.260
Current (A)	0.028	8.080	0.774	0.525	6.760	1.750
Efficiency (%)	16.5	20.3	24.7	29.9	38.5	40.7
Degradation (%/y)	1.0	3.7	3.7	0.5	0.5	0.5

Once a solar cell type is chosen, the array size is calculated based on cell degradation over time. A satellite should be able to serve its power requirements even at the end of its useful life. Thus power generator sizing is calculated based on power demand at the end of the mission based on Eq. 3.2

$$P_{eol} = P_{bol}\ L_d[W] \quad \text{given:} \quad L_d = (1 - D)^{T_m} \tag{3.2}$$

where P_{eol} and P_{bol} is the mission power requirement at the mission's end of life and beginning of life respectively. L_d represents the degradation rate, with D being the inherent degradation percentage and T_M refers to the primary mission lifetime in years.
Once P_{eol} is calculated, it is straightforward to roughly size the power generator. The bus voltage is divided by the chosen solar cell type voltage to derive the length of each array string. Likewise bus current is divided by the respective current output to calculate the required number of strings. Multiplying the array string length with the number of strings allows us to calculate the expected solar array power output.
Battery storage: Battery sizing depends on P_e (power consumption during eclipse) and t_e (time of eclipse) which allow the designer to derive the satellite energy storage requirement. The product of P_e and t_e signifies the minimum battery energy E_{bat} and E_{bat}/η_{bat} (with η_{bat} being the battery efficiency) returns the required battery pack energy.

Table 3.4: Battery cell types and characteristics [Patel, 2004]

	NiCd	NiH_2	**NiMH**	$LiFePO_4$	**LiIon**	**LiPoly**
Energy density (Wh/kg)	60	75	80	110	150	200
Voltage (V)	1.25	1.25	1.25	3.3	3.6	3.5
Efficiency $\eta_{bat} = \eta_{out}/\eta_{in}$ **(%)**	0.85	0.86	0.87	0.90	0.95	0.998

Our EPS model receives the t_e and t_d eclipse and daylight durations per orbit from the SPICE geometry system mentioned above. Individual subsystem and payload power consumptions per mission day are summed based on whether they take place during daylight P_d or eclipse P_e. Daylight energy efficiency transfer X_d was assumed to be 0.85 while the efficiency during eclipse X_e was set to 0.65. The satellite's main bus runs at 28V, driving the number of solar cell strings in series. Likewise, the maximum bus current rating drives the number of parallel strings. The power generator comprised triple junction Gallium-Arsenide ($3j$ $GaAs$) solar cells, the mission duration T_M was set to 5 years and the energy storage comprised Lithium Iron Phosphate ($LiFePO_4$) cells. The tabulated set of parameters used in our test case can be found at Table F.3.

Tracking, Telemetry and Commanding TTC subsystems can vary depending on a satellite's communication needs. Normally a TTC comprises a *link* module representing telecommunications elements such as signal modulation, antenna type etc. as well as an *amplifier* module representing the power amplification infrastructure to boost signal power over background noise. Systems engineering books like [Wertz and Larson, 1999] provide informative communications architecture facts.
To calculate a TTC subsystem's power consumption we need to work somewhat backwards. First the downlink frequency needs to be decided based on a number of factors that are beyond the scope of this manuscript. Based on the frequency and other parameters like the maximum distance from a target, the required transmission power and bit rate, the designer can calculate the transmitter gain and consequently the RF power output described in Eq. 3.4. Table 3.5 contains a set of parameters calculated for VORSAT, a European student nanosatellite project [Rodrigues Capela, 2012].
The main metric that needs to be calculated is the transmitter gain G_t according to Eq. 3.3 [Roddy, 2006], depending on the subsystem's antenna characteristics. From that gain we can derive the RF power output in Watts, to size the TTC subsys-

tem [Wertz and Larson, 1999].

Table 3.5: Link budget for downlink at 2.45GHz [Rodrigues Capela, 2012]

Feature	**Value**
Maximum distance	1160 (km)
Transmission power	25 (dBmW)
Transmission loss	1 (dB)
Transmission antenna gain	4.5 (dB)
Equivalent Isotropically Radiated Power (EIRP)	30.5 (dBmW)
Free space loss	161.47 (dB)
Atmospheric absorption	1 (dB)
Polarisation loss	3 (dB)
Antenna misalignment loss	1 (dB)
Propagation loss	166.47 (dB)
Satellite antenna gain	35 (dB)
Subsystem noise temperature	110.11 (K)
Figure of merit	14.59 (dB/K)
Boltzmann constant	-228.6 (dB/K/Hz
Received power (P_r) to noise density (N_0) (P_r/N_0)	77.22 (dBHz)
Bit rate	9600 (bit/s)
Received energy-per-bit (E_b) to noise density (E_b/N_0)	37.4 (dB
Bit Error Rate	10^{-5}
E_b/N_0 @ 10^{-5}	9.6 (dB)
Downlink margin	27.8 (dB)

In particular, the metrics required for calculating the transmitter gain are the antenna diameter d_t, the antenna efficiency η_t accounting for imperfections or losses and the communications' band wavelength λ.

$$G_t = 10log_{10}\left(\eta_t \left(\frac{\pi \ d_t}{\lambda}\right)^2\right) [dB] \tag{3.3}$$

Using Eq. 3.3 and Table 3.5 RF power output P_{RF} is calculated as

$$P_{RF} = 10^{(EIRP-G_t/10)}[W] \tag{3.4}$$

Based on the above, we can calculate the transmitter power consumption from Figure 3.5 allowing us to approximately model the mission TTC.

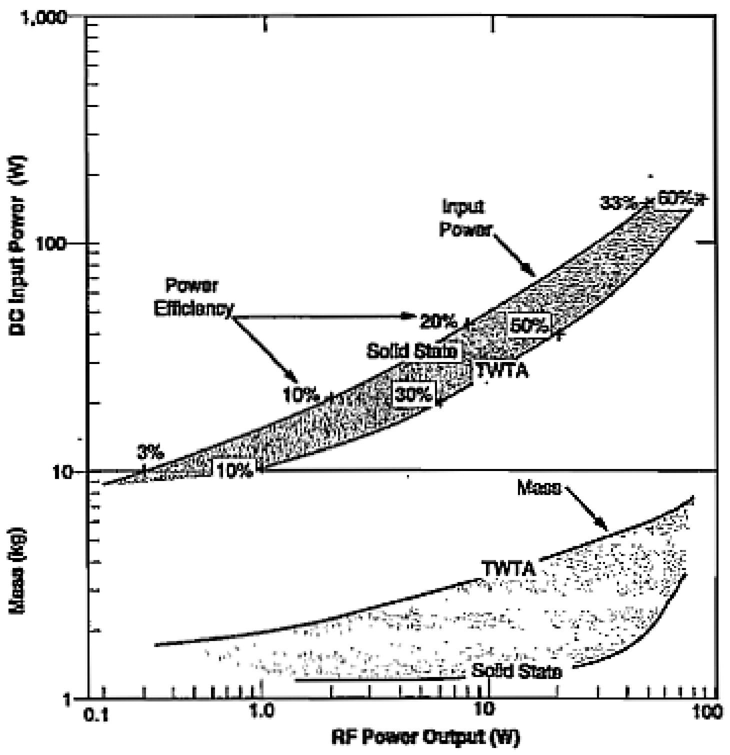


Figure 3.5: Satellite Transmitter Power and Mass vs RF Power Output [Wertz and Larson, 1999]

The TTC model we used is loosely based on nanosatellite Low Earth Orbit missions. The downlink transmission frequency is set at $2.45GHz$ (UHF band), with a corresponding wavelength λ of approximately $122mm$. We utilise a parabolic high gain antenna of $1.5m$ in diameter d_t and an efficiency η_t of 0.6. Table 3.5 contains details such as the subsystem's EIRP (Equivalent Isotropically Radiated Power) among others and the total subsystem power consumption is calculated from Fig. 3.5. Our TTC model's inputs can be found at Table F.4.

3.2.7 Optimisation module

Approximate methods are used when computational problems are too hard to solve exactly. Approximation algorithms have been extensively used on NP-hard [Garey and Johnson, 1979] problems since the early days of their development, where exact algorithms' time complexity scales exponentially, rendering even smaller instances of hard problems challenging.
In order to optimise a given problem, one needs to form an illustrative representation that reflects the problem's nature, its requirements and constraints. There are

different ways of elucidating a problem, depending on what we want to achieve. For instance, finer detail is required for studying the response per millisecond of a fast transient phenomenon. Furthermore, depending on the problem at hand information can be represented as integers, real numbers or binary values.

During this modelling effort, it was decided that a binary matrix representation can encapsulate satellite mission operations scheduling, similar to the way space operations engineers apply it in practice. This design choice simplifies information representation in our problem, maintaining a satisfactory time resolution. Since the satellite mission operations scheduling problem varies per mission and is not formalised, lending itself to a wide range of interpretations, it is not straightforward to represent it analytically. Each task activation state is therefore represented by a boolean value, true equals activation and false indicates no task execution.

The input to this module comprises an array describing the mission requirement itself (see Section 3.2.5), the mission requirement's power consumption and the satellite's power availability per time-step. The module outputs a feasible solution, expressed as a binary matrix of size $n \cdot T$ corresponding to all n instruments' and subsystems' activation states for all time-steps T. The feasible solution represents the modification of the mission requirements to achieve feasibility.

Algorithms included in the optimisation module, attempt to maximise the number of scientific and control activities to be scheduled per time unit. Meeting the exact mission requirements and thus scheduling all requested tasks would be ideal. Practically though, this is not always possible. Usually it is considerably challenging to objectively decide which instrument can be considered more important. Decisions may be based on subjective arguments, vague mission scope definitions, simple linguistic misinterpretations in international settings etc.[26]. This is all ignored here.

We employed the metric of Hamming distance for calculating generated schedules' fitness. In essence, the aim is to maximise similarity between the mission requirement array and the algorithmically generated operations schedule array, minimising their Hamming distance.

Search space

Based on the number of subsystems and instrumentation included in the satellite a two dimensional binary search space of size 2^{nT} is generated, with n representing

[26]Conversing with science operations experts, they unanimously agreed that task prioritisation discussions rarely reach a consensus, often causing tension between and among teams. Furthermore, when experts were asked to estimate mission aims for their mission to be considered a success, they could not specifically quantify aims. Thus, it is important to bring neutrality to this process and maximise scientific return, by considering all activities equally important.

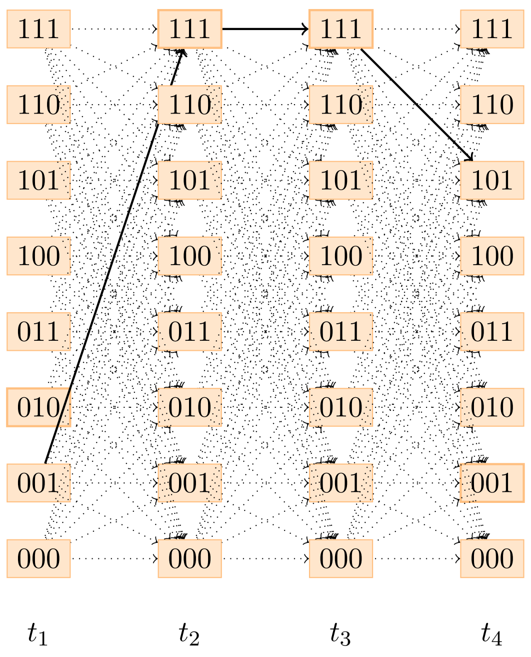


Figure 3.6: Search space structure for 3 tasks. Each column contains 2^n state combinations per time-step t. The resulting final solution vector is "001|111|111|101".

the tasks to be scheduled and T the total number of time-steps characterising the schedule span. The problem can be visualised using a trellis diagram[27] [MacKay, 2003] with columns signifying discrete time-steps and rows containing all possible perturbations of tasks' activation states. Fig. 3.6 shows an example of the search space structure for three tasks to be scheduled for four time-steps.

The cardinality of the schedule's total time-steps T depends on the schedule's temporal duration strategy (short/medium/long term) and required level of control (fine-grain vs coarse-grain schedule). When using SPICE satellite operations, experts need to define the time-step duration e.g. 30 seconds, which is used for generating the mission operations search space. Care should therefore be taken towards maintaining a balance of good orbital approximation and satisfactory mission operations' time resolution.

[27]Such diagrams were used in the trellis modulation scheme, a method to efficiently transmit information over band-limited channels like copper telephone lines.

3.3 Conclusions

In this chapter, we described the framework we developed to model a satellite's environment, its infrastructure including scientific payload and its mission requirements, feeding all relevant information to an optimisation module to generate a mission operations schedule. To achieve this we utilised a systems engineering approach, treating each subsystem as a generic unit consisting of key modules. For instance a telecommunications subsystem comprises an antenna, a link module and an amplification module. Subsystems accept inputs that vary with orbital position, lighting conditions and other mission parameters, outputting variable power requirements. Mission requirements were translated into a binary matrix, which in turn was passed on to our satellite framework to generate a mission profile. A set of algorithms were then used to optimise the mission operations schedule.
The scenario used was representative of an Earth Observation mission, with the modelled generic satellite incorporating five basic subsystems and five scientific instruments. Initially the satellite's orbital characteristics were derived through an observation geometry system. The satellite's infrastructure was developed, comprising an attitude determination system (ADS), a command and data handling (CDH) subsystem, an electrical power system (EPS), a tracking telemetry and commanding (TTC) subsystem, a generic subsystem and five observation instruments including two cameras and three generic instruments. Statistical information and some analytical expressions were used to model satellite infrastructure. Instrumentation was modelled as a set of black boxes, since instrument characteristics vary significantly thus being challenging to categorise. Validation for the models was performed empirically using a combination of real satellite data as well as previous experience from past ESA missions and experts on the field.
In the next chapter we will introduce a more formal description of the problem at hand and solve a small scale instance inspired by ESA ESEO and ESA LISA Pathfinder, to compare algorithmic performance on an even basis for a variety of optimisation algorithms. Do all algorithms solve the same problem equally well?

"It is better to solve one problem five different ways, than to solve five problems one way."

– George Pólya, Mathematician

4 Comparing algorithms for satellite operations scheduling

Approximation algorithms are broadly used for solving real-life complex problems, potentially growing exponentially. The real-life based space engineering problem at hand, is expressed as a binary array where 0 = instrument or subsystem OFF, 1 = instrument or subsystem ON. Considering that we have a total of n tasks to schedule, for a span of T time-steps, the search space grows at a rate of 2^{nT} with the exponent being the product of two likely large values. The rapid growth of this engineering problem size, renders it a good candidate on which to apply approximation methods, assuming that no sufficiently good polynomial time algorithm can be found to solve it. To test whether a polynomial time algorithm would be able to tackle this problem satisfactorily, we applied a simple greedy algorithm (see Section 4.1.10).
More formally, the problem can be stated as:

> *"Form a candidate solution array V – aiming at maximising its similarity with mission requirements array S – that represents the activation states of n tasks for a span of T time-steps. Set $V_{i,t} = 1$ to activate task $i \in \{1, \cdots, n\}$ during time-step $t \in \{1, \cdots, T\}$, else $V_{i,t} = 0$. For all time-steps the available power A_t must be respected."*

In other words,

$$\text{Form } V, \text{ with } V_{i,t} \; = \begin{cases} 1 & \text{if task } i \text{ is selected at time-step t} \\ 0 & \text{otherwise} \end{cases}$$

Such that: *H(V, S)* is minimised

$$\tag{4.1}$$

where, $H(V,\ S)$ signifies the Hamming distance between arrays V and S i.e. the number of bit substitutions required for V to become identical with S.
Every candidate vector V should satisfy the constraint of power availability per time-step A_t, i.e.

$$\Big(\sum_{i=1}^{n} C_{i,t} \cdot V_{i,t}\Big) \;\leq A_t, \forall t \in \{1, \cdots, T\} \tag{4.2}$$

where $C_{i,t}$ is the power consumption of task i at time t and $V_{i,t}$ the task i's activation status, at time t. The objective is to maximise the amount of required activated tasks per time-step, resulting in the highest *useful* operation time possible per instrument or subsystem (for a fitness calculation example, see Figure 4.1). By useful, we define the period where it is both expected and possible for a certain task to be performed, in accordance with mission requirements S. As mentioned above, the hard *constraint* is to remain within the power availability per time-step A_t.
In this work, we employed three tactics for comparison: i) a greedy approach, ii) a sequential (optimising each time-step independently) strategy and iii) a full length schedule optimisation one. Approaching this scheduling problem through the greedy lens, we stumble across the issue of premature allocation. That is, since a greedy algorithm is short-sighted, it's very probable that it will allocate too many resources too soon. And in a system with restricted resources, such as a satellite's battery employed during eclipse times, this can have negative consequences towards seeing through the mission.
Similarly, when examining the schedule timeline on a step by step basis, optimising each step without taking the bigger picture into account leads to comparable issues attributed to a myopic view of the problem. While a solution may look fit enough on a per time-step basis, it could e.g. deplete the satellite's battery before the expected end of an eclipse period.
On the other hand, attempting to optimise the full schedule and keeping a more long-sighted view of the problem allows for a more holistic viewpoint, that might result in a more well balanced solution. This approach though comes at a cost, namely a vast search space needs to be tackled so that a globally near-optimal solu-

tion can be found.

Mission requirement

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
Subsystem1	1	1	0	0	0	0	0	1	1	0
Subsystem2	0	0	0	1	1	1	1	1	1	1
Subsystem3	0	0	0	1	1	0	0	0	1	1
Payload1	1	1	0	0	0	0	1	0	0	0
Payload2	1	0	0	0	0	1	0	0	0	0
Payload3	1	0	0	0	1	0	0	0	1	0

Current solution

Subsystem1	1	1	0	0	0	0	0	1	1	0
Subsystem2	0	0	0	1	1	**0**	1	1	1	1
Subsystem3	0	0	0	1	1	0	0	0	**0**	1
Payload1	**0**	1	0	0	0	0	1	0	0	0
Payload2	1	0	**1**	0	0	1	0	0	0	0
Payload3	1	0	**1**	0	1	0	0	0	**0**	0

Fitness

$H(V_t, S_t)$	1	0	2	0	0	1	0	0	2	0
$f(V_t)$	5	6	4	6	6	5	6	6	4	6

$$\boxed{f(V) = nT - H(V,S)}$$

Figure 4.1: Fitness calculation example

Given that utilising approximation algorithms to solve the aforementioned problem is a reasonable solution, a pertinent question arises: i) Do all optimisation algorithms perform equally well, independent of the given problem expression and size? ii) How can a practitioner informedly choose one algorithm over others?

4.1 Observing algorithmic behaviour

To address the question above, practitioners need to quantify algorithmic potential, comparing how different algorithms perform when tackling the same problem, and which algorithm presents fitter solutions, given the present problem expression. Before applying an algorithm on a specific problem, we need to set it up so that it performs to its full capacity. One way of achieving that is by performing sensitivity analysis.

4.1.1 One-step-At-a-Time sampling design

A general term used to describe the process of designing computational experiments is “sensitivity analysis”, or “factor screening”. When one needs to perform sensitivity analysis of an expensive simulation, applying OAT (One-step-At-a-Time) sampling design can be useful [Santner et al., 2003]. OAT, also known as Morris method, can be used as a preliminary experiment on systems with multiple input variables or parameters in order to analyse the influence of each input parameter on the output. The aim is to understand which input variables have a substantial influence on the output, while keeping the number of runs proportional (as opposed to exponential) to the number of inputs [Morris, 1991].
The basic principle behind this experimental design method is to observe elementary effects, i.e. those changes in an output attributed to changes in a particular input. Assuming that we have input values $x_j,\ j = \{1, 2, \cdots, k\}$ affecting output y, an elementary effect would be calculated as: randomly choose a value for x_j and check the response of y, choose a value of $x_j + \Delta$ and recalculate y. This comprises one elementary effect.

4.1.2 Problem instance employed for algorithmic parameter tuning

To analyse our algorithms' control parameters response, we applied the OAT method using a single instance: an 89-minute long LEO revolution of Phobos-Grunt containing 4 tasks and 44 time-steps. Phobos-Grunt was a Russian mission designed to land on the Martian moon Phobos and return a sample to Earth. This instance was formed by the author of this thesis, by first shaping the spacecraft's early orbit using its TLE[1] (Two-Line Element set). Then, four Estrack [Estrack ground stations, 2017] ground stations were chosen (Malindi (Kenya), Raisting (Germany), Perth (Australia), Kourou (French Guyana)) that are generally used by NASA and ESA for interplanetary missions. Also, four subsystems were included to approximate the core of the mission. Namely, ADS, PROP, TTC and one payload. A random date after the mission's launch date was chosen, and a single orbit was propagated forward. Depending on the orbit start and stop times, ground stations' visibility was calculated to indicate telecommunication windows for the TTC. Similarly, daylight, night time duration and solar incidence angles were calculated for inferring the

[1] Two Line Element Set (TLE):
PHOBOS-GRUNT
1 37872U 11065A 12015.69171186 .10804850 13050-4 36619-4 0 9996
2 37872 051.4132 350.1235 0009680 342.5577 017.5128 16.57963948 10968
Reference: https://www.n2yo.com/satellite/?s=37872

EPS' power generation. ADC and PROP were engaged together every 7 time-steps throughout the simulation, starting from the 5th time-step of the initial orbit. The payload was employed for a random number of time-steps between 1 and 15, being turned off at the last time-step before eclipse.

The use of a single test instance for algorithm parameter tuning is attributed to the scarcity of available information on the topic. While one can access useful high-level space systems engineering books and some relevant public technical reports, the information included in them is usually too generic to constitute the basis of real-life test cases. Research conducted by space agencies or private space companies may occasionally use real space data, however access to that data is restricted for internal use only given appropriate security clearance. We can attest to this, since multiple visits to the European Space Agency and private space companies did not allow us to access mission data and detailed information, thus we had to be pragmatic.

Usually, a variety of instances from the same problem type are applied for tuning algorithmic parameters. That way, it is ensured that the algorithm is not only tuned for one specific instance of the problem at hand. Thus we ought to warn practitioners of the risk involved in utilising a single test instance, and encourage the space operations community to publicly share data for motivating further relevant research with improved outcomes.

4.1.3 How we tuned our algorithmic set

Generally, OAT can be applied by keeping an algorithm's parameters fixed except from the first parameter of choice, vary that parameter and notice the influence it has on the algorithm's output. Then, utilising the input value that returned the best output, the second parameter will be varied to infer its influence to the output. Continuing this process until all parameters have been analysed one at a time, will allow the practitioner to deduce a set of parameters that will allow the algorithm to perform better.

A sample of 10 runs was collected per set of parameter settings for each algorithm, calculating the median schedule fitness $\overline{f}$. We then plotted the median fitness per control parameter level e.g. $\overline{f}(\mathrm{I}_{max})$, $\mathrm{I}_{max} = \{25, 50, 75, \cdots, 750\}$ to infer the best performing parameter value. A boxplot per parameter was also generated, for an intuitive understanding of that parameter's influence to the schedule fitness.

We applied OAT for tuning our set of algorithms as follows. Let's take ACS (section 4.1.6) as an example: the parameters examined were the iterations number I_{max}, the number of ants NP, the weight β of heuristic information η, the probability q_0 of exploring the search space over exploiting known search space information, the initial pheromone level τ_0 and the global pheromone evaporation constant ρ. Initially varying I_{max}, and keeping all other parameters fixed, we set $I_{max} = \{25, 50, 75, \cdots, 750\}$ with a step of 25. The rest of the tuning parameters were set to an arbitrary but sensible level depending on their utility (e.g. ρ is a percentage ranging [0, 1] but normally is kept low to avoid evaporating pheromone rapidly, β is an exponent acting as a weight thus it's usually a small positive integer value etc.). Plotting $\overline{f}(I_{max})$ helped us pick the best iterations value. Updating the algorithm parameter set accordingly, we then varied NP, using a constant as a binding between NP and I_{max} to keep the computational effort constant for fairness of comparison. After plotting $\overline{f}(NP)$ and updating NP and I_{max} accordingly, we varied β. Once the highest $\overline{f}(\beta)$ was found, we proceeded to varying q_0. Finally, after updating q_0 to the value returning the best fitness $\overline{f}(q_0)^+$ we fluctuated τ_0 to infer its influence and finally we varied ρ which concluded our ACS OAT.

Note on solution representation Since some of the algorithms of choice represent solutions using real numbers whereas others (e.g. GA) utilise binary values, a common representation of these solutions needed to be utilised. We chose to utilise the binary domain since it is flexible enough to represent natural or real numbers while simple enough to represent states, such as ON and OFF. Thus all non-binary based algorithms normalised their solution vector values to the [0, 1] interval under the assumption that **if** $x \leq 0.5 \rightarrow x = 0$ **else** $x > 0.5 \rightarrow x = 1$.

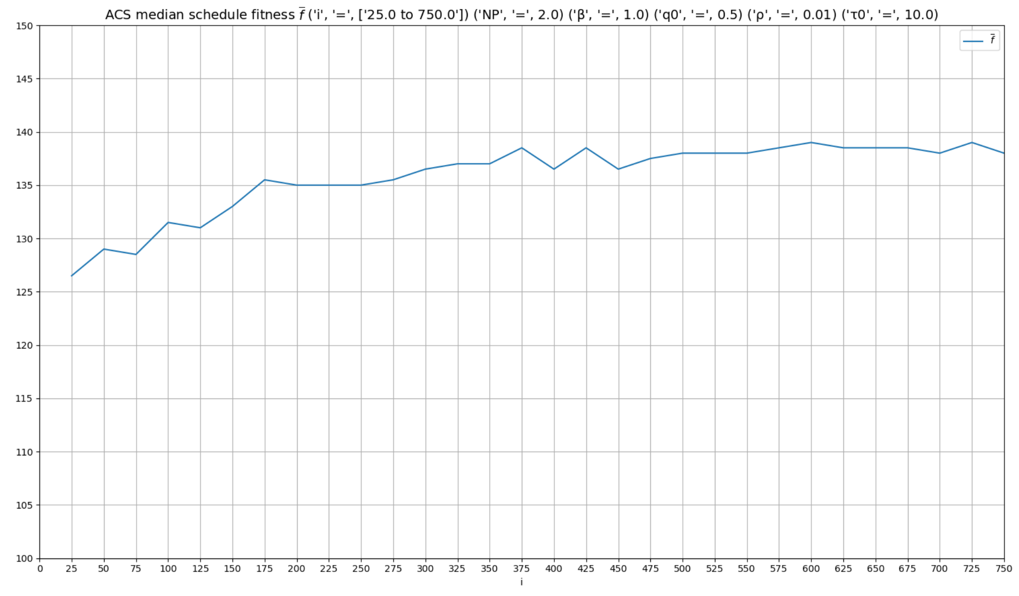
4.1.4 One-At-a-Time analysis on our algorithms

Ant Colony System

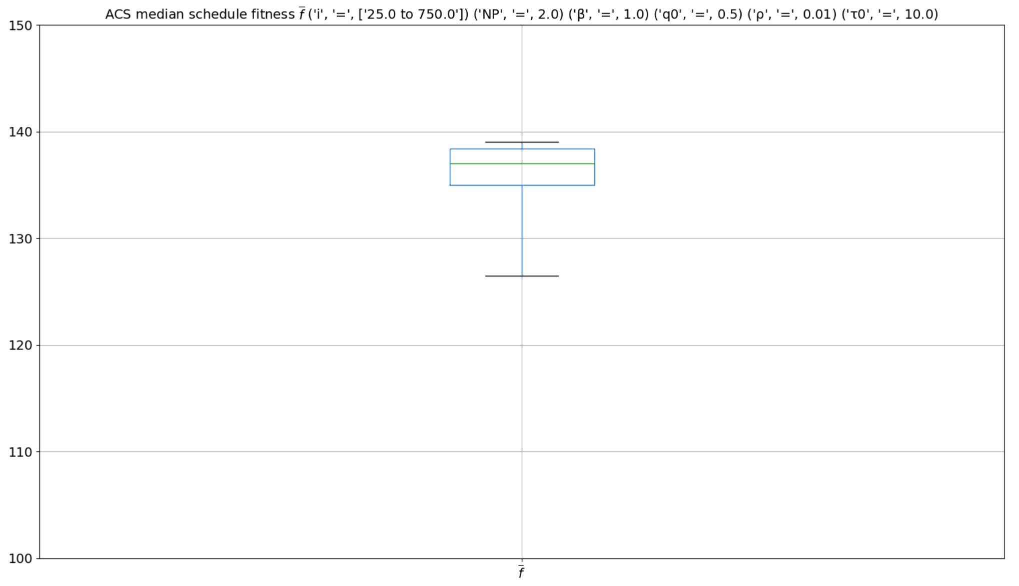
Our ACS (see section 4.1.6) has 6 controls, since the pheromone weight α is typically kept as 1 [Dorigo and Gambardella, 1997], local pheromone update was not necessary in a non Hamiltonian-cycle path [Komninou et al., 2011] and we used one colony per run. Namely, the parameters analysed were the number of iterations I_{max}, number of ants NP, heuristic information weight β, exploration vs exploitation probability q_0, global pheromone evaporation constant ρ and initial pheromone level τ_0.
We first fixed all tuning parameters but one, to a reasonable value ($NP = 2,\ \beta = 1,\ q_0 = 0.5,\ \rho = 0.01,\ \tau_0 = 10$), and analysed the influence of I_{max} to the generated schedule fitness using a step of 25 iterations. Once we established the best I_{max} seen in Figure 4.2a, we established the following rule to promote computational effort fairness: $NP \cdot I_{max} = C$. Where C, the total number of evaluations per run, is a

constant (here $C = 1200$, using NP $= 2$ times $I_{max} = 600$ which returned the highest fitness). Connecting these two controls allows us to perform a fair comparison of setups, by keeping the total number of evaluations constant. With this binding in place, we analysed the influence of NP with a step of 2 ants. Apparently, when less population members are allowed to explore for more iterations, the algorithm converges closer to a global optimum.

Adopting the new best I_{max} and NP, we went on to analyse the influence of β (Figure 4.4a) to the output, with a step of 1. Then, q_0 was examined (see Figure 4.5a) with a step of 0.1 and finally ρ (step size 0.01) and τ_0 (step size 1) were examined as seen in Figure 4.6a and 4.7a. The tuned setup comprises $I_{max} = 600,\ NP = 2,\ \beta = 3,\ q_0 = 0.5,\ \rho = 0.02,\ \tau_0 = 4$.

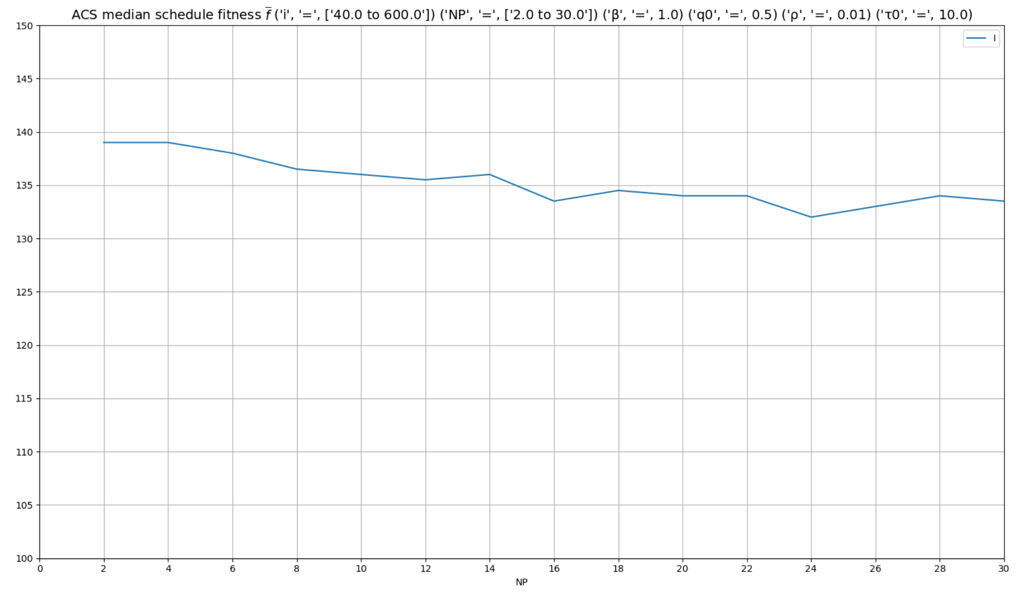


(a) Fitness $\overline{f}$ as a function of number of iterations I_{max}

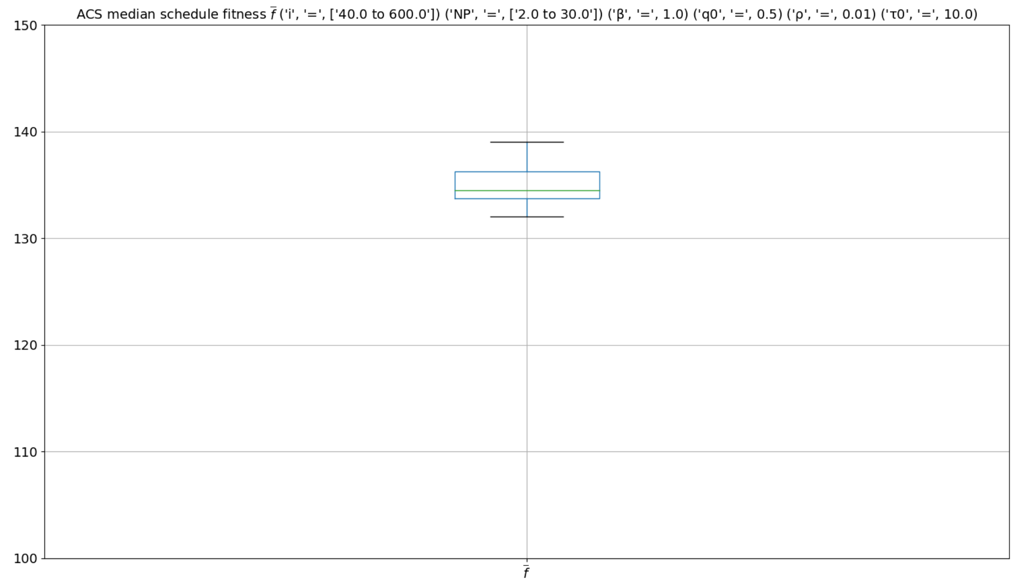


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.2: Influence of number of iterations I_{max} on schedule median fitness $\overline{f}$

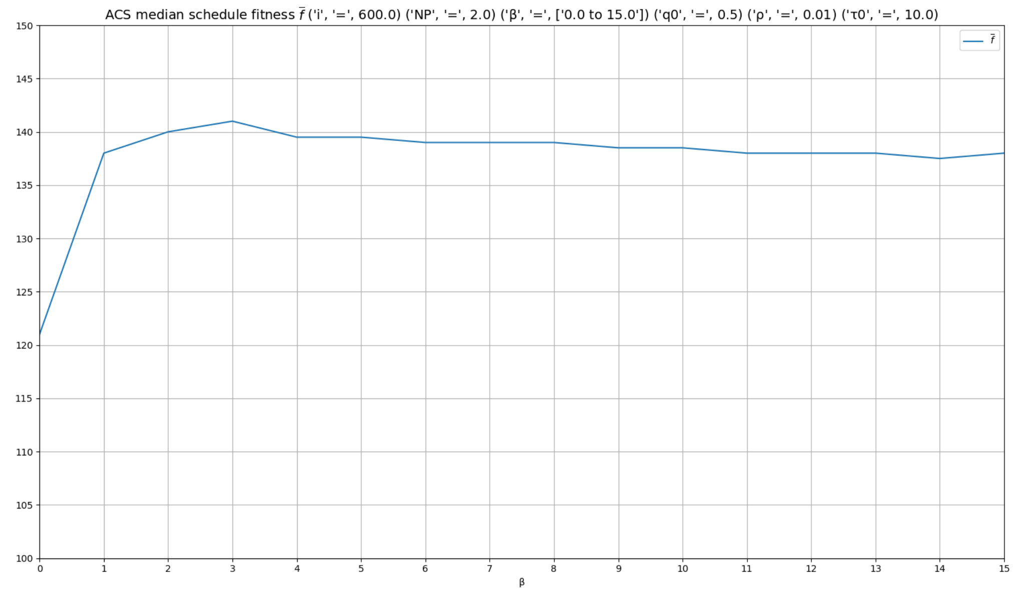


(a) Fitness $\overline{f}$ as a function of number of ants NP

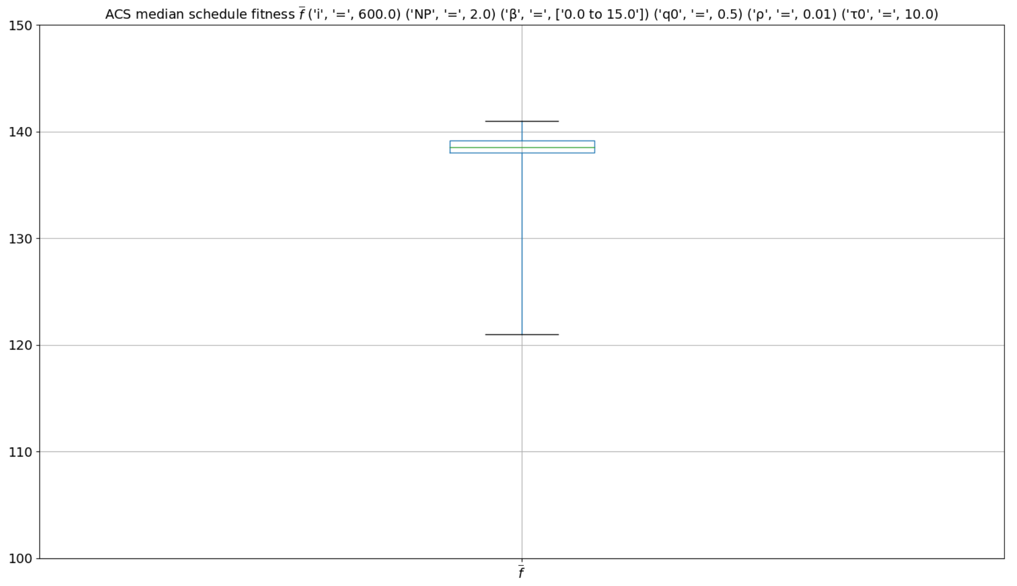


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.3: Influence of colony's number of ants NP on schedule median fitness $\overline{f}$

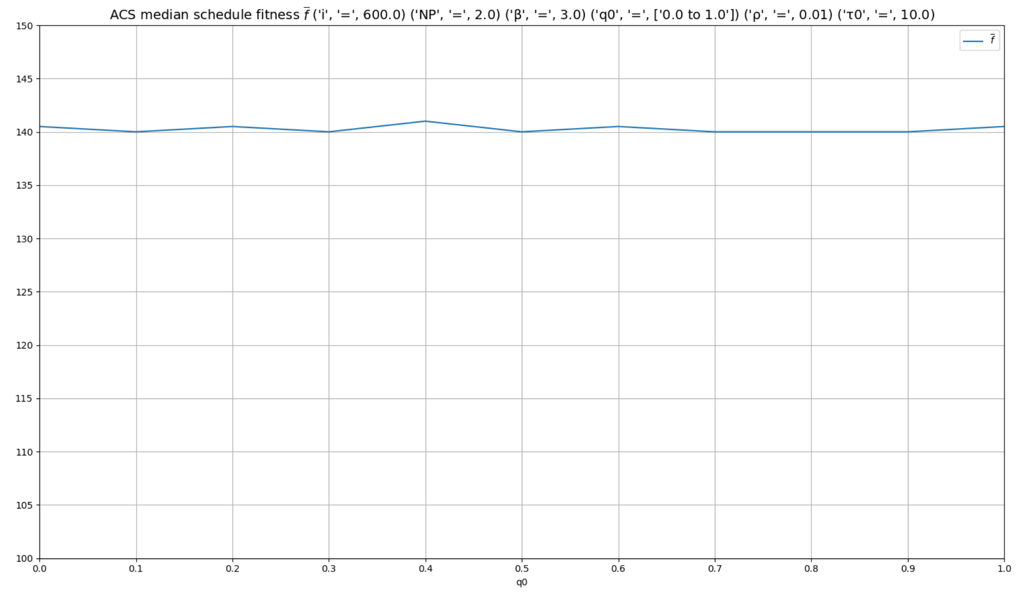


(a) Fitness $\overline{f}$ as a function of heuristic value weight β

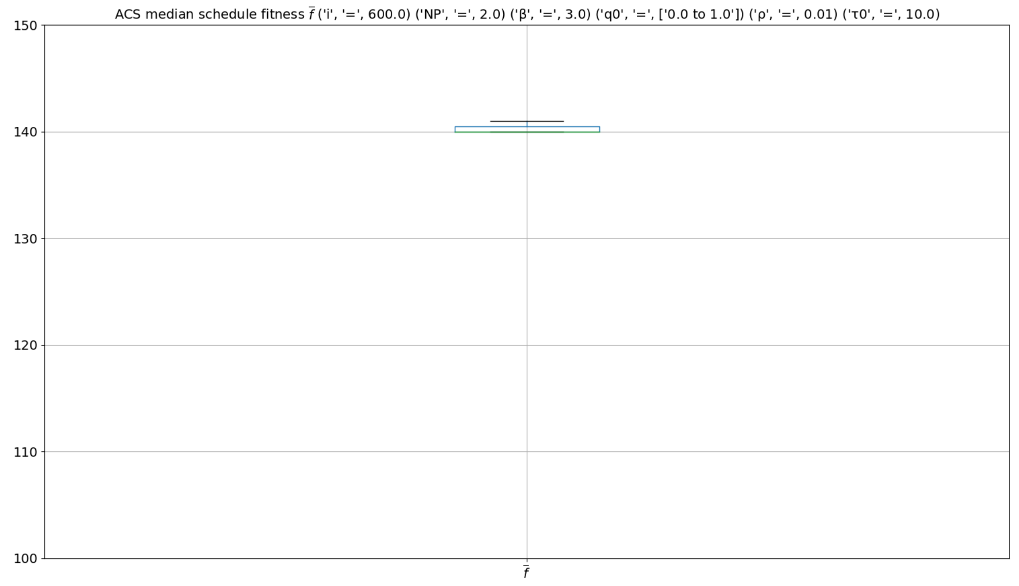


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.4: Influence of number of heuristic value weight β on schedule median fitness $\overline{f}$

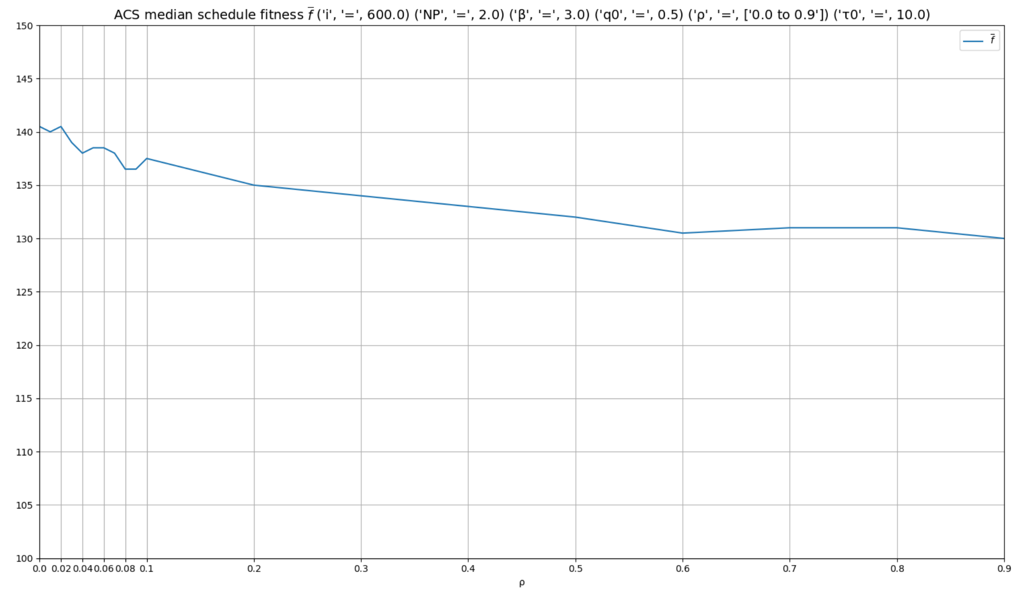


(a) Fitness $\overline{f}$ as a function of exploration vs exploitation probability q_0

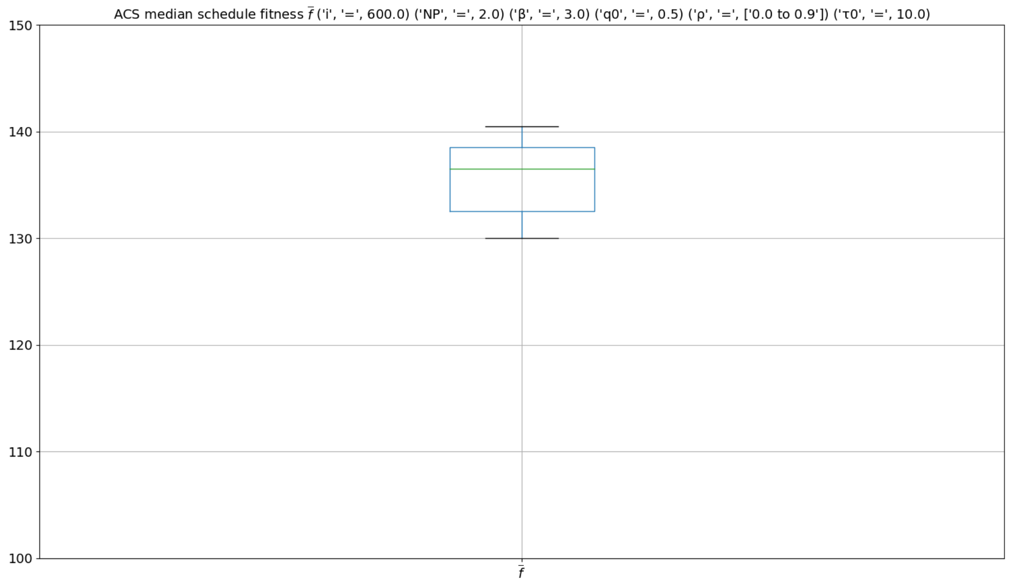


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.5: Influence of exploration vs exploitation probability q_0 on schedule median fitness $\overline{f}$

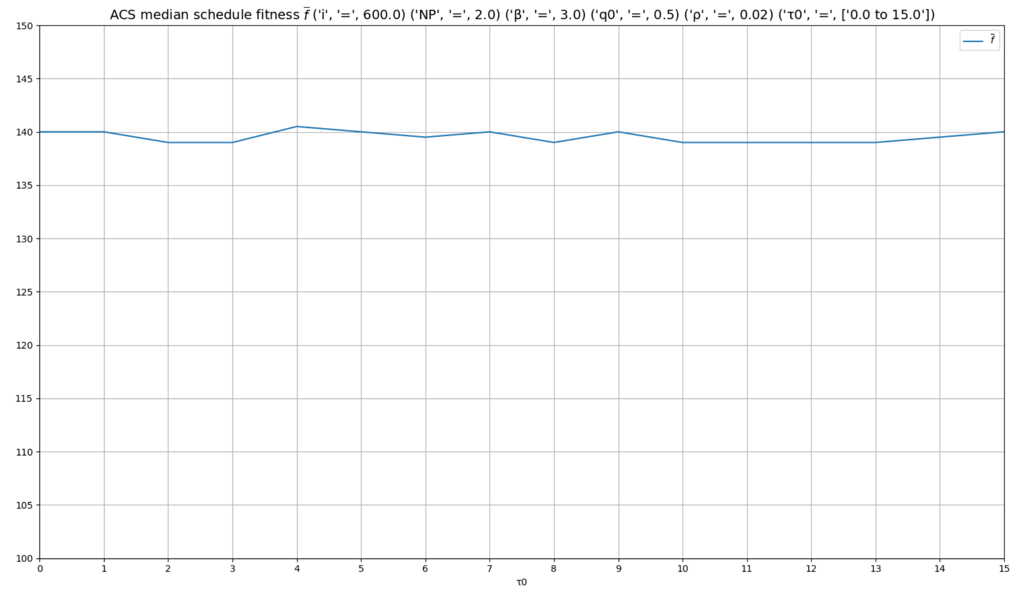


(a) Fitness $\overline{f}$ as a function of global pheromone evaporation ρ

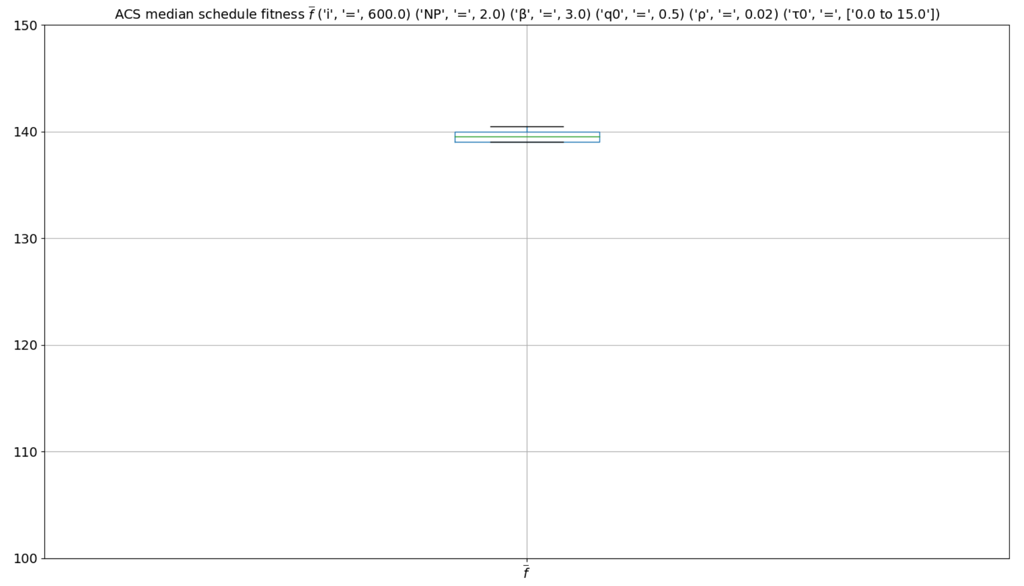


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.6: Influence of global pheromone evaporation ρ on schedule median fitness $\overline{f}$



(a) Fitness $\overline{f}$ as a function of initial global pheromone level τ_0



(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

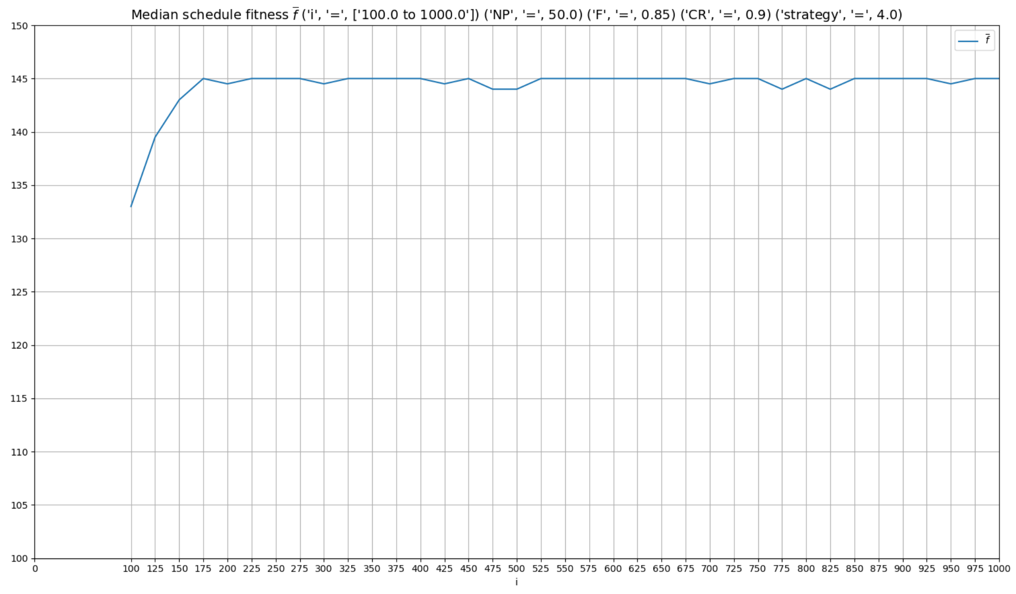
Figure 4.7: Influence of initial global pheromone level τ_0 on schedule median fitness $\overline{f}$

Differential Evolution

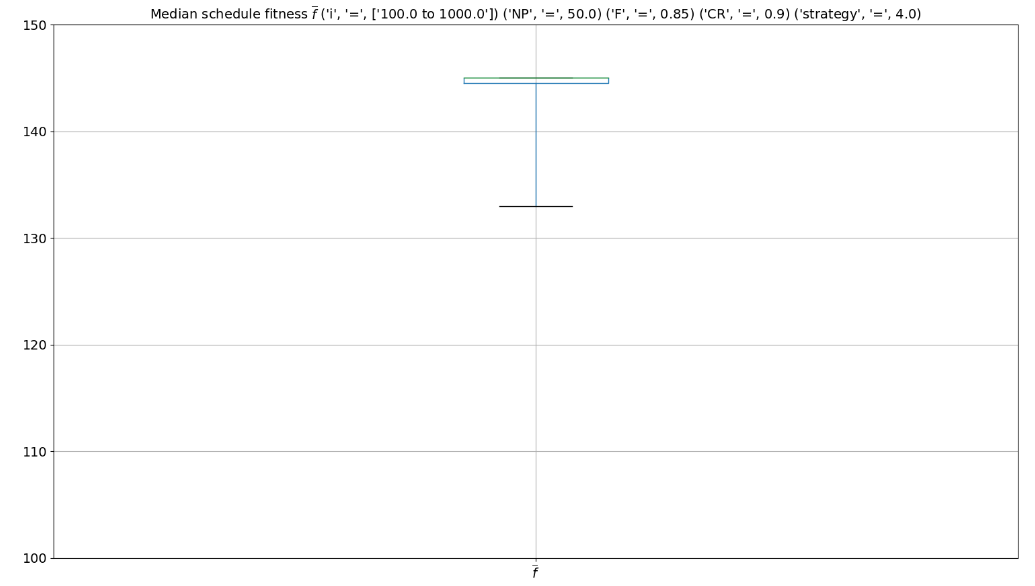
DE (section 4.1.7) control parameters comprise the number of iterations I_{max}, number of population members NP, amplification factor F, crossover probability p_{cr} and the choice of search strategy used [Price et al., 2005].
The strategies utilised in this experiment were:

1. DE/rand/1
2. DE/local-to-best/1
3. DE/best/1 with jitter
4. DE/rand/1 with per-vector-dither
5. DE/rand/1 with per-generation-dither
6. DE/rand/1 either-or-algorithm

Here, dither refers to selecting $F \in [0.5, 1.0]$ randomly per vector (strategy no. 4) or per generation (strategy no. 5), as it was found to "*improve convergence behaviour significantly, especially for noisy objective functions*" [Price and Storn, 1997]. Jitter is known in engineering as the deviation from true periodicity, given a periodic signal. In this case, it acts as low-level random noise applied to amplification factor F. Either-or-algorithm refers to randomly applying either the classical mutation rule as stated in Algorithm 2 or the F-K-Rule [Price et al., 2005] i.e. $K = 0.5(F + 1)$. All parameters were fixed, except I_{max} ($NP = 100, F = 0.85, p_{cr} = 0.9, strategy = 4$). Using a step of 25 iterations, we examined the algorithm's output response when increasing I_{max} as seen in Figure 4.8a. We then used the $NP \cdot I_{max} = C$ binding, with $C = 5000$ i.e. $NP = 50$ times the initial I_{max} value we applied, for inferring the best population size with fair comparison in mind. Tethering these two values, we examined the influence of NP on the algorithm's output, factorising C to ensure that I_{max} will be an integer. It appears that $NP = 40$ ($I_{max} = 125$) returns the highest fitness, according to Figure 4.9a. Using those values, we then examined the influence of the amplification factor F (Figure 4.10), with a step of 0.1. Finally, we examined the effect that p_{cr} has (step 0.1) seen in Figure 4.11 and how the choice of strategy influences the output (Figure 4.12). DE's tuned setup is $I_{max} = 125,\ NP = 40,\ F = 0.7,\ p_{cr} = 0.3,\ strategy = 4$.

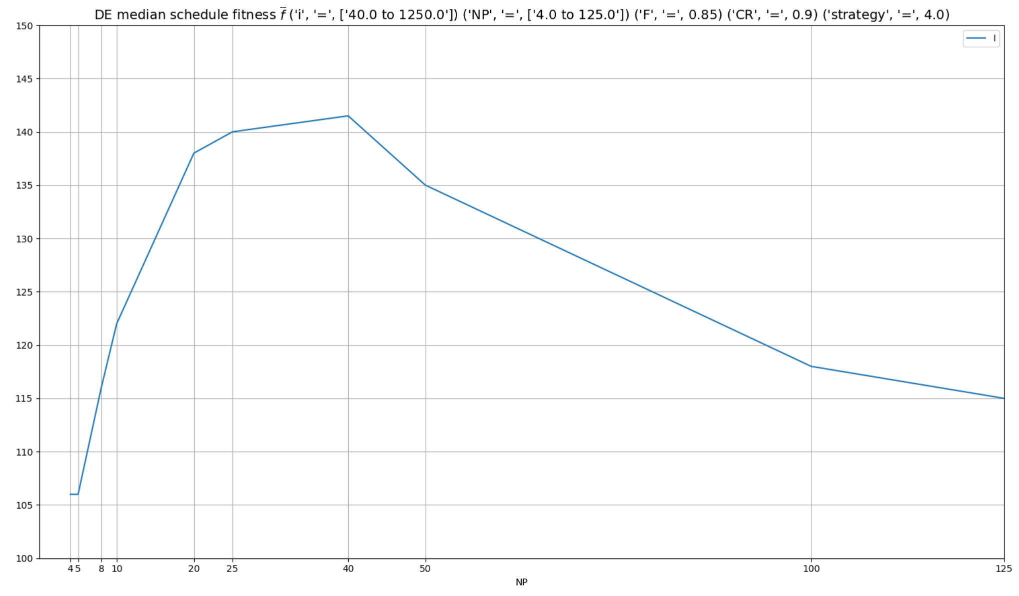


(a) Fitness $\overline{f}$ as a function of maximum generations I_{max}

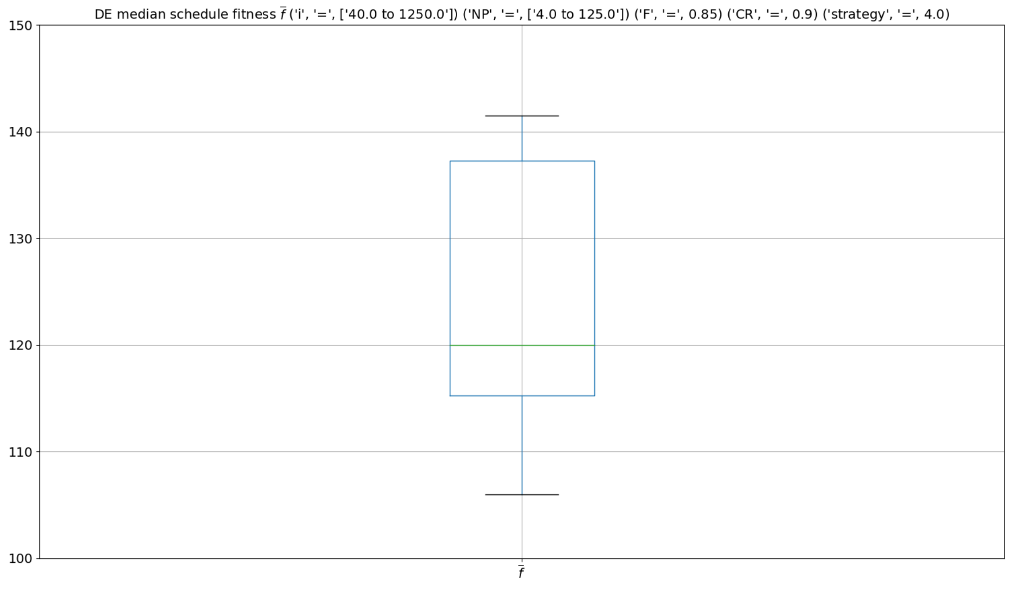


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.8: Influence of maximum generations I_{max} on schedule median fitness $\overline{f}$

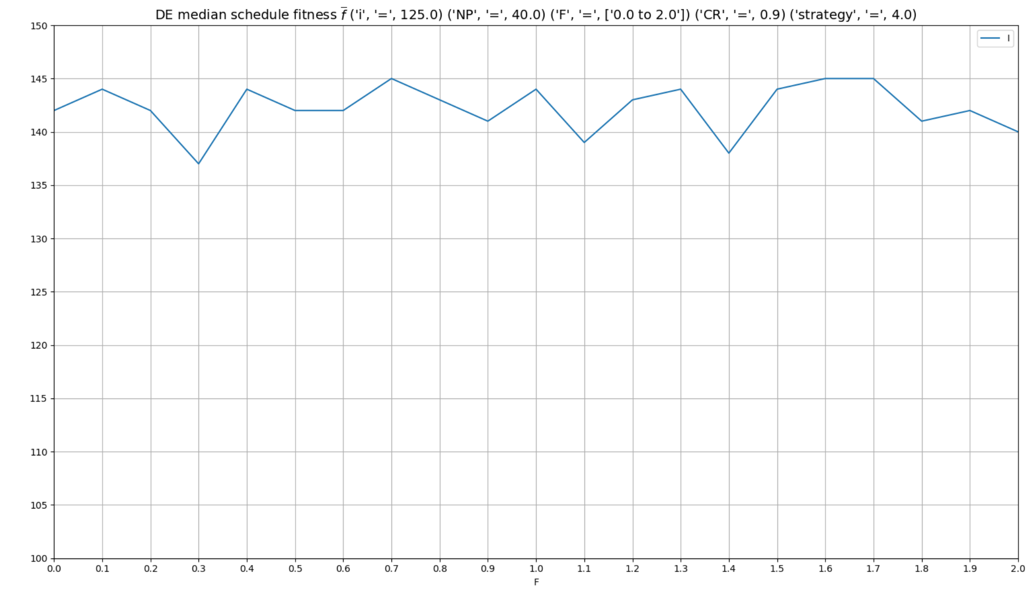


(a) Fitness $\overline{f}$ as a function of number of population members NP

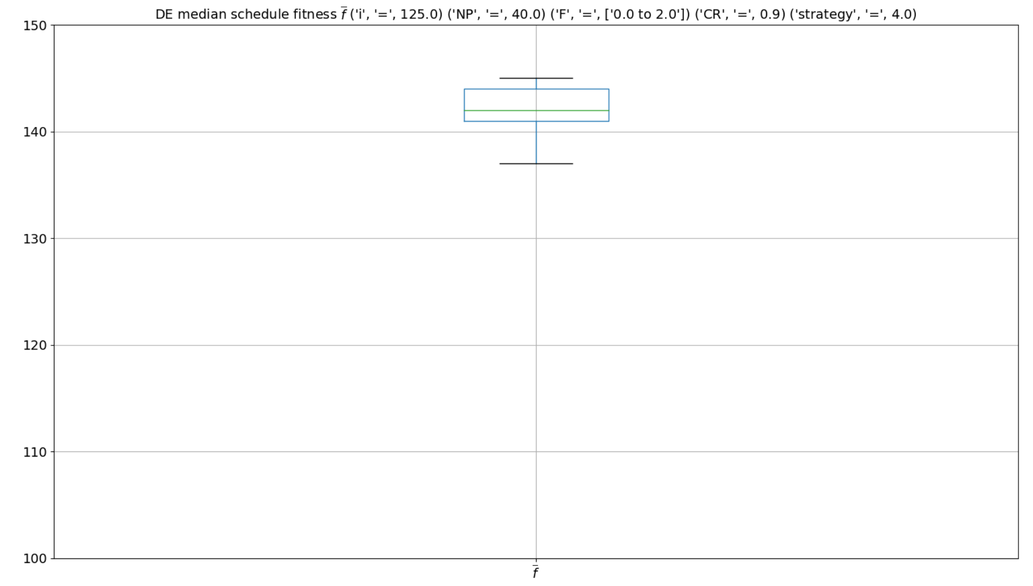


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.9: Influence of number of population members NP on schedule median fitness $\overline{f}$

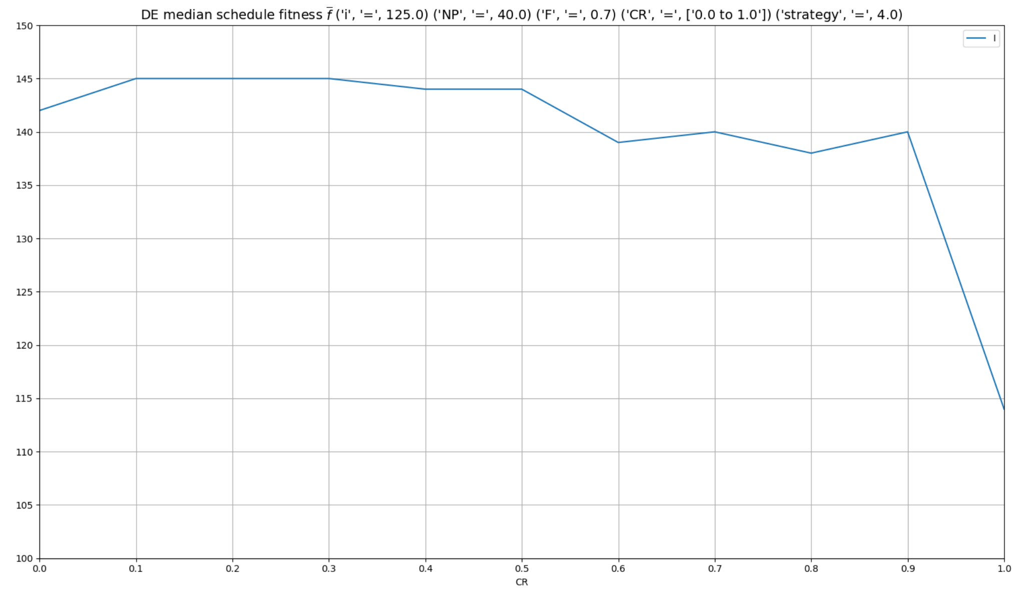


(a) Fitness $\overline{f}$ as a function of amplification factor F

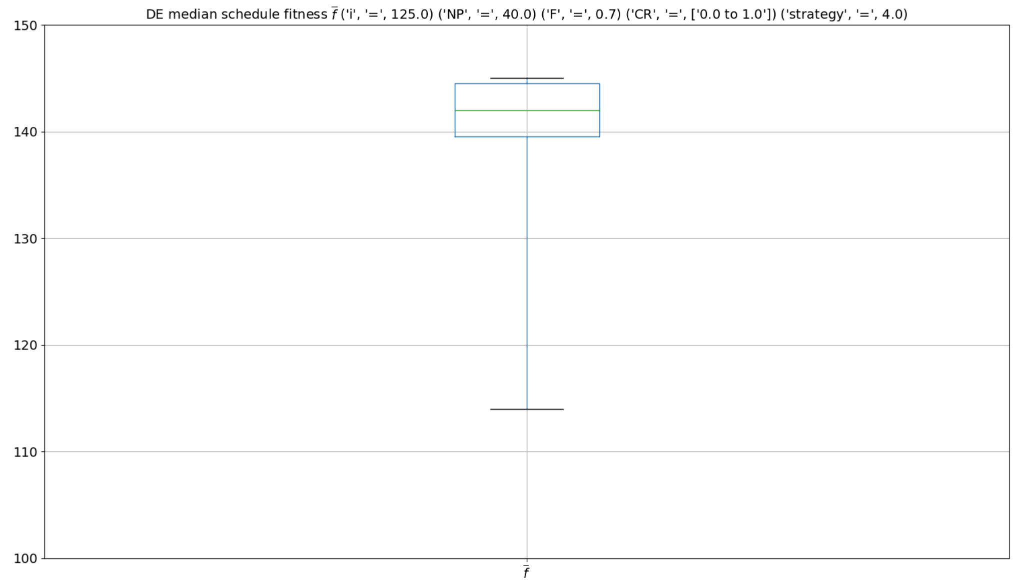


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.10: Influence of vector amplification factor F on schedule median fitness $\overline{f}$

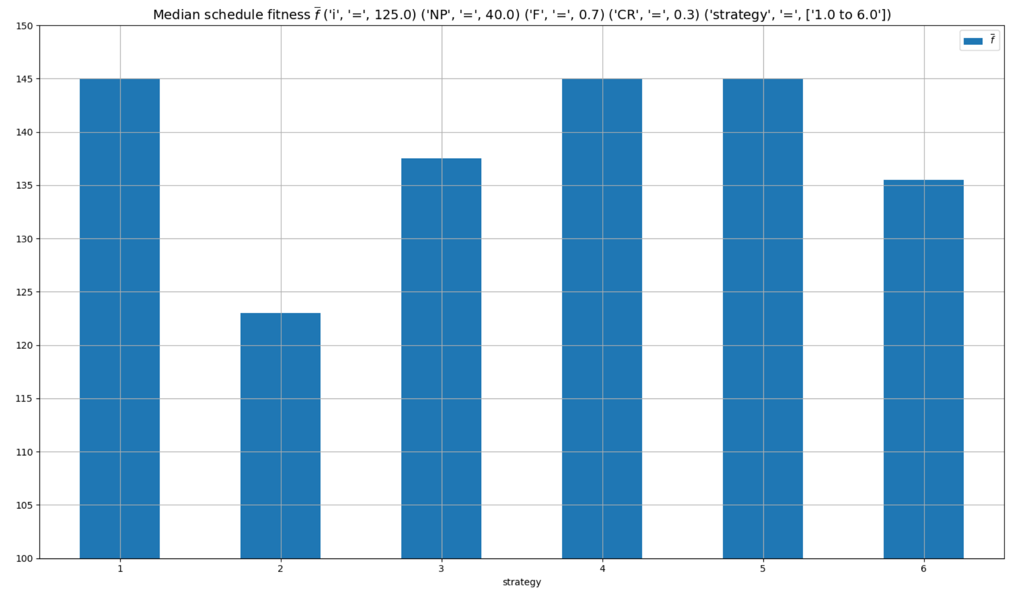


(a) Fitness $\overline{f}$ as a function of crossover rate p_{cr}

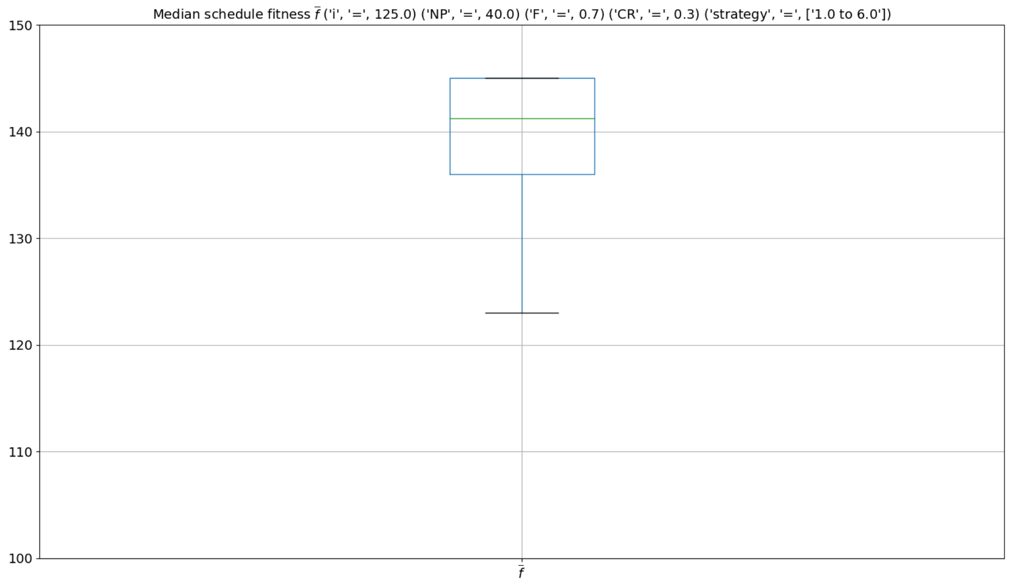


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.11: Influence of crossover rate p_{cr} on schedule median fitness $\overline{f}$



(a) Fitness $\overline{f}$ as a function of search strategy

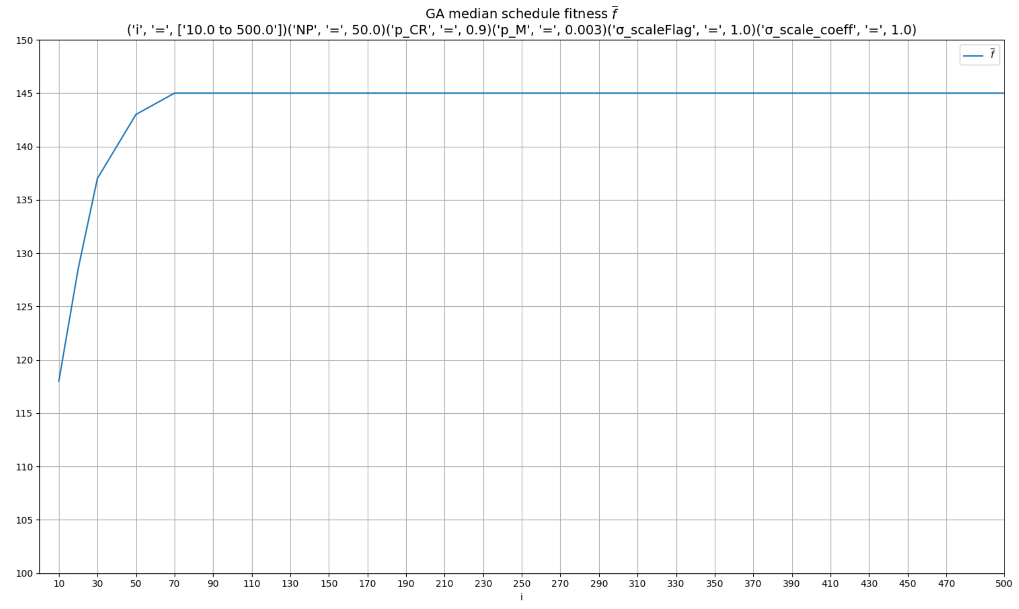


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

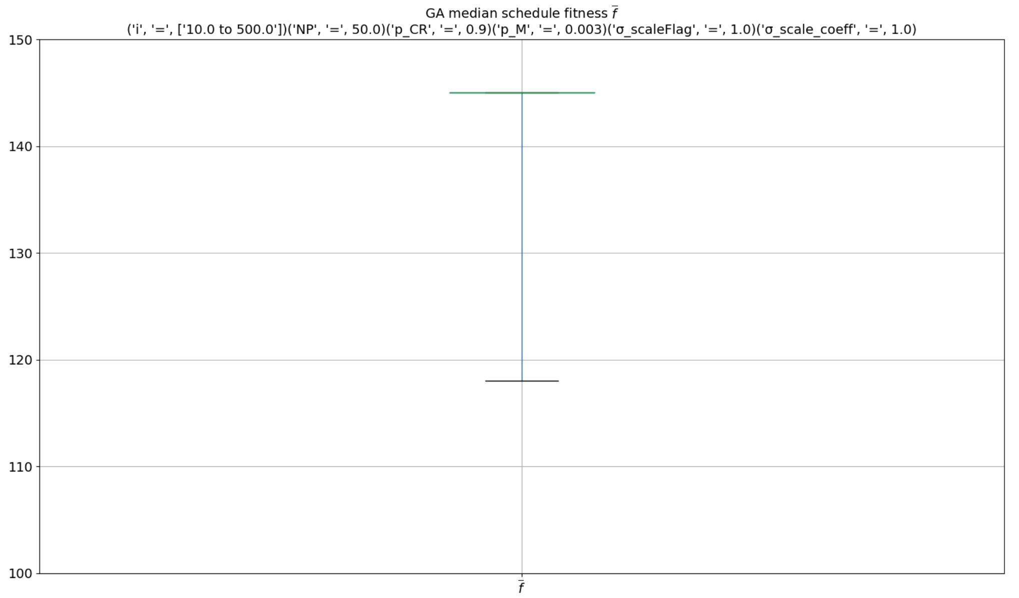
Figure 4.12: Influence of search strategy on schedule median fitness $\overline{f}$

Genetic Algorithm

The GA parameters (see section 4.1.8) that control our algorithm's behaviour are the number of generations I_{max}, number of population members NP, crossover probability p_{cr}, mutation per bit probability p_m and use of σ-scaling or not.
Keeping the rest of the parameters fixed, we first investigated the influence of I_{max} ($NP = 50$, $p_{cr} = 0.9$, $p_m = 0.003$, σ-scaling on) using a step of 20 iterations (Figure 4.13a). Following this, we bound $NP \cdot I_{max} = C$ ($C = 5000$, the product of $NP = 50$ that we initially used, times $I_{max} = 100$ which happens to return the highest fitness) to deduce the best population size, with fair computational effort in mind. Factoring C for generating integer I_{max} values, $NP = 50$ – with $I_{max} = 100$ – is a pair of values leading to closer convergence to a global optimum as seen in Figure 4.14a. Utilising these values, p_{cr} influence is then examined using a step of 0.1 (Figure 4.15a). We then observe the influence of p_m with a step of 0.001 per bit, seen in Figure 4.16a. Finally, we notice the impact of σ-scaling in Figure 4.17a. The final parameters' setup for GA is: $I_{max} = 100$, $NP = 50$, $p_{cr} = 0.9$, $p_m = 0.003$ per bit and σ-scaling on.

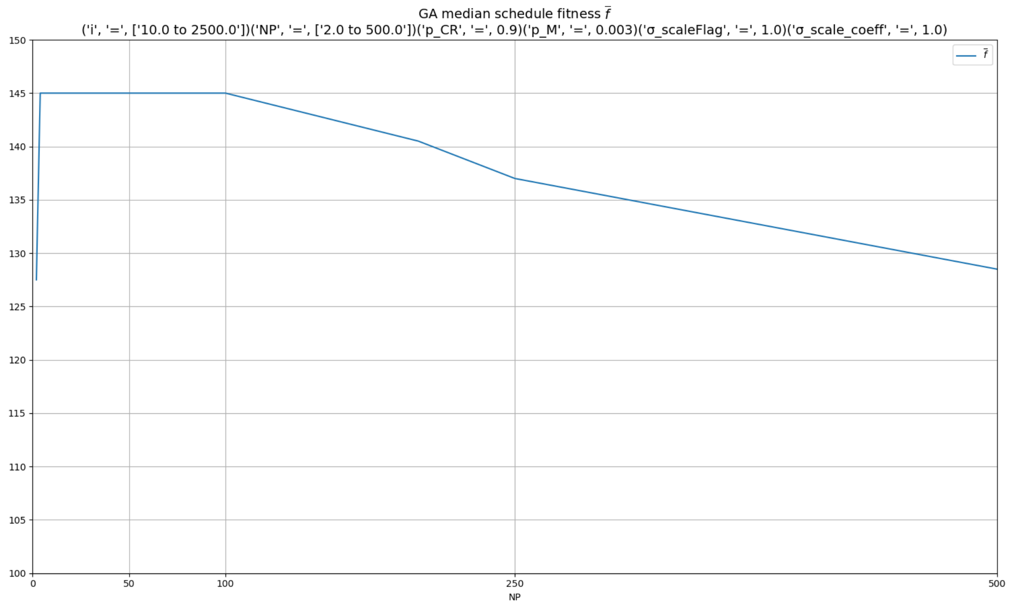


(a) Fitness $\overline{f}$ as a function of number of generations I_{max}

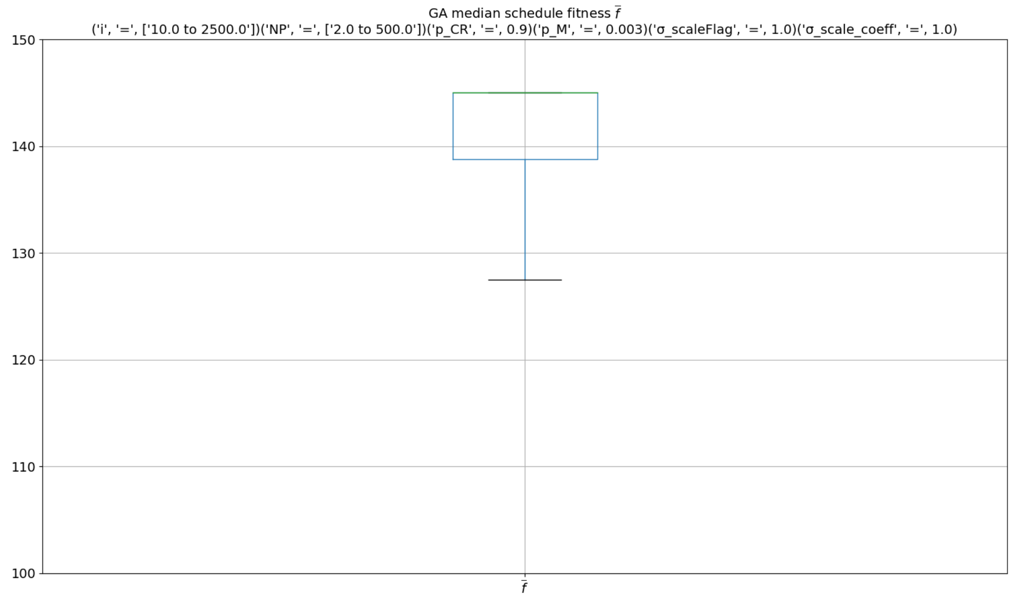


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.13: Influence of number of generations I_{max} on schedule median fitness $\overline{f}$

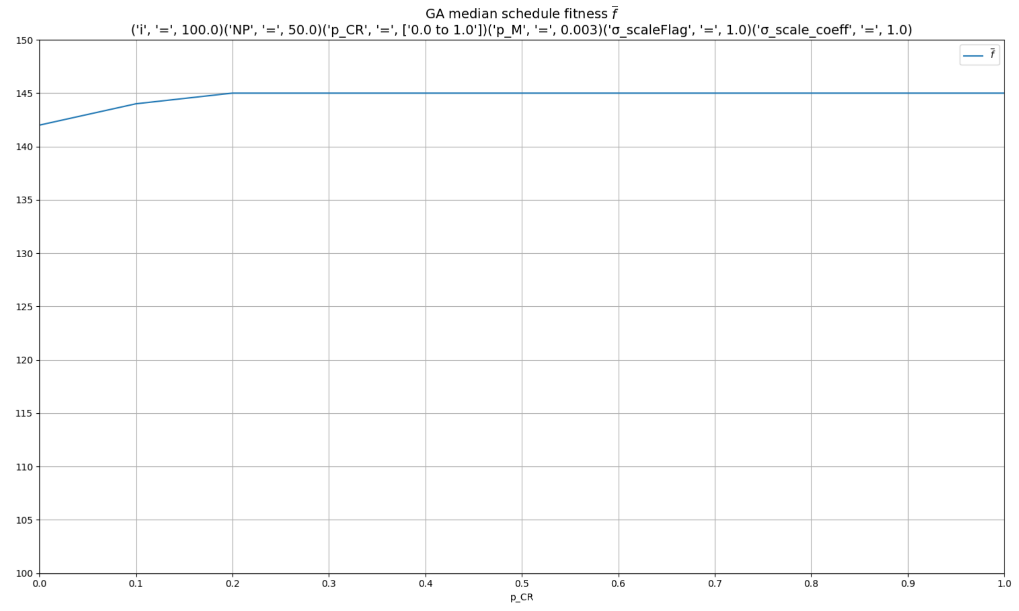


(a) Fitness $\overline{f}$ as a function of number of population members NP

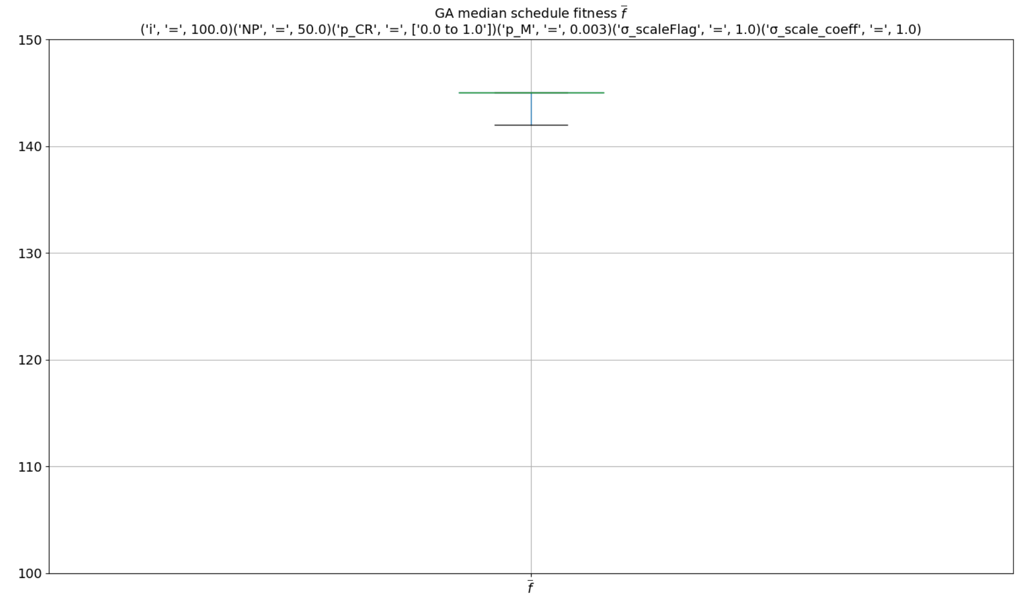


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.14: Influence of number of population members NP on schedule median fitness $\overline{f}$

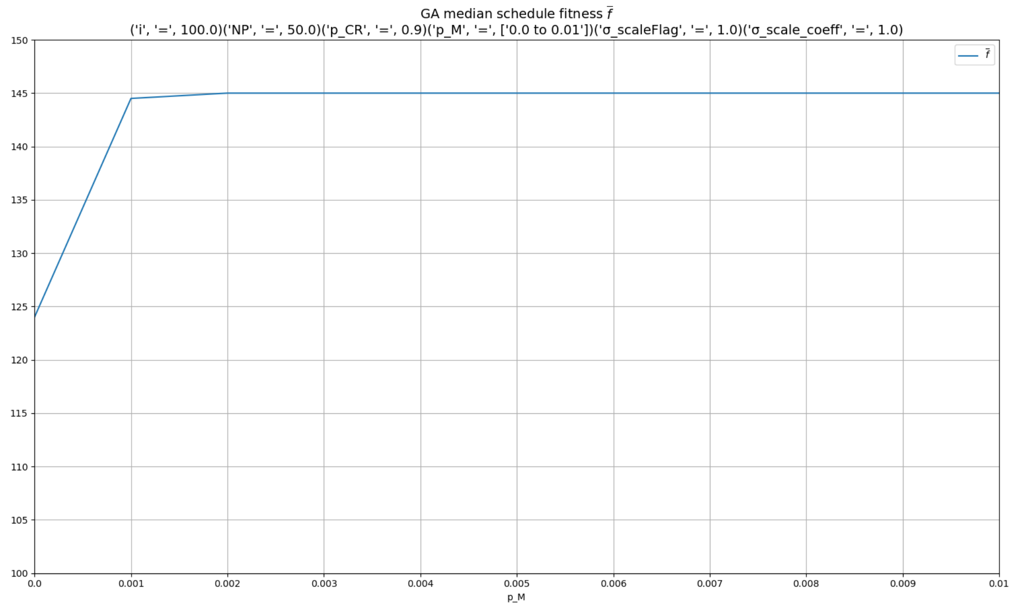


(a) Fitness $\overline{f}$ as a function of crossover probability p_{cr}

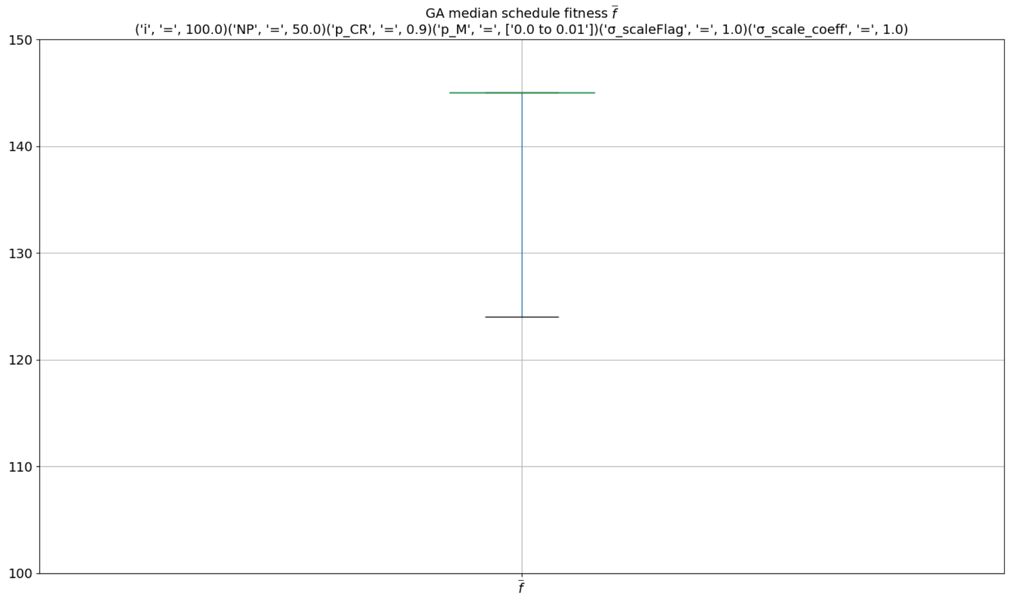


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.15: Influence of crossover probability p_{cr} on schedule median fitness $\overline{f}$

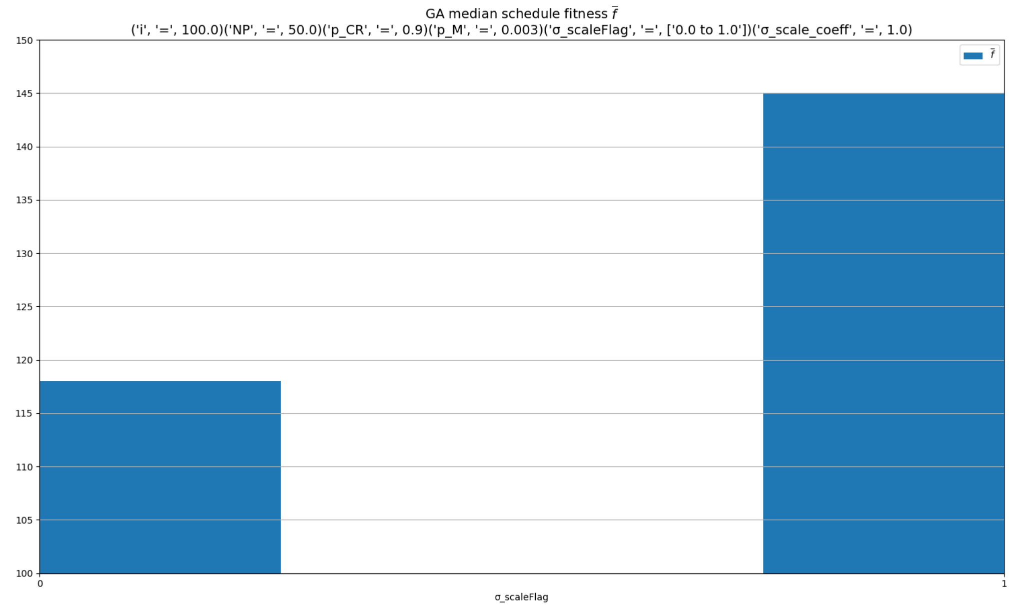


(a) Fitness $\overline{f}$ as a function of mutation probability p_m

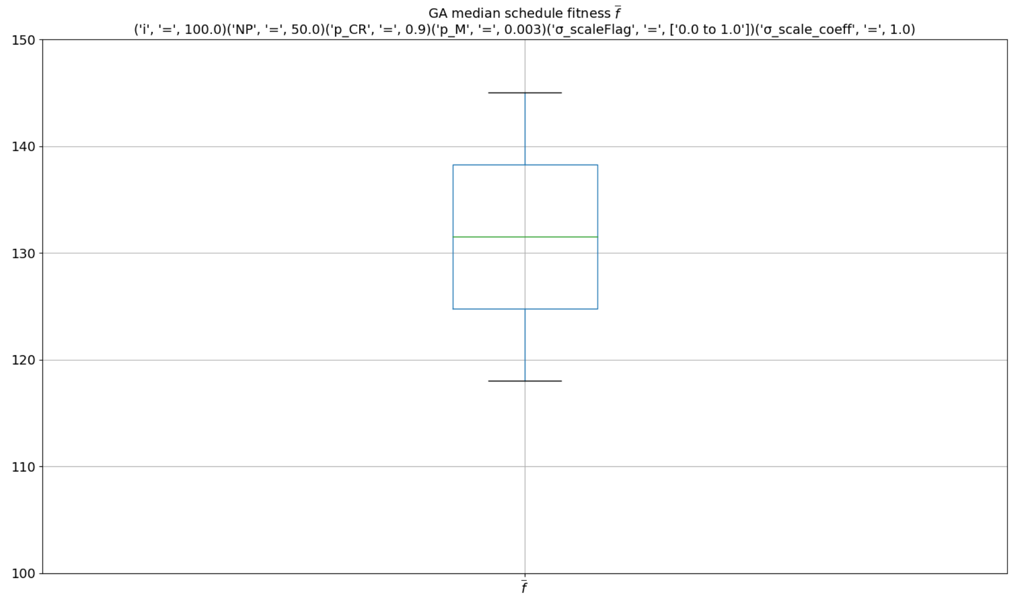


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.16: Influence of mutation probability p_m on schedule median fitness $\overline{f}$



(a) Fitness $\overline{f}$ as a function of using σ-scaling



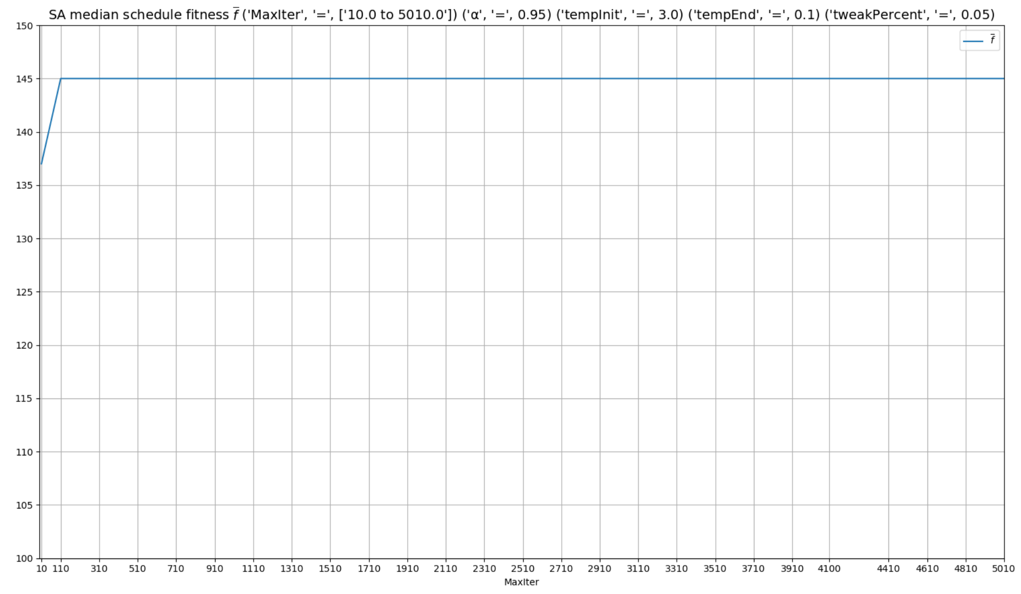
(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.17: Influence of σ-scaling on schedule median fitness $\overline{f}$

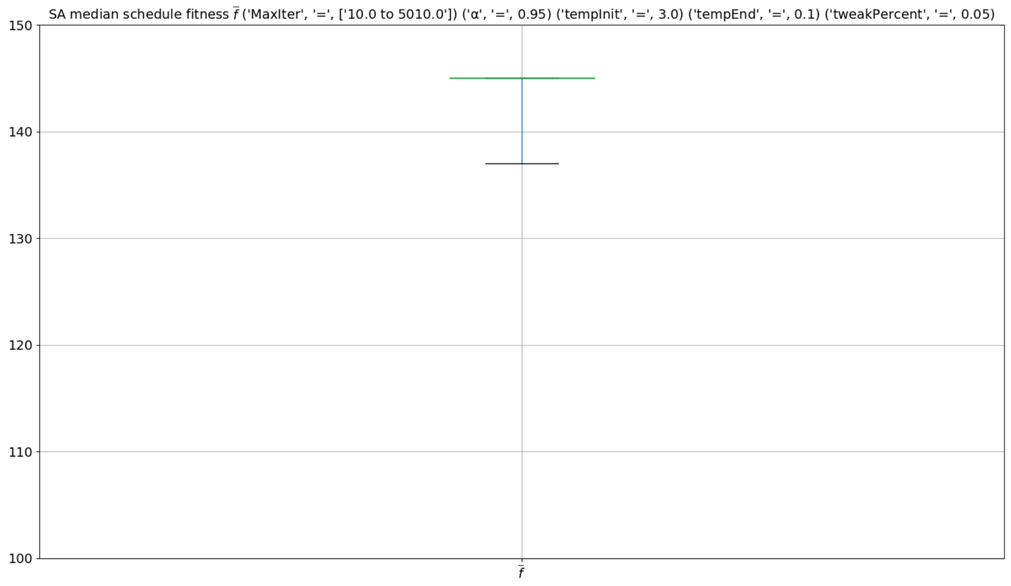
Simulated Annealing

Traditionally, SA uses 4 control parameters. Namely, the maximum number of iterations I_{max} per temperature level, a cooling schedule control (here, rate of change α), an initial system temperature $temp_{init}$ and the final system temperature $temp_{min}$. Our SA (section 4.1.11) contains a fifth control parameter, d_{max}, which specifies the maximum percentage of solution tweaking per iteration, chosen randomly per iteration. Tweaking a solution will therefore range between 1 bit and d_{max} % bits of the solution length.

We first started by observing the influence of I_{max} per temperature level (Figure 4.18a), with a step of 10 iterations, keeping every other parameter fixed ($\alpha = 0.95,\ temp_{init} = 3,\ temp_{min} = 0.1,\ d_{max} = 0.05$). Fixing the iterations limit according to the first experiment, we analyse the effect of α to the output, using a step of 0.05. Once the best rate of change of system temperature is decided, we observe if $temp_{init}$ influences SA's output (step size 0.5). Finally, we examine the influence of $temp_{min}$ with a step of 0.1 and with this result in mind we notice the influence of d_{mqx} with a step of 0.1. The best performing SA setup is $I_{max} = 110,\ \alpha = 0.95,\ temp_{init} = 3,\ temp_{min} = 0.1,\ d_{max} = 0.05$.

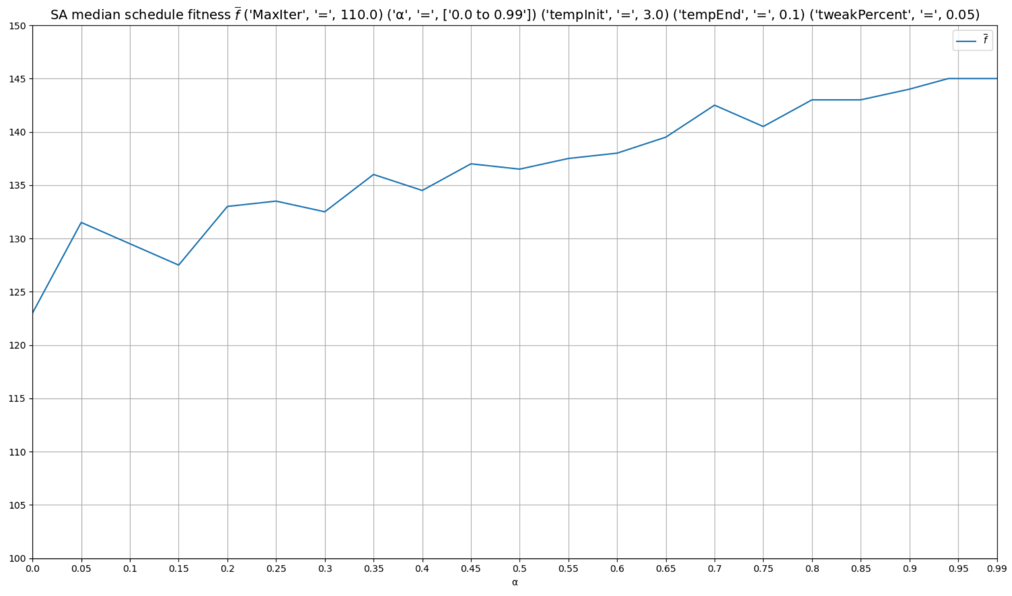


(a) Fitness $\overline{f}$ as a function of maximum iterations I_{max} per temperature level

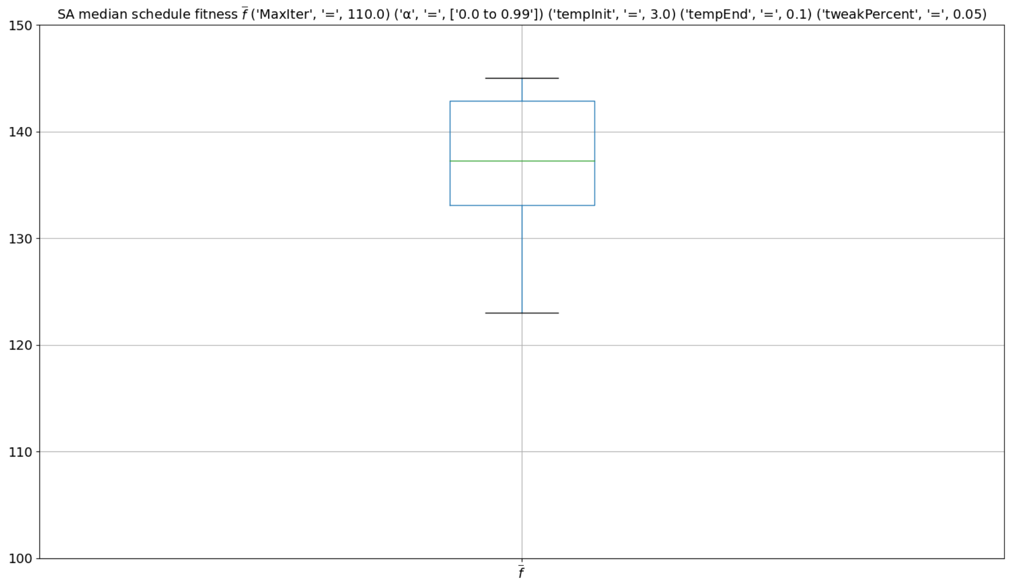


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.18: Influence of maximum iterations I_{max} per temperature level, on schedule median fitness $\overline{f}$

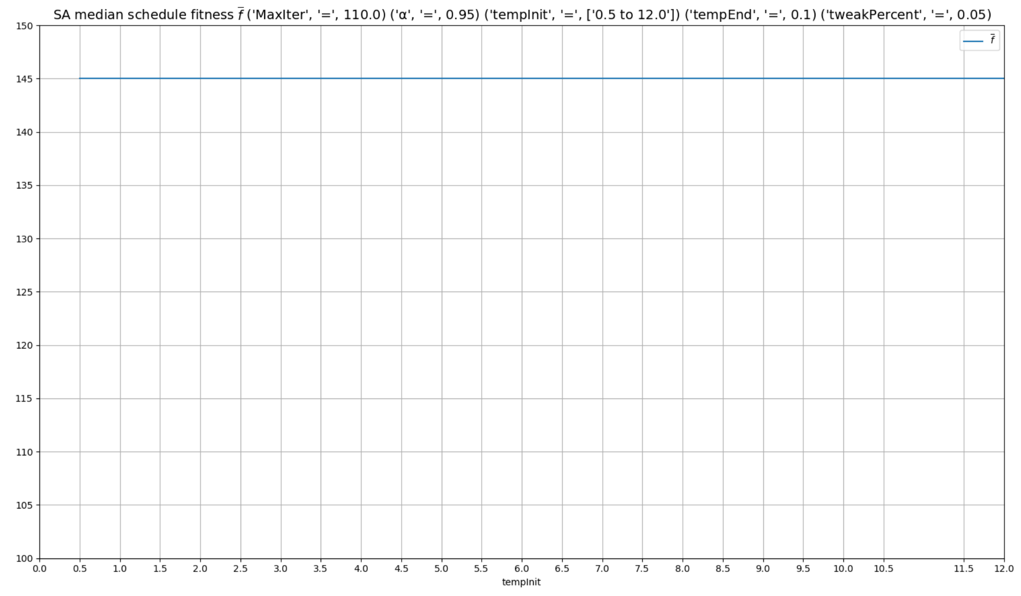


(a) Fitness $\overline{f}$ as a function of cooling rate α

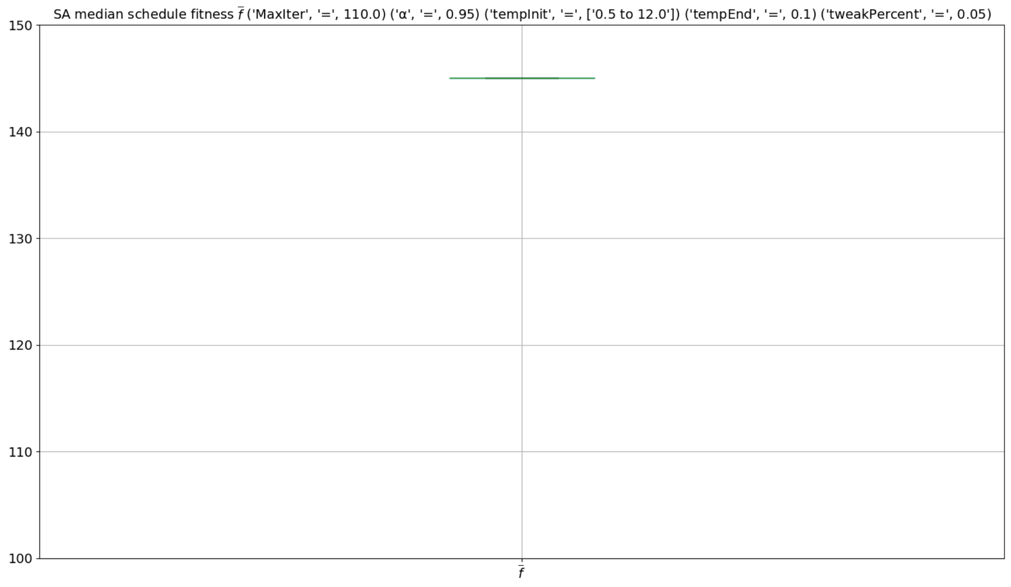


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.19: Influence of cooling rate α on schedule median fitness $\overline{f}$

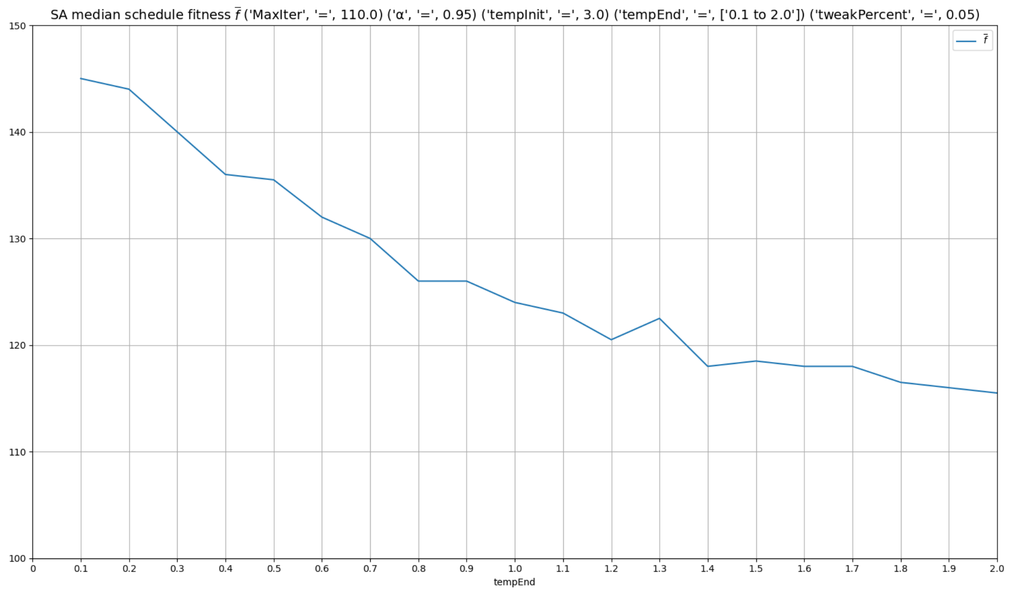


(a) Fitness $\overline{f}$ as a function of initial system temperature level $temp_{init}$

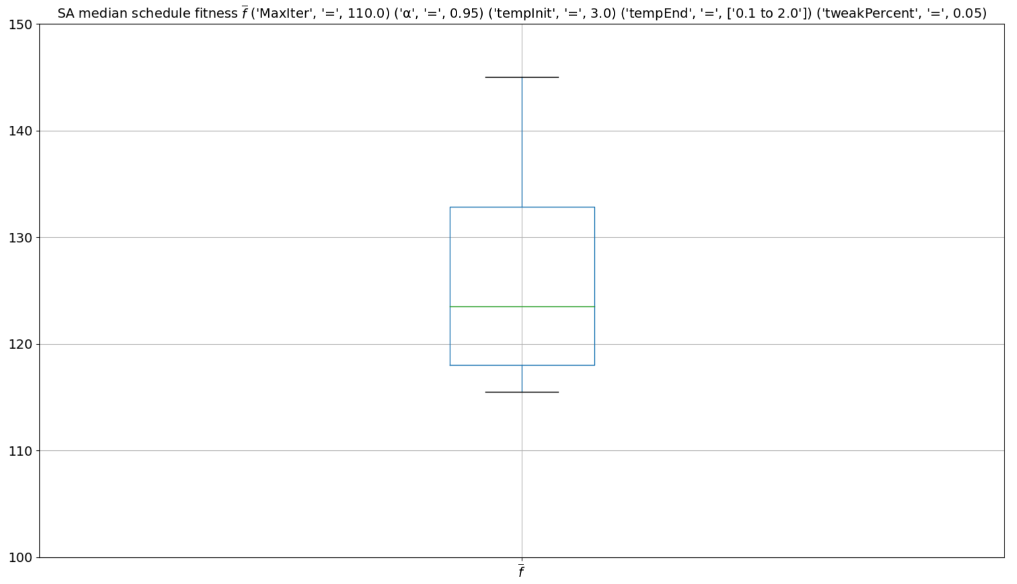


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.20: Influence of initial system temperature level $temp_{init}$ on schedule median fitness $\overline{f}$

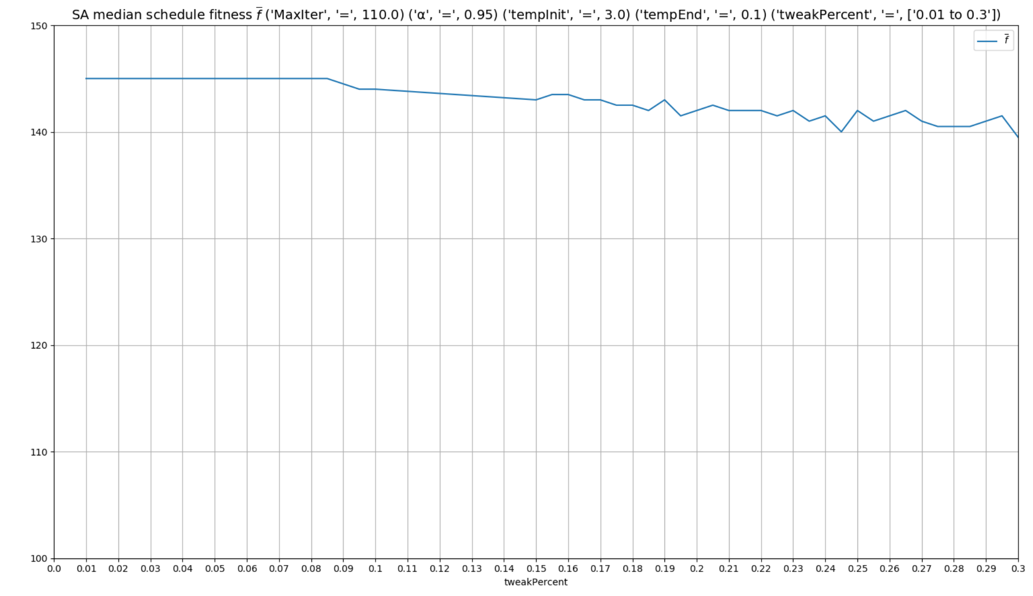


(a) Fitness $\overline{f}$ as a function of minimum system temperature level $temp_{min}$

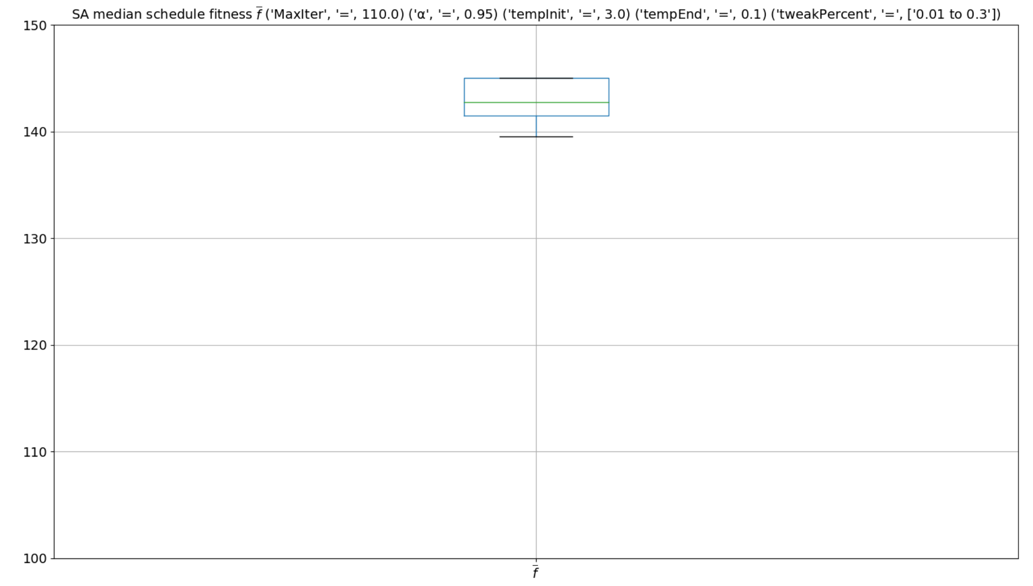


(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.21: Influence of minimum system temperature level $temp_{min}$ on schedule median fitness $\overline{f}$



(a) Fitness $\overline{f}$ as a function of maximum solution tweak percentage d_{max}



(b) Boxplot of fitness $\overline{f}$ with whiskers from minimum to maximum

Figure 4.22: Influence of maximum solution tweak percentage d_{max} on schedule median fitness $\overline{f}$

4.1.5 Algorithms used

The metaheuristics' family taxonomy contains local-search, constructive and population-based algorithms [Glover and Sörensen, 2015; Brownlee, 2011]. Local-search methods iteratively perturb a given solution using various strategies attempting to converge to global optima, taking small steps around search space "neighbourhoods" to improve a candidate solution. Constructive algorithms form solutions element-wise, choosing the best possible element per step based on a set of fitness metrics. Population-based methods operate by forming populations of initial solutions through a given mechanism e.g. randomly and then combining existing population members in an attempt to converge to a global optimum.

Our system's optimisation module contains representative algorithms from the aforementioned taxonomy, comparing their performance on an even basis. Namely the algorithms to be put to the test are ACS, DE, GA, Random Search and SA. Those algorithms form a representative sample of available techniques inspired by stigmergy, weighted vector differences, evolutionary biology and metallurgy covering a variety of optimisation strategies.

All algorithms generate their initial population or start solution randomly, when such a step is required. Every algorithm was tested both individually and combined with a local search (see Algorithm 8) to observe potential fitness improvement and execution time variation.

Below we describe the implementation of each algorithm, expressing our code algorithmically. For consistency, we have used the same notation when possible. Namely, I = iterations/generations, NP = number of members in population, V = solution candidate, D = length of solution candidate, p or P = probability, $f(\cdot)$ = fitness function, T = total number of time-steps and S = mission requirement bit-string.

The test case we utilised contains elements from two missions. Namely, it was inspired by the small European student satellite ESEO preliminary mission profile, with the calculated orbit[2] resembling the initial stages of the LISA Pathfinder scientific mission. Six tasks corresponding to ADS, CDH, TTC, PROP and two generic payloads were included. Starting from launch, which signifies $t = 0$, ADS with PROP were engaged for two minutes (1 time-step) starting from time-step 5 with a cadence of 7 steps. CDH was always on and TTC was used whenever a ground station from the aforementioned set (Malindi, Raisting, Perth, Kourou) was

[2]Launch date: *15th of November 2010, 12.00:00 UTC*, semi-major axis $\alpha = 9105.6$ km, eccentricity $e = 0.2444$, inclination $i = 34.9398$ deg., right ascension $RA = 110.4872$ deg., argument of periapsis $\omega = 90$ deg., mean anomaly $M = 130.42$ deg.

visible for more than 5 minutes. Payload 1 was turned on for a random amount of time-steps in the interval $[1, 15]$ until the last time-step before eclipse. Payload 2 was engaged at a random time-step within an eclipse and kept running for the following 13 time-steps. Similar to the OAT experiment, the EPS calculated power availability and battery energy based on the orbital characteristics. The total test case schedule length was 2.4 hours, corresponding to roughly 1 orbit, amounting to 72 time-steps.

4.1.6 Ant Colony System

The algorithm below describes how we implemented ACS (Algorithm 1). Our ACS was written by the author of this thesis, after closely following the original ACO framework[3] written by Thomas Stützle in C and licenced under the GNU General Public License. Our code was tested and verified using a randomly selected TSP problem (verified against Stützle's ACS corresponding solution), and a Euclidean 2D plane problem and 3D space problem (verified by generating the shortest path between two opposite corners). It was then applied to satellite harness routing optimisation [Komninou et al., 2011] initially, and subsequently to satellite operations scheduling.

By design, ACS is a graph traversal algorithm. This posed the challenge of having to interpret our scheduling problem using graph concepts such as edges (tour segments) and nodes (cities). To achieve this, we made some assumptions. In particular: a) ants' movements represent a step forward in time, not space i.e. city i corresponds to time-step t and city j sits on $t + 1$, b) all ants start their tour by selecting the first time-step of the mission requirement S_{t_1}, c) when ant k visits time-step t, it determines t's set of neighbouring solutions J_t by calculating which of those neighbours l, $\forall l \in J_t$ represent *feasible* solutions respecting power availability A_{t+1}, d) ant k's calculation of the heuristic values η_{tl}, correspond to the fitness of all potential next steps l within the feasible neighbourhood J_t, e) the global pheromone matrix $\boldsymbol{\tau}$ is updated by depositing the quotient of the global best fitness divided by the greedy fitness[4] $\frac{f(V^+)}{f(V_{greedy})}$ on all edges comprising the global best tour V^+, f) the global pheromone matrix is updated at the end of each iteration, with the global best tour V^+ receiving pheromone reinforcement equalling $\frac{f(V^+)}{f(V_{greedy})}$, g) The

[3]http://iridia.ulb.ac.be/~mdorigo/ACO/downloads/ACOTSP-1.03.tgz

[4]Greedy fitness V_{greedy} gives an idea of how large V^+ is expected to be. It is used as a scaling factor, to keep pheromone amplification relatively comparable. Greedy fitness was calculated as follows: for each time-step, cumulatively sum up required tasks' power consumption. When the power constraint is violated, turn off the least power draining task and continue doing so until the power constraint is met. Repeat this process for all time-steps. Sum up individual time-steps' fitness to calculate $f(V_{greedy})$

global pheromone matrix is a square matrix of size $\boldsymbol{\tau}_{vector\_lenght \times vector\_lenght}$ since it's a graph distance matrix[5] [Weisstein; Felsenstein, 2003]. The reason for utilising a distance matrix in ACS, is because "*when [the algorithm] is applied to asymmetric instances it is possible that* $\tau(r, s) \neq \tau(s, r)$" [Dorigo and Gambardella, 1997].

[5]A graph distance matrix, also known as all-pairs shortest path matrix, is the square matrix $(d_{(ij)})$ consisting of all graph distances from vertex v_i to vertex v_j. The graph's mean distance is the mean of all distances (in a connected graph). The graph diameter is defined as the maximum value of all distance matrix elements.

Algorithm 1 Ant Colony System algorithm

```
1: Initialise $I_{max}$, $NP$, $q_0 \in [0,1)$, $\beta$, $\tau_0$ {$q_0$ = exploration vs exploitation probability, $\beta$ = weight, $\tau_0$ = initial pheromone level, $\rho$ = pheromone evaporation rate}
2: Lay out $\tau_0$ pheromone on all edges of global pheromone matrix $\boldsymbol{\tau}$
3: for $I = 1 : I_{max}$ do
4:   for $k = 1 : NP$ do
5:     for $t = 1 : T-1$ do
6:       Calculate $V_{I,k,t}$ as follows:
7:       if $t = 1$ then
8:         $V_{I,k,t} = S_t$ {If this is the first time-step, set $V_{I,k,t}$ to Mission Requirement $S_t$}
9:       else if $t = T-1$ then
10:        $V_{I,k,T} = S_T$ {If this is the penultimate time-step, set final solution step $V_{I,k,T}$ to Mission Requirement $S_T$}
11:      if $rand\ [0,1) \leq q_0$ then
12:        Exploitation step $V_{I,k,t} = arg\ max(\tau_{tl} \cdot \eta_{tl}^{\beta})\ \forall l \in J_t$ {$J_t$ is ant $k$'s feasible neighbourhood (i.e. feasible task allocations for $t+1$) of ant $k$ when being at time-step $t$}
13:      else
14:        Transition probability $\begin{cases} p_{t,t+1} = \frac{\tau_{t,t+1} \cdot \eta_{t,t+1}^{\beta}}{\sum_{l \in J_t} \tau_{tl} \cdot \eta_{tl}^{\beta}}, \text{ if } t+1 \in J_t\} \end{cases}$
15:
16:        Exploration step $V_{I,k,t} = rouletteWheel(p_{t,t+1})$ {Fitness proportionate selection}
17:    Evaluate solution fitness $f(V_{I,k})$
18:  Find best-so-far $V^{bst} = arg\ max(f(V_{I,k})), k \in \{1..NP\}$
19:  if $I = 1$ then
20:    $V^+ = V^{bst}$
21:  else
22:    $V^+ = arg\ max(f(V^+),\ f(V^{bst}))$
23:  $\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \Delta\tau_{i,j}\ \forall (i,j) \in V^+$ {Global pheromone update on $V^+$. $\Delta\tau_{i,j} = \frac{f(V^+)}{f(V_{greedy})}$, $V_{greedy}$ = using Mission Requirement, turn off lowest consumption tasks per time-step until power constraint is met}
24: return $V^+$
```

The main control parameters for Ant Colony System include the colony's number of ants NP, each run's number of iterations I_{max}, the exploration vs. exploitation tendency governed by probability q_0, the global pheromone initial value τ_0, the use of global pheromone update and its rate of evaporation ρ and the amplification β of the heuristic value used. Finally the algorithm's selection strategy can also affect its performance, with roulette wheel selection being the method normally used.
The number of ants NP in a colony determines how widely will the search space be explored. Ants need to be given enough time to perform their search, thus a balance needs to be kept between the number of iterations I_{max} and the number of ants. In multi-objective optimisation, more than one colony could be used, with the total number of colonies corresponding to the number of objectives to solve for. Exploration vs exploitation of information also needs to be carefully balanced since little exploration may lead to stagnation while little exploitation may lead to a virtually random search. Pheromone evaporation rate ρ should allow ants enough time to explore the search space before rendering it less favourable thus excluding it from ant search. Finally, amplification factors α and β specify how much influence does heuristic information hold over pheromone (reflecting ant experience).
The best performing setup was: $I_{max} = 600$, $NP = 2$, $\beta = 3$, $q_0 = 0.5$, $\rho = 0.02$, $\tau_0 = 4$.

4.1.7 Differential Evolution

Differential Evolution (DE) (Code authors: Rainer Storn, Ken Price, Arnold Neumaier, Jim Van Zandt from the University of California, Berkeley and released under the GNU General Public License) was tested and verified for performing constrained optimisation using Rosenbrock's function [Rosenbrock, 1960a] with bound constraints.
DE incorporates different search strategies of the form DE/x/y/z. Where x specifies which vector will be mutated (rand or best), y is the number of difference vectors used, z states the crossover scheme (bin = binomial or exp = exponential as described by Storn et al. [Storn and Price, 1997], with binomial being the most competitive scheme [Das and Suganthan, 2011]).
In DE, we represented a mission operations schedule as a vector of length $n \cdot T$, which corresponds to one population member. Generated solution vectors are always checked and corrected for feasibility, by cumulatively summing the power consumption of each scheduled task according to the mission requirements, until the

power availability ceiling is met. The remaining required tasks are not included.

Algorithm 2 Differential Evolution algorithm

```
1: Initialise $I_{max}$, $F$, $p_{cr}$, $NP$, $D$ {$F$ = amplification of differential variation ($\overrightarrow{V}_b - \overrightarrow{V}_c$), $p_{cr}$ = crossover constant, $D$ = vector dimensions i.e. length}
2: for $j = 1 : NP$ do
3:     Initialise target vector $\overrightarrow{V}_{1,j}$ as $D$-length vector of random real numbers $(0, 1)$
4: for $i = 1 : I_{max}$ do
5:     for $j = 1 : NP$ do {Calculate mutant vectors}
6:         Randomly choose 3 population members $\overrightarrow{V}_{i,a}, \overrightarrow{V}_{i,b}, \overrightarrow{V}_{i,c}$ $s.t.$ $\overrightarrow{V}_{i,a} \neq \overrightarrow{V}_{i,b} \neq \overrightarrow{V}_{i,c} \neq \overrightarrow{V}_{i,j}$
7:         Calculate mutant vector $\overrightarrow{v}_{i+1,j} = \overrightarrow{V}_{i,a} + F \cdot (\overrightarrow{V}_{i,b} - \overrightarrow{V}_{i,c})$ for each target vector
8:     for $j = 1 : NP$ do {Crossover – Calculate trial vectors}
9:         for $k = 1 : D$ do
10:            if $rand[0, 1) \leq p_{cr}$ $or$ $k = rand[1, D]$ then
11:                $crossover = 1$
12:            else
13:                $crossover = 0$
14:            Form trial vector $\overrightarrow{u}_{i+1,j,k} = \begin{cases} v_{i+1,\ j,\ k} & \text{if } crossover = 1 \\ V_{i,\ j,\ k} & \text{otherwise} \end{cases}$
15:        for $j = 1 : NP$ do {Selection}
16:            if $f(\overrightarrow{u}_{i+1,j}) > f(\overrightarrow{V}_{i,j})$ then
17:                $\overrightarrow{V}_{i+1,j} = \overrightarrow{u}_{i+1,j}$
18:            else
19:                $\overrightarrow{V}_{i+1,j} \rightarrow \overrightarrow{V}_{i,j}$
20:    return $\overrightarrow{V}^+$
```

Differential Evolution can be controlled through its number of population members NP, the total number of iterations I_{max} per run, the vectors' crossover probability p_{cr}, the vectors' difference weight F as well as the algorithmic search strategy. Typically DE utilises DE/1/rand/bin as a standard strategy, selecting a random vector to mutate, one difference vector and a binomial crossover scheme. The number of parameter vectors NP involved decides how wide of a search the algorithm performs, given enough time to explore i.e. enough iterations I_{max}. The probability p_{cr} governs how frequently will population members share their experience through crossover. Vectors' difference weight F amplifies the vector difference, acting as a step size when exploring the search space. Finally the standard DE strategy selects vectors randomly to generate new – possibly fitter – vectors using binomial crossover, a scheme that seems to behave more predictably [Zaharie, 2007]. The setup allowing DE to perform best was: $I_{max} = 125$, $NP = 40$, $F = 0.7$, $p_{cr} = 0.3$, $strategy = 4$ (corresponding to DE/1/rand/bin). It is worth noting that K. Price and R. Storn – the original authors of this algorithm – give some useful practical setup advice in DE's homepage [Price and Storn, 1997]. They encourage practitioners to try some classical settings first, with their advice being based on empirical findings using real-world problems. For example, setting the number of population NP to 10 times the solution vector length D, setting the difference weight $F = 0.8$ and the probability of crossover $p_{cr} = 0.9$.

4.1.8 Genetic Algorithm

The Genetic Algorithm (Code author: Keki Burjorjee [Burjorjee, 2009], Brandeis University, released under the Apache License 2.0) version we used, as described in Algorithm 3, was verified on its ability to perform maximisation using a Royal Road[6] fitness function [Mitchell, 1998]. We then introduced our problem including its power constraint and verified that the constraint was respected. The present GA code implements the specification of a simple GA found in Mitchell's influential book with two main differences; i) To avoid selection bias, SUS (Stochastic Universal Sampling) [Baker, 1987] was used in place of FPS (Fitness Proportional Selection). SUS differs from FPS in that instead of spinning a "roulette wheel" N times for selecting N parents, we spin the wheel once but generate N equally spaced pointers used to select N parents. ii) To moderate selection pressure throughout the algorithm's run, σ-scaling was applied. The scaling function below, was used to adjust fitness $f(V_i)$ for chromosome i during generation I:

$$f(V_i) = \begin{cases} 1 + \frac{f(V_i) - \overline{f}(V)}{s \cdot \sigma(f(V))} & \text{if } \sigma(f(V)) \neq 0 \\ 1 & \text{if } \sigma(f(V)) = 0 \end{cases} \tag{4.3}$$

where $f(V_i)$ is the fitness value of chromosome i, $\overline{f}(V)$ is the population's mean fitness, s is the scaling factor and $\sigma(f(V))$ is the standard deviation of the population fitnesses at current generation I. The current population fitness' standard deviation is used to scale fitness values, in order to maintain somewhat constant

[6]According to Mitchell: *"[To investigate recombination in detail we] designed a class of fitness landscapes, called Royal Road functions, that were meant to capture the essence of building blocks in an idealized form."*

selection pressure during the optimisation process. Chromosomes with fitness below $s \cdot \sigma(f(V))$ are assigned a very low fitness. At the beginning of a run, the population's fitnesses standard deviation is typically high, with fitter chromosomes not being many standard deviations above the mean. As the run progresses, the population is more converged and the standard deviation is lower. Fitter individuals will stand out more, allowing evolution to continue [Hancock, 1994].

Algorithm 3 Genetic Algorithm

```
1: Initialise $I_{max}$, $NP$, $p_{cr}$, $p_m$ {$I_{max}$ = maximum number of generations, NP = population size (it must be even, due to algorithm design), $p_{cr}$ = crossover probability, $p_m$ = probability of mutation}
2: Generate random population array $V$ comprising $NP$, $D$-long chromosomes {$D = n \cdot T$, the number of tasks $n$ times the number of time-steps $T$}
3: while $I \leq I_{max}$ do
4:   for $i = 1 : NP$ do
5:     Evaluate candidate solution fitness, $f(V_i)$
6:     Apply $\sigma$-scaling on $f(V_i)$ {Increase selection pressure}
7:   Select $NP/2$ number of chromosomes $parents1$ from $V$, using SUS {Selection of one half of each mating pair}
8:   $parents2 = V - parents1$ {Other half of each pair. Used for selection “without replacement”, each chromosome is selected once}
9:   $Vcand = [\ ]$
10:  for $i = 1 : NP/2$ do {Crossover}
11:    if $rand[0,1) \leq p_{cr}$ then
12:      Randomly select crossover locus $loc = rand[2, \|parents1\|)$
13:      $child1_i = parents1_i[1 : loc/2] + parents2_i[loc/2 + 1 : end]$ {First offspring}
14:      $child2_i = parents2_i[1 : loc/2] + parents1_i[loc/2 + 1 : end]$ {Second offspring}
15:    for $j = 1 : D$ do {Mutation}
16:      if $rand[0,1) \leq p_m$ then
17:        $child1_{i,j} = \textbf{not}\ child1_{i,j}$
18:      if $rand[0,1) \leq p_m$ then
19:        $child2_{i,j} = \textbf{not}\ child2_{i,j}$
20:  $Vcand = Vcand + [child1,\ child2]$ {Merge children into a single candidate population}
21:  for $i = 1 : NP$ do {Evaluate candidate chromosomes’ fitness}
22:    if $f(Vcand_i) > f(V_i)$ then {Replace old population with fitter offspring}
23:      $V_i = Vcand_i$
24: return $V^+$ =0
```

The processes found in a Genetic Algorithm are governed mainly by the following control parameters: the number of chromosomes NP forming a population,

the number of generations I_{max} per run, the crossover probability p_{cr}, the mutation probability p_m introducing randomness during offspring generation, the crossover scheme used for producing offspring such as single point crossover, and finally the method for selecting parents per generation e.g. roulette wheel selection.
The available chromosomal pool NP determines how much diversity exists in the initial population. A randomly generated population may be diverse but not necessarily fit enough. Braun et al. [Braun et al., 2001] argue that providing a relatively fit initial population may improve offspring fitness after a moderate number of generations I_{max}. Mutating offspring occasionally may enhance diversity, possibly inserting genes that improve performance. However, more frequent mutations can lead to the opposite result. The crossover scheme applied, may affect fitness by shuffling genome sequences in a more flexible manner. Parent selection is important, as premature convergence to a seemingly fitter parental pool may exclude chromosomes that can otherwise contribute to successful evolution. Therefore, maintaining selection pressure through σ-scaling, is important for contributing to offspring variation.
Following OAT sampling described above, we found that the tuning parameters leading to best performance were: $I_{max} = 100$, $NP = 50$, $p_{cr} = 0.9$, $p_m = 0.003$ per bit and σ-scaling on.

4.1.9 sequential Genetic Algorithm

sGA (sequential Genetic Algorithm) is algorithmically identical to Algorithm 3. The core change is the search space passed to sGA, where sGA is now seeking for an optimal solution *per time-step* instead of tackling the full schedule timespan at once. When a solution has been generated for every time-step constituting a mission operations schedule, the final schedule is assembled by aggregating all individual solutions.
By dividing the problem's search space into sub-problems, we aim at observing the differences in fitness and processing time between a traditional GA and this sequential version, applied on a constrained problem like satellite mission operations scheduling.

Algorithm 4 sequential Genetic Algorithm

```
1: Initialise $I_{max}$, $NP$, $p_{cr}$, $p_m$, $T$ {$I_{max}$ = maximum number of generations, NP = population size (it must be even, due to algorithm design), $p_{cr}$ = crossover probability, $p_m$ = probability of mutation, $T$ = number of time-steps reflecting schedule timespan}
2: for $t = 1 : T$ do {For every time-step}
3:   Generate random population array $V$ comprising $NP$, $D$-long chromosomes {$D = n \cdot T$, the number of tasks $n$ times the number of time-steps $T$}
4:   while $I \leq I_{max}$ do
5:     for $i = 1 : NP$ do
6:       Evaluate candidate solution fitness, $f(V_i)$
7:       Apply $\sigma$-scaling on $f(V_i)$ {Increase selection pressure}
8:     Select $NP/2$ number of chromosomes $parents1$ from $V$, using SUS {Selection of one half of each mating pair}
9:     $parents2 = V - parents1$ {Other half of each pair. Used for selection "without replacement", each chromosome is selected once}
10:    $Vcand = [\ ]$
11:    for $i = 1 : NP/2$ do {Crossover}
12:      if $rand[0, 1) \leq p_{cr}$ then
13:        Randomly select crossover locus $loc = rand[2, \|parents1\|)$
14:        $child1_i = parents1_i[1 : loc/2] + parents2_i[loc/2 + 1 : end]$ {First offspring}
15:        $child2_i = parents2_i[1 : loc/2] + parents1_i[loc/2 + 1 : end]$ {Second offspring}
16:      for $j = 1 : D$ do {Mutation}
17:        if $rand[0, 1) \leq p_m$ then
18:          $child1_{i,j} = \textbf{\textit{not}}\ child1_{i,j}$
19:        if $rand[0, 1) \leq p_m$ then
20:          $child2_{i,j} = \textbf{\textit{not}}\ child2_{i,j}$
21:    $Vcand = Vcand + [child1,\ child2]$ {Merge children into a single candidate population}
22:    for $i = 1 : NP$ do {Evaluate candidate chromosomes' fitness}
23:      if $f(Vcand_i) > f(V_i)$ then {Replace old population with fitter offspring}
24:        $V_i = Vcand_i$
25:    $Vseq_t = V$ {Aggregate individual sub-schedules $V$ in $Vseq_t\ \forall\, t \in \{1 \cdots T\}$}
26: $V^+ = Vseq$
27: return $V^+$
```

The algorithmic setup is identical to GA. We used the same parameters for sGA. Time-constrained runs form an exception. In order to complete an 8 hour run, we divided 28800 – the number of seconds in 8 hours – with the number of time-steps That is, sGA will spend 400 seconds optimising each of the 72 time-steps comprising our 2.4 hours instance.

4.1.10 Greedy algorithm

Greedy was verified against our problem's constraints, adhering to power availability A_t per time-unit t. It is a simple unsophisticated algorithm that poses interest though, since it allows us to see how well can our problem be tackled by a simple Greedy approach.

Algorithm 5 Greedy algorithm

```
1: Set candidate solution $V$ equal to mission requirement $MR$
2: for $t = 1 : T$ do {For every time-step $t$ in schedule}
3:   for $k = 1 : n$ do {For every task $n$ in schedule}
4:     if $MR(k,t) = 0$ then
5:       $V(k,t) = 0$
6:     else if $P(V_t) \leq A_t$ then {If current power consumption $P(V_t)$ is less than or equal to available power $A_t$}
7:       $V(k,t) = 1$
8:     else
9:       $V(k,t) = 0$
10: return $V$
```

4.1.11 Simulated Annealing

A materials science-based stochastic optimisation algorithm described in Chapter 2.1, the Simulated Annealing (Code author: Joachim Vandekerckhove, University of California, Irvine, released under the 2-Clause BSD License) version below, was tested and verified to ensure it works in accordance with our problem's constraints.

Algorithm 6 Simulated Annealing algorithm

```
1: Initialise $temp_{init}$, $I_{max}$, $temp_{min}, \alpha$ {$temp_{init}$ = initial system temperature, $I_{max}$ = maximum number of iterations per temperature level, $temp_{min}$ = minimum system temperature, $0.0 < \alpha < 1.0$ = scale factor}
2: Generate initial bit string $V_0$ at random
3: while $temp > temp_{min}$ do
4:     I = 0
5:     temp = $temp_{init}$
6:     while $I \leq I_{max}$ do
7:         $V_{I+1}$ = Tweak($V_I$) {Perturb solution. $V_I$ is changed randomly from 1-bit to 5% of the vector's overall length.}
8:         $V_{I+1}$ = Trim($V_{I+1}$) {If $V_{I+1}$ violates the power constraint, cumulatively sum all scheduled tasks' power consumption, and trim the ones coming after power availability has been reached.}
9:         Evaluate new solution fitness $f(V_{I+1})$
10:        Evaluate old solution fitness $f(V_I)$
11:        if $f(V_{I+1}) > f(V_I)$ then
12:            Accept $V_{I+1}$ as new solution
13:        else if rand[0, 1) $< exp(-\frac{f(V_{I+1})-f(V_I)}{k \cdot temp})$ then
14:            Accept $V_{I+1}$ as new solution
15:    Decrease system temperature $temp = \alpha \cdot temp$
16: return $V^+$
```

Simulated Annealing and its sequential variant contain three main controls. The algorithm's cooling schedule (here $\alpha \cdot temp$) governing the current temperature "*temp*" per time instant, the minimum $temp_{min}$ and initial $temp_{init}$ temperatures respectively. Finally, the current solution selection probability $exp(\frac{-\Delta f}{k \cdot temp})$ contains the Boltzmann constant k that is used for scaling when fitness values have a larger magnitude. The algorithm can be iteration-constrained in every temperature level, setting an iteration limit I_{max}.
Simulated Annealing initially performs a random search, whereby the cooling rate within a temperature range ($temp_{init}$, $temp_{min}$) determines how fast will the algorithm start shifting towards pure hill-climbing. Selection probability $p = exp(\frac{-\Delta f}{k \cdot temp})$ contributes towards avoiding local optima by generating a solution acceptance probability (even if it has inferior fitness), which decreases with time as the "annealing" process progresses.
SA can be implemented either with multiple iterations performed per temperature

level [Skiena, 2008], or with a single iteration performed per temperature [Brownlee, 2011]. Skiena suggests that neither of those methods provide a clear advantage, as long as the total number of iterations performed is comparable. We opted for performing multiple iterations per temperature level.
Our OAT experiment showed that $I_{max} = 110,\ \alpha = 0.95,\ temp_{init} = 3,\ temp_{min} = 0.1,\ d_{max} = 0.05$ lead to best performance.

4.1.12 sequential Simulated Annealing

sSA (sequential Simulated Annealing) is roughly identical with Algorithm 7. The only difference is the search space passed to sSA, which is considerably smaller in size, attempting to “divide and conquer”. The algorithm therefore seeks for a global optimum *per time-step*, with a search space size of 2^n binary state perturbations for n tasks. Once all time-steps have been optimised, the final schedule of length T is reconstituted from all individual solution column vectors into a final solution matrix sized $n \cdot T$.
That way we aim to compare the performance of a traditional SA approach with that of a step-wise SA approach in terms of fitness and execution time. If sSA presents fitter solutions, then dividing the search space into significantly smaller quanta and aggregating the individual results may be a viable approach to consider when performing search optimisation on large search spaces.

Algorithm 7 sequential Simulated Annealing algorithm

```
1: Initialise $temp_{init}$, $I_{max}$, $temp_{min}, \alpha$ {$temp_{init}$ = initial system temperature, $I_{max}$ = maximum number of iterations per temperature level, $temp_{min}$ = minimum system temperature, $0.0 < \alpha < 1.0$ = scale factor}
2: for $t = 1 : T$ do {For every time-step}
3:   Generate initial bit string $V_0$ at random
4:   while $temp > temp_{min}$ do
5:     I = 0
6:     temp = $temp_{init}$
7:     while $I \leq I_{max}$ do
8:       $V_{I+1}$ = Tweak($V_I$) {Perturb solution. $V_I$ is changed randomly from 1-bit to 5% of the vector's overall length.}
9:       $V_{I+1}$ = Trim($V_{I+1}$) {If $V_{I+1}$ violates the power constraint, cumulatively sum all scheduled tasks' power consumption, and trim the ones coming after power availability has been reached.}
10:      Evaluate new solution fitness $f(V_{I+1})$
11:      Evaluate old solution fitness $f(V_I)$
12:      if $f(V_{I+1}) > f(V_I)$ then
13:        Accept $V_{I+1}$ as new solution $V$
14:      else if rand[0, 1) $< exp(-\frac{f(V_{I+1})-f(V_I)}{k \cdot temp})$ then
15:        Accept $V_{I+1}$ as new solution $V$
16:    Decrease system temperature $temp = \alpha \cdot temp$
17:  $Vseq_t = V$ {Collect individual sub-schedules $V$ in $Vseq_t$ $\forall\, t \in \{1 \cdots T\}$}
18: $V^+ = Vseq$
19: return $V^+$
```

The algorithmic setup remains the same as SA. As with sGA, in the time-constrained case sSA spends 400 seconds per time-step.

4.1.13 Local Search

This naïve LS algorithm attempted to complement the approximation methods we utilised, somewhat encouraging solution improvement. It was tested and verified to perform constrained optimisation on our problem. The algorithm generates a random feasible solution V_t per time-step and compares it to the existing solution V_t^{bst} generated by any of the algorithms in our experimental set. If $f(V_t) > f(V_t^{bst})$, V_t^{bst} is substituted by V_t. Otherwise, if both solutions are equally fit, one of the two is chosen randomly.

Algorithm 8 Local Search algorithm

```
1: for $n = 1 : n_{pass}$ do
2:   for $t = 1 : T$ do
3:     Randomly generate feasible bit string $V_t$ {$V_t$ is of length $n$, which is the number of tasks to be scheduled per time-step.}
4:     if $f(V_t) > f(V_t^{bst})$ then {Compare fitness of $V_t$ with the solution generated by optimisation algorithm $V_t^{bst}$}
5:       $V_t^{bst} = V_t$
6:     else if $f(V_t) = f(V_t^{bst})$ then
7:       $V_t^{bst} = rand(V_t, V_t^{bst})$
8: return $V^+$
```

Local Search requires one adjustment, the number of passes n_{pass} it performs in an attempt to improve V^{bst}. One pass corresponds to applying LS to all time-steps $\{1 \cdots T\}$ of the schedule. It is expected that the higher the n_{pass}, the higher the probability of schedule fitness improvement, as long as the user's time constraint permits it. We set $n_{pass} = 1$, attempting to keep processing time low, since the scheduling optimisation process would be scaled up 70-fold (7 days of operations).

4.1.14 Random Search

This describes a Random Search as seen in Algorithm 9 performing a fully stochastic exploration of the problem's search space.

Algorithm 9 Random search algorithm

```
1: for $I = 1 : I_{max}$ do
2:   Form solution vector $V_I$ at random
3:   Evaluate solution fitness $f(V_I)$
4: return $V^+$
```

Random search is a simple algorithm with minimal controls to set. Deciding the number of iterations I_{max} and possibly introducing more than one search agent NP for parallel processing, are the main parameters governing the algorithm's functionality.
Normally, Random search can produce better results when allowed enough time to explore. A high number of iterations I_{max} will most likely return fitter output. Likewise, if more than one search population members are used, they will most likely

search a larger portion of the search space, possibly finding a global optimum. Random Search does not employ more sophisticated mechanisms such as crossover and mutation or stigmergy, rendering it computationally faster. To give the algorithm a fairer opportunity to explore the search space, we set the number of iterations I_{max} to 60000, bringing the processing time to roughly 30 seconds.

4.2 Quantifying algorithmic performance

A comparison of all algorithms' performance in terms of fitness and execution time was made. All algorithms were tested under the same assumptions, with each test repeated for 50 runs for the same fixed number of iterations per run. Each metaheuristic was also tested using a time constraint, with each run spanning 8 hours. Time-constrained tests were repeated for 10 runs per algorithm, due to hardware restrictions. The reason for performing both iteration- and time- constrained runs was to establish and measure the gain achieved from running a given algorithm for a longer period of time. The chosen time constraint duration was one working day i.e. 8 hours.

4.2.1 Fitness and execution times

In this test case, six tasks need to be scheduled for a time period of 2.4 hours, with a resolution of 120 seconds per time-step, equalling 72 time-steps. The theoretical maximum fitness that can be achieved is $n \cdot T$ i.e. the product of the number of tasks to be scheduled times the number of time-steps considered. Realistically, the feasible maximum fitness is expected to be lower due to resource constraints. In this test case, the maximum theoretical fitness is 432. The 2.4 hours experiment – also called "small instance" – results are summarised in Table 4.1.
The hardware setup used for all experiments in this thesis comprised a 64-bit Intel Core i7 8-core @ 3.4GHz processor with 16Gb RAM running Ubuntu 12.02. The codebase was developed in MATLAB$^{©}$, to maintain compatibility with Strathclyde's ASCL (Advanced Space Concepts Laboratory) codebase, where the author of this thesis set the foundations of this work. Post-processing was performed using Jupyter Notebook [Kluyver et al., 2018], SciPy [Jones et al., 2001], Matplotlib [Hunter, 2007] and statistical analysis was performed with R [R Core Team, 2019].

In all the experiments contained in Tables 4.1, 4.2 and 4.3, there was one measurement variable (schedule fitness) and one nominal variable (algorithm). The former is considered a dependent variable, which is not normally distributed. The latter is an independent variable indicating the algorithm used per experiment. In

Table 4.1: Median algorithmic execution time (sec) and fitness. Best performance in **bold**.

Algorithm	Time (sec)	Fitness
ACS	142.7	348.0
DE	16.53	322.0
GA	97.66	368.5
GREEDY	0.04	240.0
sGA	395.75	**374.0**
Rand	27.42	265.0
SA	3.06	355.5
sSA	1.97	**374.0**

Table 4.2: Median algorithmic execution time (sec) and fitness when paired with Local Search. Best performance in **bold**.

Algorithm	Time (sec)	Fitness
ACS +LS	145.3	349.5
DE +LS	16.46	328.0
GA +LS	120.8	370.0
GREEDY +LS	0.06	262.0
sGA +LS	435.94	**374.0**
Rand +LS	28.24	284.0
SA +LS	3.09	358.0
sSA +LS	1.96	**374.0**

this instance, we have several measurements of the same dependent variable taken using different algorithms. The normality assumption of repeated measures ANOVA has not been met, thus we will apply the Friedman test. If our statistical analysis is significant i.e. it rejects the null hypothesis that all sample medians are the same, and since there are more than three samples compared, we will perform a post-hoc test [Lund Research Ltd., 2012; Marshall and Marquier, 2014].

A Friedman test for significance was performed on the iteration-constrained cases (Tables 4.1 and 4.2) and a separate one on the 8 hour time-constrained case (Table 4.3). In the iteration-constrained cases summarised in Figure 4.23, the Friedman test indicated that there are differences between the median ranks among the sixteen experiments we conducted, $\chi^2(15, N = 50) = 738.49$, $p < 0.001$. A Nemenyi post-hoc test showed that schedules generated by sGA and sGA +LS are significantly fitter than other algorithms' schedules, with the exception of sSA and sSA +LS. We also see that GREEDY and GREEDY +LS generated the least fit schedules in this experiment, but not significantly less fit than RAND and RAND +LS. For the full

Table 4.3: Median algorithmic fitness for an 8 hour time-constrained run. Best performance in **bold**.

Algorithm	Fitness
ACS 8hrs +LS	372.0
DE 8hrs +LS	366.5
GA 8hrs +LS	**377.0**
sGA 8hrs +LS	374.0
Rand 8hrs +LS	293.0
SA 8hrs +LS	372.0
sSA 8hrs +LS	374.0

post-hoc test results, see Appendix G.

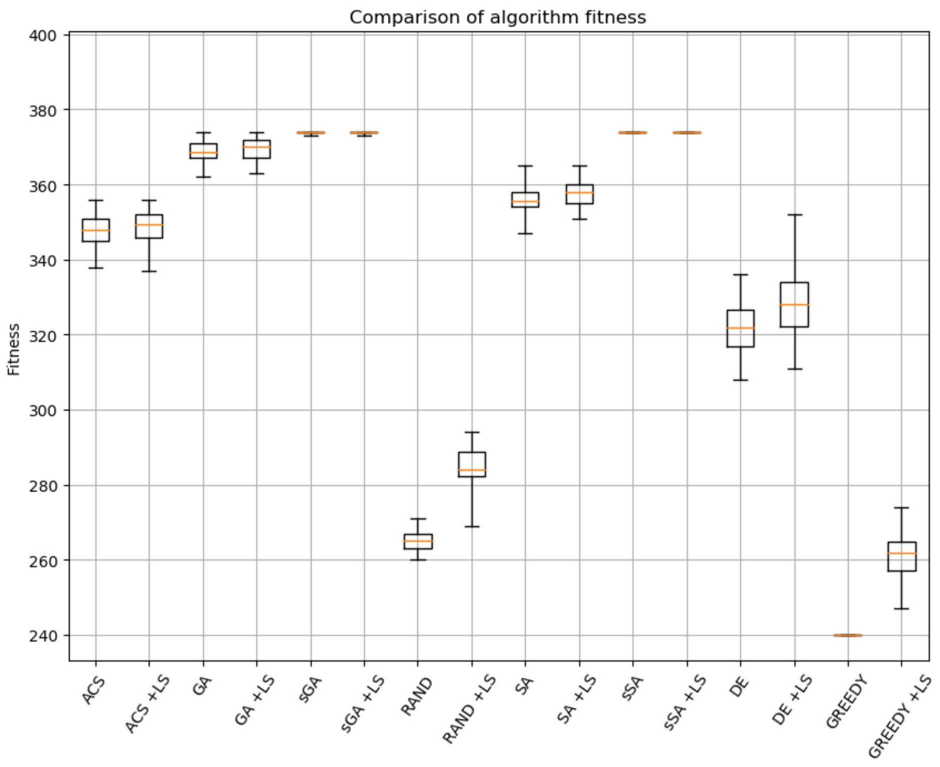


Figure 4.23: Boxplots with whiskers from minimum to maximum of fitness for all iteration-constrained algorithms

Testing the time-constrained case (summary in Figure 4.24) for significance, the Friedman test showed that there are differences between the seven experiments' median ranks, $\chi^2(6, N = 10) = 50, p < 0.001$. Performing a Nemenyi post-hoc analysis, we see that schedules created by GA were significantly fitter than ACS, DE and RAND ones. And also that RAND generated schedules were significantly less

fit than SA, sGA and sSA.
We notice that longer runs generally lead to better fitness with one exception. When an algorithm tackles the problem on a time-step basis, like sSA and sGA do, they both converge to an equally fit global optimum irrespective of the time spent exploring. Complementing our algorithmic set with a naïve Local Search did occasionally somewhat improve solution fitness.
Our experiments showed that when GA is allowed to run for 8 hours and accompanied by LS, it converges closer to the global optimum but not significantly closer than SA 8hrs +LS, sGA 8hrs +LS and sSA 8hrs +LS. Overall, we notice that allowing algorithms to run for longer does present small improvements but not considerable given the 8 hour time limit – which was not always welcomed by mission operations scheduling experts anyway.

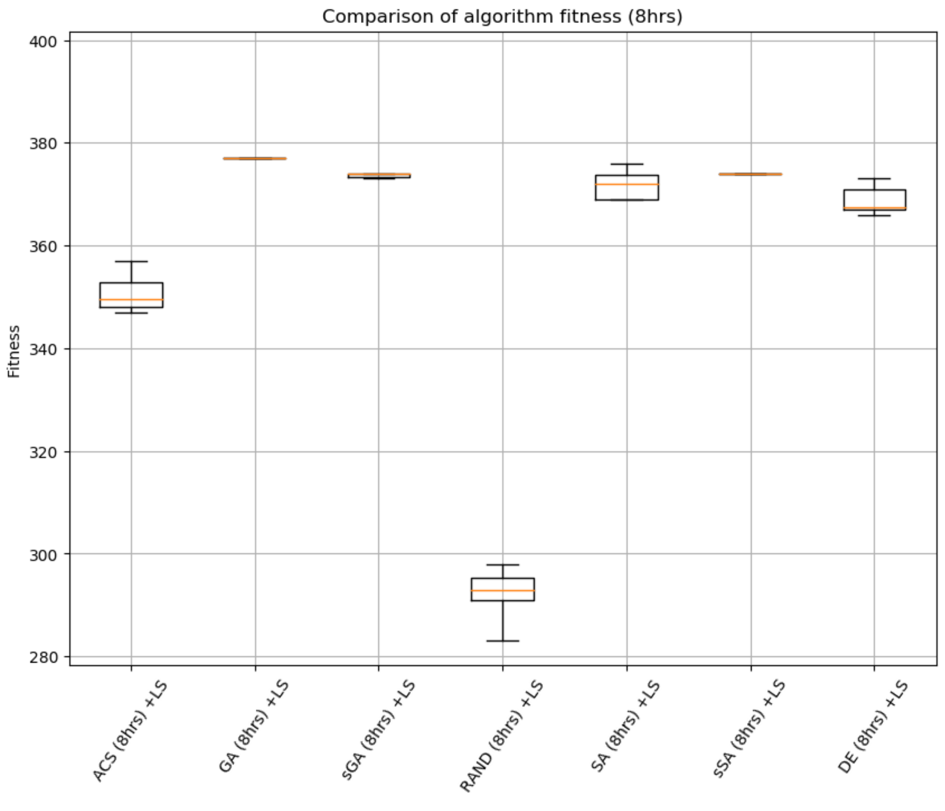


Figure 4.24: Boxplots with whiskers from minimum to maximum of fitness for all time-constrained algorithms with Local Search

It is possibly the evolutionary mechanism behind GA (selection, crossover, mutation) combined with the problem size, that allow the algorithm to naturally converge to fitter offspring when given enough generations and a large enough pop-

ulation. When allowed to evolve for a longer period of time (time-constrained run), the algorithm returns a slightly fitter solution. Given the length of time GA has been allowed to explore, the observed improvement is minimal leading us to think that long time-constrained runs are not much more beneficial for a moderately sized problem like this one.
sGA, utilising the aforementioned mechanisms and tackling a smaller search space, seems to converge to a solution of identical fitness irrespective of the run's ending criterion (iterations, time) or use of LS or not. This perhaps implies that when an algorithm has a more short-sighted focus on this problem instance, it can potentially converge to a global optimum. Interestingly, we notice that sSA also converges to an equally fit solution. Which might be attributed to the sequential myopic approach, irrespective of the algorithm's strategy employed.
Note that applying a simple GREEDY approach did not yield fit enough schedules, even when compared to RAND. While there are more powerful Greedy techniques such as Squeeky Wheel Optimisation [Clements and Joslin, 2011] which has been successfully applied to fibre-optic production line scheduling, we believe that the landscape of our problem does not lend itself to a Greedy approach. Likewise, utilising RAND is clearly not a viable solution for such a dynamic and challenging problem.
DE being a continuous optimisation method by design, employing evolutionary concepts like mutation and crossover, performed roughly on par with most algorithms except from ACS and significantly better than RAND.
ACS is an algorithm originally designed for graph traversal, relying on swarm intelligence and stigmergy for information sharing. For that reason, its adaptation to this domain was somewhat graceless. Nevertheless it generated relatively fit schedules overall, but not significantly better than the ones constructed by DE and SA.
Finally, SA relying on metal annealing combines local search with hill-climbing in moderation, but does not rely on gradients. It generated fit schedules overall but not significantly fitter than GA.
Overall, we notice that the sequential approach is consistent and does very well when we want to generate a solution quickly. If the practitioner wants to find a better solution for this problem instance, GA (8hrs) +LS is the best choice.

4.3 Conclusions

In this chapter we introduced the problem this thesis is attempting to address and the tools used for solving it. A formulation of this engineering problem, shaped after interacting with industry, was described. Algorithmic behaviour, when constructing the same satellite schedule, was observed on the basis of generated schedule fitness and execution time elapsed. For comparing our results, we utilised the Median for central tendency, IQR for dispersion, non-parametric statistical analysis and post-hoc testing for discerning the best performing algorithm(s).

A quantitative experiment was set up comprising a number of iteration- and time-constrained test cases. Namely: 50 runs of pure algorithms ACS, DE, GA, RAND, SA, sSA, sGA (with GREEDY used for comparison) and 50 runs of the aforementioned algorithms combined with a naïve Local Search (choosing randomly between equally fit solutions) ACS +LS, DE +LS, GA +LS, RAND +LS, SA +LS, sSA +LS, sGA +LS (GREEDY +LS for comparison) ran for a specific number of iterations. Also 10 runs (due to hardware constraints) of ACS (8hrs) +LS, DE (8hrs) +LS, GA (8hrs) +LS, Rand (8hrs) +LS, SA (8hrs) +LS, sSA (8hrs) +LS and sGA (8hrs) +LS with all iterative algorithms ran for 8 hours.

We observed that time-constrained multi-agent metaheuristics performed best usually, but only by a small fitness margin, followed by iteration-constrained multi-agent sequentially applied metaheuristics coupled with Local Search and finally pure multi-agent metaheuristics. Algorithmic setup was discussed, mentioning how many parameters does the practitioner need to take into consideration and their influence in algorithmic performance.

In the next chapter we will run the above algorithmic set on a larger search space representing a real-life inspired scenario, to observe how does each algorithm scale up to this larger case. Do they all tackle larger search spaces comparably well? Will their behaviour change at all?

"I never am really satisfied that I understand anything; because, understand it well as I may, my comprehension can only be an infinitesimal fraction of all I want to understand about the many connections and relations which occur to me, how the matter in question was first thought of or arrived at, etc., etc."

– Augusta Ada King, Mathematician

5 Earth Observation operations test case

In the previous chapter we observed how various algorithms tackled a scheduling problem of moderate size using three different approaches. Namely, iteration-constrained algorithms coupled with Local Search or solo and time-constrained algorithms coupled with Local Search. Statistical analysis allowed us to make inferences on algorithmic fitness performance for this problem instance, also observing how execution time varies. The mechanisms powering every method in our algorithmic set of choice are unalike, with some relying on random choice and others relying on shared experience for instance. Consequently, different strategies approached a given problem from a different angle.
Most of the bibliography referenced in this thesis supports the choice of an algorithm over others to a variable degree, usually citing the seminal work that introduced a novel algorithm to the Artificial Intelligence community along with an argument as to why this algorithm is fit for addressing the problem. Normally, original publications illustrating the workings of a new algorithm contain some form of quantitative analysis to demonstrate the potential of this strategy. It is quite common – and sensible – to see algorithmic quantitative analysis performed using standard moderately sized benchmarks of familiar shape and size, with a known global optimum (or optima if the function is multi-modal). Nonetheless, it is less common to find publications analysing and comparing algorithmic performance for mission operations

scheduling. We observe this in relevant literature, some examples include ESA's MrSPOCK [Cesta et al., 2011] applying GA, AIMS [Pralet and Verfaillie, 2009] employing local search, STScI's SPIKE [Giuliano, 2014] using a multistart stochastic repair heuristic, NASA's SAMPLE [Dupnick and Wiggins, 1980b] applying a greedy algorithm, and individual researchers such as Sun and Chester [Sun and Chester, 2010] utilising GA to address their scheduling problem. We notice that one trend emerging is the infrequency of comparison and analysis of algorithms for solving mission operations problems. Inevitably, a pertinent question arises: do algorithms applied to mission operations scheduling, that work well for smaller instances, also work for larger real world instances?

Attempting to address this question, we applied the algorithmic set used in Chapter 4 onto a larger realistic Short Term Plan inspired by Earth Observation mission operations. This chapter examines each algorithm's performance when solving an operations scheduling problem of realistic length, addressing the question of algorithmic scalability.

5.1 Algorithmic scalability

Eight algorithms were put to the test, solving two scheduling scenarios on a desktop computer of average performance[1]. Due to time and hardware constraints, each algorithm per problem instance was run 5 times. Little fitness variance was exhibited. We applied two operations' time spans, a 7 days long mission operations schedule reflecting the generally used Short Term Plan, and a 1 day schedule acting as an intermediate step between the 2.4 hours instance and the 7 days one. In both problem instances all algorithms were run for 8 hours, corresponding to one working day or an overnight calculation duration.

A set of 25 tasks were considered per time-step, with each time-step representing 30 seconds of schedule time, presenting a good trade-off between temporal resolution and problem size. The 1 day long schedule solution vector spans 72000 bits (25 tasks times 2880 time-steps) while a 7 day long schedule vector spans 504000 bits (25 tasks times 20160 time-steps). Each time-step in our search space comprises about 33.5 million nodes (2^{25}), representing all possible bit strings corresponding to setting the 25 tasks at hand. This translates to a practically infinite search space, due to combinatorial explosion.

As mentioned in Chapter 4, an OAT sampling experiment was performed for algorithmic parameter tuning, quantifying each control parameter's influence over

[1]An 8-core Intel Core$^{\text{TM}}$ i7 CPU @3.4GHz with 16Gb RAM running Ubuntu 12.04

every algorithm's output fitness. Once a satisfactory setup per algorithm was established, it was used consistently throughout all tests of Chapters 4 and 5 for uniformity of comparison.

5.1.1 Realistic Short Term Plan

The Solar System Science Operations Division of ESA's Research and Science Support Department shared with us a technical note (Appendix B), describing at a high level the process of the PROBA-2 Earth Observation mission's science operations. This document, supplemented with facts from the informative eoPortal PROBA-2 article [Kramer, 2002], allowed us to approximate the mission's technology and objectives.

Figure 5.1: Artist's view of the deployed PROBA-2 spacecraft (image credit: ESA)

Using the mission's continually updated ancillary data [Science Center (P2SC), 2009], the launch date was set as well as the starting and ending operations dates of interest. PROBA-2 was launched on *November 2nd 2009, 01:20:00 UTC*. This is $t = 0$ for each schedule generated. Every other activity recorded in the schedule is added to the time-line based on the launch date. In practice there is a delay between satellite launch and operational phase commencement, including preparation and tests to ensure the satellite's orbit and health. Since this phase varies depending on each mission, we did not include it in this work. Our Short Term Plan instances of choice span the *13 - 14th of March 2013* as well as the week of *13 - 19th of March 2013*. The chosen ground stations for controlling and communicating with the satellite are ESA Redu [Redu Space Services, 2007] and Kongsberg SvalSat [Kongsberg Satellite Services, 1997]. Using SPICE [Acton, 1996] (initially introduced in Chapter 3), visibility windows per ground station are generated. A constraint of 9 minutes minimum visibility duration per ground station was applied to ensure ground station visibility windows are adequately long to host meaningful communications. Once valid ground station visibility windows are established, they are passed to the mission requirements vector. During those windows, the TTC system is utilised for communications and commanding.

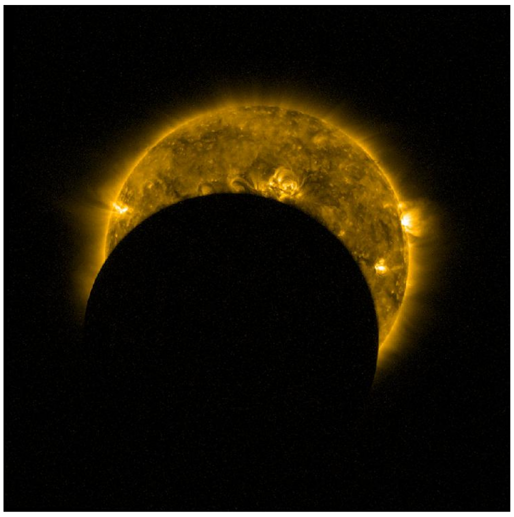


Figure 5.2: Partial eclipse taken in UV by PROBA-2's SWAP, recording the Sun's turbulent surface and its swirling corona (image credit: ESA, Royal Observatory of Belgium)

The mission requirements vector is then complemented by all system and payload requirements found in Chapter 3, inspired by PROBA-2. At the same time, the satellite's orbital characteristics and incident solar radiation allow for a power profile to be determined for the given time frame. Once all mission requirements have been set and resource availability has been calculated, the optimiser has to its disposal all the information it needs to generate a schedule.

Overall the aforementioned problem instances contain most building blocks found in scientific space missions, such as number and frequency of tasks involved, length of schedule, time resolution, resource availability and resource constraints. Whenever it wasn't possible to gain access to more PROBA-2 specific mission details, information gaps were filled in by using general Earth Observation mission facts from space engineering handbooks and technical reports. For example, the PROBA2 TTC amplifier noise was not readily available in published technical reports, thus we consulted a handbook on satellite communications [Roddy, 2006] for a representative value. Similarly, PROBA2 ADC gyroscope drift rate was taken from a seminal space systems engineering handbook [Wertz and Larson, 1999], and also a generally useful space engineering resource we consulted is NASA's technical reports server [National Aeronautics and Space Administration, 2014].

As described in Chapter 3.2.3, the current chapter's problem instances contain more tasks, the time-step grain is finer and the satellite's orbit differs in inclination and periodicity. In comparison, the problem instance used in Chapter 4 comprises fewer tasks, a coarser time-step grain, a longer orbit period and different ground stations.

5.1.2 Algorithmic setup

All algorithmic tuning parameters remain the same as in Chapter 4, excluding the sequential versions i.e. sGA and sSA. They form an exception, since time spent per time-step depends on the overall length of the schedule. If sGA was to be run with the parameters chosen in Chapter 4.1.3, it would run for 22.7 seconds per time-step – roughly 127.1 hours (5.2 days) for a 7 days schedule – which exceeds the 2 day schedule refinement cycle applied to STP operations. Similarly, sSA would run for 0.2 seconds per time-step – 1.1 hours for a 7 day schedule – which is well below the 8 hours limit we have set. Since we wanted to compare the performance of all algorithms over an 8 hour run, some modification was needed.
In this chapter, the instances examined have a length of 2880 and 20160 time-steps for the 1 day and the 7 days case respectively. In order to complete an 8 hour run, we divided 28800 – which is the number of seconds in 8 hours – with the number of time-steps. That is, in the 1 day test case sequential algorithms will run for 10 seconds per time-step, and in the 7 days case those algorithms will run for 1.43 seconds per time step. Care should be taken when applying sequential optimisation techniques on finer grained and longer schedules, as less time will need to be spent optimising each time-step.

5.2 Observing algorithmic scalability

Using the above setup, we quantitatively analysed each algorithm's behaviour. Table 5.1 containing the 1 day schedule fitness and Table 5.2 including the 7 days schedule fitness summarise each algorithm's median fitness over 5 runs per case, providing a small sample to draw conclusions from while keeping the required computational effort feasible for our desktop setup.
As mentioned previously in Chapter 2.1, we utilised the median μ as the measure of central tendency and the interquartile range *iqr* as the measure of dispersion, since our resulting statistical distributions were not guaranteed to be Gaussian due to the dynamic nature[2] of our search space.
During this process, it became obvious that one of our algorithms of choice could

[2]Satellite scheduling presents an element of unpredictability, since requirements can change per orbit, potentially rendering the problem search space multi-modal, discontinuous etc.

not scale up due to an intrinsic design characteristic. ACS, originally designed as a graph traversal algorithm, initialises a global pheromone matrix (known as a distance matrix in graph theory) at the very beginning of the algorithmic run. If the graph contains N elements, the distance matrix will be of size N^2. Taking as an example a moderate test case of 1 day of satellite operations, with 25 tasks to be scheduled over 2880 time-steps, the global pheromone matrix becomes prohibitively large. That is, almost 100 billion (96636764160) bits.

Single day schedule generation

Table 5.1: Algorithmic fitness for the 1 day mission operations schedule. Best performance per category in **bold**.

Algorithm	Fitness
DE +LS 1 day	51125.0
GA +LS 1 day	48209.0
sGA +LS 1 day	70996.0
RAND +LS 1 day	46420.0
SA +LS 1 day	51712.0
sSA +LS 1 day	**71242.0**
GREEDY +LS 1 day	66784.5

Applying the Friedman test on the 1 day case showed that at least one algorithm presents significantly different fitness from the rest, $\chi^2(6, \text{N} = 5) = 30$, $\text{p} < 0.001$. By observation of boxplots in Figure 5.3, we see that both sSA and sGA are strong contenders in the 1 day instance, with sSA returning the highest fitness value for this instance as seen in Table 5.1. By observation of the boxplots in Figure 5.3, and using GREEDY for comparison, we see that the algorithm does not converge close enough to a global optimum but it does better than DE, GA and SA. This is perhaps an indication that the size of the problem, given our problem formulation, is becoming too large for those metaheuristics without applying significant modification. We notice though that the sequential versions of GA (sGA) and SA (sSA) perform best, since the solution vector size they evolve is considerably more manageable. RAND generates the least fit solution, which is to be expected from a simplistic random approach.

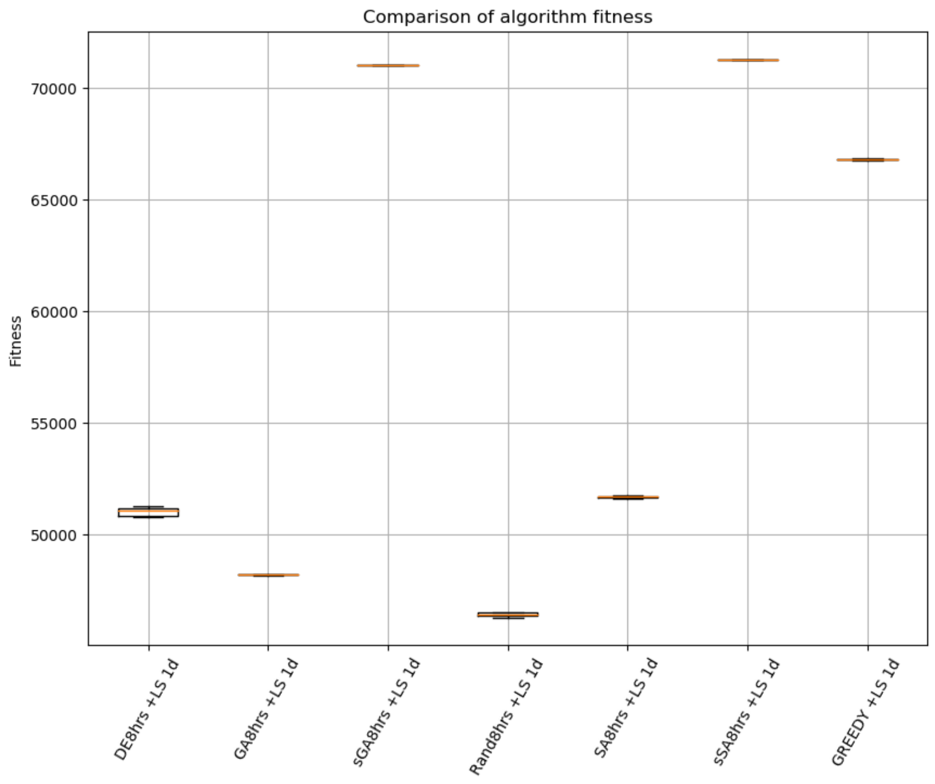


Figure 5.3: Boxplots with whiskers from minimum to maximum of fitness for 1 day-long operations

Week long schedule generation

Table 5.2: Algorithmic fitness for the 7 days mission operations schedule. Best performance per category in **bold**.

Algorithm	Fitness
DE +LS 7days	332198.0
GA +LS 7 days	324978.0
sGA +LS 7 days	454621.0
RAND +LS 7 days	319947.0
SA +LS 7 days	327989.0
sSA +LS 7 days	**495370.0**
GREEDY +LS 7 days	377666.0 .0

Testing the 7 days case for statistical significance using the Friedman test, we find that at least one algorithmic fitness median differs from the rest, χ^2(6, N = 5) = 30, p < 0.001. By observation of the boxplots in Figure 5.4 below, we see that sSA produced a fitter schedule. In this instance, we notice that the sGA generated schedule fitness is no longer as close to the sSA's solution fitness. Also, we see GREEDY presenting a fairly inferior solution but still fitter than DE, GA, SA and RAND. As mentioned above, we reckon that this can be attributed to the sheer size of our search space, which is difficult to tackle effectively without significant modifications. Unsurprisingly, RAND produced the least fit schedule.

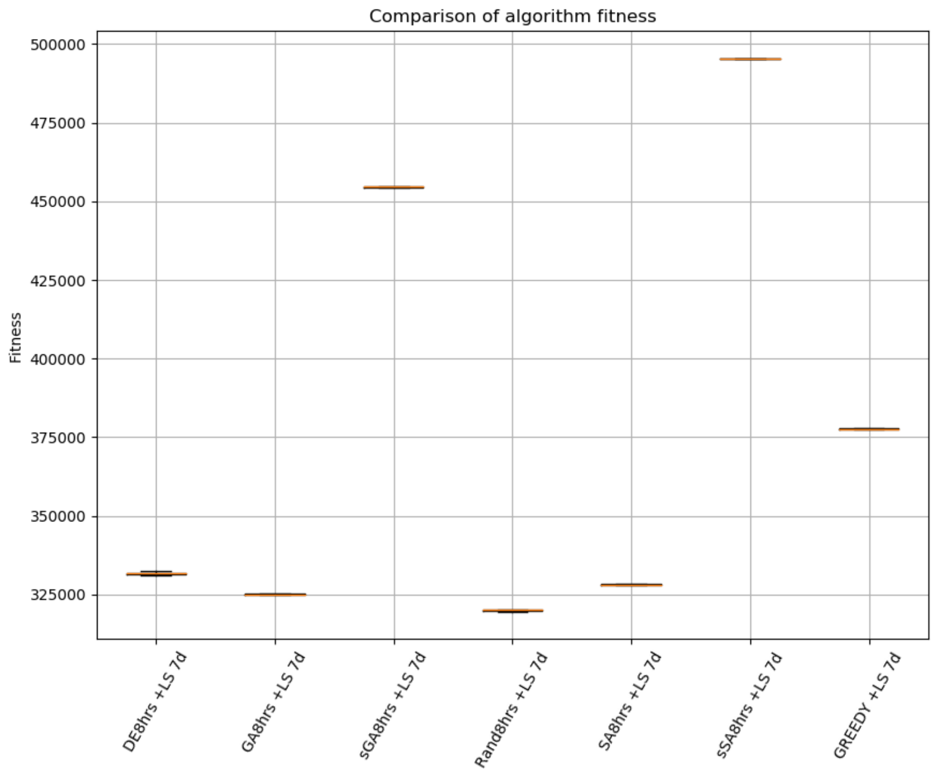


Figure 5.4: Boxplot with whiskers from minimum to maximum of fitness for 7 day-long operations

5.3 Conclusions

In this chapter, we looked at how our algorithm set scales when tackling a large instance of the problem at hand. We employed the same algorithmic set, as seen in Chapter 4, to generate mission operation schedules for a 1 day mission duration and a week's worth of satellite operations – which corresponds to a Short Term Plan. Algorithms were run for a time limit of 8 hours, using the same parameters as in the previous chapter with the exception of sequential algorithms. Depending on the problem instance, sSA and sGA were ran for an appropriate duration per time-step that cumulatively sums up to 8 hours per run.

It became obvious from the start that ACS was not able to tackle large search spaces. The reason was its intrinsic feature of global pheromone update on a distance matrix with a span equal to the square of the total number of bits of the original solution length. Furthermore, we observed that a simple Greedy approach produces a relatively fit solution in short elapsed time, but cannot optimise this solution any further. We also noticed that a simple Random approach is too naïve for generating a noteworthy operations schedule, especially for larger lengths of time e.g. 7 days.

Metaheuristics – applied in their standard form – consistently presented poor performance both in the 1 day and 7 days instances, likely because of the extensive size of each case's search space given the current problem expression. In the 1 day schedule instance, we see that SA and DE present comparable solution fitness with GA generating less fit schedules and Rand generating the least fit result. Greedy generated a reasonably fit solution. However, the strongest contenders are both sequential algorithms used i.e. sSA and sGA, both of which generated the fittest solutions for this instance.

Comparably, in the 7 days instance we notice a similar trend. Namely, Random Search generates the least fit solution, DE and SA present a solution with somewhat higher fitness but only marginally. Greedy generates a solution of mediocre fitness. In this case we see that sGA cannot keep up with sSA, since sGA is not given enough time to converge for a 25 bit chromosome, rendering sSA the algorithm generating the highest fitness schedules.

In the following and final chapter we will summarise our contributions, offering an overview of what was the problem, how we attempted to tackle it and what are our findings.

"Education is what remains after we forget what we're taught"

– Albert Einstein

6 Conclusions

To summarise

This work revolved around four points: i) study small scale satellites, and study the mission operations scheduling problem, ii) make an attempt at formalising the problem, iii) apply a set of search algorithms in an effort to optimise schedules, and iv) observe if and how well do these algorithms scale up for solving real-sized problems.

Throughout this process, we noticed how important are design decisions such as the problem expression, algorithmic setup, fitness function definition and search space size. It was also evident that undertaking applied research adds to the challenge, as the problem at hand is tangled and less well defined, reflecting the intricacies of an engineering problem from the aerospace and defence domain.

From an applied Computer Scientist's point of view, we see approximation methods as a collection of potent tools that empower practitioners to achieve increased efficiency in diverse engineering applications such as operations scheduling, physical design, manufacturing among others. However, time and effort need to be invested in algorithmic experimentation in order to find the most well fitting algorithm for the problem at hand, before committing to one.

6.1 Conclusions

This thesis attempts to address three questions in logical order: a) What is the problem, b) how did we attempt to address it, c) what did we discern through our experimental work. We hereby address those questions:

6.1.1 What is the problem?

Following a satellite launch, operations and their scheduling component are what ensures mission success. The satellite needs to be commanded, frequently with sub-second accuracy, to perform specific tasks during appropriate time periods. This is a challenging task, given the rigid technological constraints imposed combined with a satellite's perpetual motion and the physical constraints this brings. Therefore, in order to successfully run a space mission, a set of distributed teams need to effectively communicate their subsystems' or payloads' mission requirements to the satellite's operations centre. There, requirements are laid out into a common timeline, compared against constraints and the job of the operations scheduling experts begins.
Operations engineers normally have to devise three types of what is called a plan in Space Engineering, what we call a schedule in Computer Science. They start from a higher level 3-6 month long schedule (LTP), describing roughly what the next mission steps are going to include. This schedule is then somewhat elaborated, describing in more detail the objectives of the upcoming month (MTP). Finally, a week-long schedule is constructed in as much detail as possible (STP), tested for conflicts and uplinked to the satellite for execution.
Interestingly, thus far operations scheduling has been mainly performed using office productivity tools such as Microsoft Project and Excel, occasionally complemented by in-house variants of a Project-like interface. There is a corpus of research on the topic of automating and optimising satellite operations scheduling, outlined in Chapter 2.1. Prominent researchers from Space Agencies, national Aerospace laboratories and research institutions have attempted to address this problem using various algorithmic methods. Nevertheless, the choice of algorithm is rarely adequately supported by evidence in this particular problem domain. By comparing a number of algorithms on a small satellite mission operations scheduling instance in Chapter 4, we inferred which algorithm performs best for the given problem expression. A corollary question that arose was how well do those algorithms scale up, for solving a real-life sized problem instance. Chapter 5 attempts to answer this question, by observing each algorithm's response and indicating the best contender.

6.1.2 How was it addressed

In order to decide which algorithm would work better for this engineering application, a practitioner needs to compare a set of techniques from different classes to maximise the chance of utilising an appropriate technique for the present test case. We tried to address this question by using eight algorithms from different categories, used with a naïve Local Search and without it. This algorithmic set was applied to a small problem instance, using an iteration constraint derived from performing sensitivity analysis per algorithm, and a time constraint amounting to one work day. Algorithm selection was done based on method popularity such as GA or algorithmic class e.g. swarm intelligence.

Initially, a low fidelity numerical model of a small satellite was developed, approximating a typical small (roughly 100kg) LEO satellite's main functions. Those include orbit propagation, target visibility, telecommunications, power generation and consumption, data processing, attitude control and some generic scientific instruments. The satellite model outputs mission requirements and constraints, which are in turn passed to each algorithm for schedule generation.

Then, three LEO timelines of varying lengths were generated. Namely, a smaller instance corresponding to 2.4 hours of operations with a grain of 120 seconds per time-step and 6 tasks to schedule, as seen in Chapter 4. As well as more realistic instances spanning 1 day and 7 days of operations, with 30 seconds per time-step and 25 tasks to schedule, as used in Chapter 5. A fourth instance spanning 89 minutes with 120 seconds per time-step grain and 4 tasks was put together, for algorithmic tuning purposes. Note that the risk of overfitting the current algorithmic setup to this particular problem instance is real, since it was very difficult to find more instances for tuning purposes. A compromise had to be reached between fitness and execution time, in an attempt to balance performance with scalability. Thus, algorithms were tuned to perform satisfactorily but not necessarily optimally.

Finally, we employed ACS, DE, GA, SA, Random Search, a simple Greedy algorithm and a sequential variant of GA (sGA) and SA (sSA) with and without a simple Local Search. Sequential algorithms are generally identical to their original version with one exception. The search space passed to them corresponds to a single time-step instead of the full schedule timeline, thus an optimisation run comprises optimising each time-step separately. The sum of the fitness of each individual time-step's solution makes up the total schedule fitness.

Each individual experiment of the 2.4 hours instance was run 50 times with and without Local Search using an iteration constraint, and 10 times with Local Search using an 8 hour time constraint. For longer timelines (1 day, 7 days) a sample of 5 runs was generated per algorithm, where all algorithms were run for 8 hours with Local Search. To infer which algorithm is the strongest contender in Chapter 4, we tested for significance using the Friedman method and performed a post-hoc test. In Chapter 5, we applied the Friedman test to test for significance. Since the sample size was smaller, by observation of the boxplots we inferred the best candidate.

6.1.3 Findings

In the small instance spanning 2.4 hours, studied in Chapter 4, we notice that in the iteration-constrained experiments sequential methods generated significantly fitter schedules. We also see that the least fit schedules were constructed by GREEDY and RAND. Furthermore, a slight increase in fitness is usually observed when LS is applied, indicating a small positive contribution.
When algorithms are allowed to run for 8 hours, a small increase in fitness is observed, but not in the sequential algorithms' case. Sequential algorithms seem to converge to the same global optimum irrespective of the time they are allowed to run for. Overall, GA 8hrs +LS generated significantly fitter solutions than ACS 8hrs +LS and DE 8hrs +LS while RAND 8hrs +LS generated the least fit solutions out of our algorithmic set. This could be attributed to GA's global overview of the problem, compared to sequential algorithms that have a more narrow per time-step view. So if one wants to maximise the probability of optimality, they should be prepared to run the algorithm for longer.
Scaling up to larger instances, examined in Chapter 5, we observe a somewhat different landscape. When attempting to solve the 1 day scheduling instance, we see that non-sequential methods such as GA, DE or SA struggle to converge closer to a good optimum, while GREEDY performs better, contrary to Chapter 4 where GA 8hrs +LS was the strongest contender and GREEDY the weakest. However, sequential algorithms have the lead, with sSA returning the highest fitness by a small margin, in the 1 day schedule generation experiment.
Solving the 7 day schedule instance, which represents a real life STP time-span, the ranking changes again. This time, sGA 8hrs +LS generates less fit solutions leaving a larger gap between itself and sSA 8hrs +LS which generates the fittest schedule. GREEDY can no longer keep up with the problem size, exhibiting a discernible fitness decrease. The rest of our algorithmic set present substandard solutions, since the problem size – and hence the search space for these algorithms – has grown considerably.
Generally, we see that LS somewhat contributes to improving the fitness of a smaller instance (Chapter 4), but is not useful in the larger instances of Chapter 5. For a

Local Search to potentially contribute to larger instances, a more tailored approach will be needed, to address the problem's size and idiosyncrasies.
An important lesson learned has been the significance of investing more time to develop a problem expression that will encapsulate well enough the problem at hand, possibly reducing it into a well-known problem from the combinatorial optimisation domain. This would generate a smaller, more manageable search space, leading to more efficient and effective computation.
Finally, developing a descriptive enough fitness function for this real-life case poses a challenge, as the problem is convoluted reflecting the intricacies of an engineering application from the aerospace and defence domain.

6.1.4 Future work

This work can be expanded in different directions depending on each practitioner's interests. We are particularly interested in applying approximation methods to large real-life scheduling problems in the space domain but other engineering areas too, thus we would like to expand this work in three dimensions:
On the one hand we would like to see the development of a more complete fitness function for the problem at hand, incorporating many of the important decision criteria used in real-life mission operations such as cost minimisation, on-board memory maximisation, best target observation opportunity etc. Developing such a fitness function can be a lengthy process incorporating technical knowledge and management principles as well as a statistically significant sample of human experts expressing their views and practices. It might be useful to consider Machine Learning techniques combined with past mission data which will allow us to learn objectively from previous experience and generate a satisfactory problem expression. Furthermore, utilisation of formal Knowledge Engineering practices would complement the process well, allowing us to achieve a well rounded result.
On the other hand we'd like to extend this work to include more approximation algorithms, enriching it to provide a reference for this constrained scheduling problem expression. Given the nature and size of this problem, we believe that approximation methods can tackle effectively both the dynamic nature of mission operations and the size of an average schedule if allowed to utilise enough resources. Since satellite mission operations constitutes a particular niche, we believe that with enough care taken towards the development of a flexible enough generalised complex operations problem expression, this work will be applicable to a wider area of scheduling applications such as logistics or transport for instance. Likewise, a flexible enough fitness function can incorporate problem agnostic elements such as cost, capacity,

precedence among others, which are commonly found in different processes.
The third dimension is the use of meta-optimisation [Grefenstette, 1986; Birattari et al., 2002b; Nannen and Eiben, 2006] to improve the performance of all approximation methods included in this thesis. This will allow us to quantify the improvement achieved through automating algorithmic control parameter selection, compare it with the computational and cognitive effort expended and conclude whether meta-optimisation renders large enough performance gains to support the extra resources required for this process.

Bibliography

C. H. Acton. Ancillary data services of NASA's Navigation and Ancillary Information Facility. *Planetary and Space Science*, 44:65–70, 1996. doi: 10.1016/0032-0633(95)00107-7.

C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In I. P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009*, pages 142–157, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04244-7.

A. Arcuri and G. Fraser. Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, 18: 594–623, 2013. doi: 10.1007/s10664-013-9249-9.

J. E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In *Proc. of the 2nd Intl Conf on GA, isbn = 0-8058-0158-8, pages = 14-21, key = baker-1987a*. Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1987.

C. A. Balanis. *Antenna Theory Analysis and Design*. Wiley, 2005. ISBN 978-0471667827.

S. Baluja. An Empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics,. Technical report, September 1995.

J. Rodriguez Banga and J.J. Casares Long. Integrated Controlled Random Search: Application to a wastewater treatment plant model. In *IChemE Symposium on Process Optimisation*, 1987.

L. Barbulescu, A. E. Howe, J. P. Watson, and L. D. Whitley. Satellite range scheduling: A comparison of genetic, heuristic and local search. In Juan Julián Merelo Guervós, Panagiotis Adamidis, Hans-Georg Beyer, Hans-Paul Schwefel, and José-Luis Fernández-Villacañas, editors, *Parallel Problem Solving from Nature — PPSN VII*, pages 611–620, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45712-1.

L. Barbulescu, J. P. Watson, L. D. Whitley, and A. E. Howe. Scheduling space–ground communications for the air force satellite control network. *Journal of Scheduling*, 7(1):7–34, Jan 2004. ISSN 1099-1425. doi: 10.1023/B:JOSH.0000013053.32600.3c.

R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1:9–32, 1995. doi: 10.1007/bf02430363.

T. Bartz-Beielstein, C. W. G. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 773–780, 2005. doi: 10.1109/CEC.2005.1554761.

R. Battin. MIT postgraduate taught course: 16.346 Astrodynamics, 2008. URL https://ocw.mit.edu/. Accessed 27-3-2019.

E. Bell and .E Grayzeck. National Space Science Data Center Master Catalogue, 2013. URL https://nssdc.gsfc.nasa.gov/nmc/. Accessed 27-3-2019.

D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statist. Sci.*, 8(1):10–15, 02 1993. doi: 10.1214/ss/1177011077.

M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 11–18, San Francisco, CA, USA, 2002a. Morgan Kaufmann Publishers Inc. ISBN 1-55860-878-8.

M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pages = 11-18,*, 2002b.

J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on Scheduling: From Theory to Applications*. International Handbooks on Information Systems. Springer Berlin Heidelberg, 2007. ISBN 9783540322207.

C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.*, 35(3):268–308, sep 2003. ISSN 0360-0300. doi: 10.1145/937503.937505.

E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence From Natural to Artificial Systems*. Santa Fe Institute: Studies in the Sciences of Complexity. Oxford University Press, 1999. doi: 0-19-513159-2.

M. J. Box. A New Method of Constrained Optimization and a Comparison With Other Methods. *The Computer Journal*, 8(1):42–52, January 1965. ISSN 0010-4620, 1460-2067. doi: 10.1093/comjnl/8.1.42.

M. J. Box. A Comparison of Several Current Optimization Methods, and the use of Transformations in Constrained Problems. *The Computer Journal*, 9(1):67–77, January 1966. ISSN 0010-4620, 1460-2067. doi: 10.1093/comjnl/9.1.67.

T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61:810–837, 2001. doi: 10.1006/jpdc.2000.1714.

C. D. Brown. *Elements of Spacecraft Design*. AIAA Education Series. American Institute of Aeronautics and Astronautics, Incorporated, 2002. ISBN 9781563475245.

J. Brownlee. *Clever algorithms: nature-inspired programming recipes*. Lulu, 2011. ISBN 9781446785065.

K. M. Burjorjee. *Generative fixation: A unified explanation for the adaptive capacity of simple recombinative genetic algorithms*. PhD thesis, Brandeis University, 2009.

F. Castellini and M. Lavagna. Advanced planning and scheduling initiative XMAS tool: aI for automatic scheduling of XMM-newton long term plans. In *The 6th International Workshop on Planning and Scheduling for Space, IWPSS-09, July 19th-July 21st*, 2009.

CCSDS. Blue Books: Recommended Standards. Technical report, 2001. URL https://public.ccsds.org/Publications/BlueBooks.aspx. Accessed 27-3-2019.

CCSDS. Magenta Books: Recommended Practices. Technical report, 2004. URL https://public.ccsds.org/Publications/MagentaBooks.aspx. Accessed 27-3-2019.

A. Cesta, A. Finzi, S. Fratini, A. Orlandini, and E. Tronci. Merging planning, scheduling & verification-a preliminary analysis. In *ASTRA-08. Proceedings of the 10th Workshop on Advanced Space Technologies for Robotics and Automation*, 2008.

A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. MrSPOCK: A long-term planning tool for MARS EXPRESS. In *IWPSS-09. 6th International Workshop on Planning and Scheduling for Space, Pasadena, CA*, 2009.

A. Cesta, G. Cortellessa, S. Fratini, and A. Oddi. MrSPOCK: Steps In Developing An End-to-end Space Application. *Computational Intelligence*, 27(1):83–102, February 2011. ISSN 1467-8640. doi: 10.1111/j.1467-8640.2010.00373.x.

H. Chang and J. M. Williams. Scheduling algorithm for mission planning and logistics evaluation users' guide. *NASA STI/Recon Technical Report N*, 76, May 1976.

S. Chien. Timeline-based Scheduling and Spatial coverage scheduling. In *International Conference on Automated Planning and Scheduling (ICAPS) Summer School 2013*, 2013.

S. Chien and G. Rabideau. ASPEN-Automated Planning and Scheduling for Space Mission Operation. In *Int. Symposium on AI, Robotics and Automation in Space (i-SAIRAS)*, 1997.

S. Chien, G. Rabideau, J. Willis, and T. Mann. Automating planning and scheduling of shuttle payload operations. *Artificial Intelligence*, 114(1-2):239–255, 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00069-7.

S. Chien, G. Rabideau, D. Tran, J. Doubleday, D. Chao, F. Nespoli, M. P. Ayucar, M. C. Sitja, C. Vallat, B. Geiger, and others. Activity-based Scheduling of Science Campaigns for the Rosetta Orbiter: An Early Report on Operations. In *i-SAIRAS 2014 (International Symposium on Artificial Intelligence, Robotics, and Automation in Space)*, 2014.

J. S. Chun, H. K. Jung, and S. Y. Hahn. A study on comparison of optimization performances between immune algorithm and other heuristic algorithms. *IEEE Transactions on Magnetics*, 34(5):2972–2975, September 1998. ISSN 0018-9464. doi: 10.1109/20.717694.

D. P. Clements and D. Joslin. Squeaky Wheel Optimization. *CoRR*, abs/1105.5454, 2011. URL http://arxiv.org/abs/1105.5454.

IEEE-SASB Coordinating Committees. 488.2-1992. Technical report, 1992. URL https://standards.ieee.org/standard/488_2-1992.html. Accessed 27-3-2019.

G. W. Corder and D. I. Foreman. *Nonparametric Statistics for Non-Statisticians*. Nonparametric statistics. John Wiley & Sons Inc, 2011. ISBN 978-0-470-45461-9.

C. A. Cruzen, J. M. Gunn, and P. J. Amadieu. *Space Operations: Exploration, Scientific Utilization, and Technology Development*. Progress in Astronautics and Aeronautics. American Institute of Aeronautics and Astronautics, 2011. ISBN 9781600868177.

W. W. Daniel. *Applied Nonparametric Statistics*. The Duxbury advanced series in statistics and decision sciences. PWS-Kent Publ., 1990. ISBN 9780534919764.

S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. In *IEEE Transactions on Evolutionary Computation*, volume 15, pages 4–31, 2011. doi: 10.1109/TEVC.2010.2059031.

A. Donati. Infusion of innovative technologies for mission operations. *Acta Astronautica*, 67:1132–1137, November 2010. ISSN 0094-5765. doi: 10.1016/j.actaastro.2010.06.023.

M. Dorigo and L. M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, 2004.

M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29 –41, feb 1996. ISSN 1083-4419. doi: 10.1109/3477.484436.

Thompson J. M. Dowsland, K. A. Simulated annealing. In Bäck T. Kok J.N. (Eds.) Rozenberg, G., editor, *Handbook of Natural Computing*, pages 1623–1655. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-540-92910-9_49.

E. Dupnick and D. Wiggins. Scheduling Algorithm for Mission Planning and Logistics Evaluation (SAMPLE). Volume 1: User's guide. Technical report, feb 1980a. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19800021559.pdf. Accessed 27-3-2019.

E. Dupnick and D. Wiggins. Scheduling Algorithm for Mission Planning and Logistics Evaluation (SAMPLE). Volume 3: The GREEDY algorithm. Technical report, February 1980b. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19800021559.pdf. Accessed 27-3-2019.

R. C. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, number 1447 in Lecture Notes in Computer Science, pages 611–616. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64891-8 978-3-540-68515-9.

K. Eggensperger, F. Hutter, H. H. Hoos, and K. Leyton-brown. Surrogate benchmarks for hyperparameter optimization. In *ECAI workshop on Metalearning and Algorithm Selection*, 2014.

K. Eggensperger, M. Lindauer, and F. Hutter. Pitfalls and best practices in algorithm configuration. *CoRR*, 2017.

A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. 1:19–31. doi: 10.1016/j.swevo.2011.02.001.

E. Elbeltagi, T. Hegazy, and D. Grierson. Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43–53, January 2005. ISSN 1474-0346. doi: 10.1016/j.aei.2005.01.004.

Estrack ground stations. Technical report, 2017. URL `https://www.esa.int/Enabling_Support/Operations/Estrack/Estrack_ground_stations`. Accessed 27-3-2019.

S. Falkner, M. Lindauer, and F. Hutter. Spysmac: Automated configuration and performance analysis of sat solvers. In *Theory and Applications of Satisfiability Testing – SAT 2015*, pages 215–222, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24318-4.

J. Felsenstein. Distance matrix methods. In *Inferring Phylogenies*, chapter 11, pages 141–171. Oxford University Press, 2003. ISBN 9780878931774.

M. Feurer, J. T. Springenberg, and F. Hutter. Using meta-learning to initialize bayesian optimization of hyperparameters. In *Proceedings of the 2014 International Conference on Meta-learning and Algorithm Selection - Volume 1201*, MLAS'14, pages 3–10, Aachen, Germany, Germany, 2014. CEUR-WS.org. ISBN 1613-0073.

P. Fortescue, G. Swinerd, and J. Stark. *Spacecraft Systems Engineering, 4th Edition.* 2011. ISBN 978-0-470-75012-4.

S. Fratini. From Scheduling to Planning with Timelines: A history of successful applications in Space (part 2). In *International Conference on Automated Planning and Scheduling (ICAPS) Summer School 2013*, 2013.

A. Fukunaga, G. Rabideau, S. Chien, and D. Yan. Towards an application framework for automated planning and scheduling. In *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation for Space*, 1997.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

S. I. Gass and M. Fu, editors. *Encyclopedia of Operations Research and Management Science*. Springer, New York, 2013.

D. Gaylor, T. Berthold, and N. Takada. Java Astrodynamics Toolkit (JAT), 2002. URL https://opensource.gsfc.nasa.gov/projects/JAT/. Accessed 27-3-2019.

M. Gendreau and J. Y. Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2010. ISBN 978-1-4419-1663-1.

M. E. Giuliano. Multi-Objective Planning and Scheduling with Astronomical Applications. In *International Conference on Automated Planning and Scheduling (ICAPS) Summer School 2013*, 2013.

M. E. Giuliano. Exploring high dimensional metric spaces: A case study using hubble space telescope long range planning. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014*. AAAI, 2014. ISBN 978-1-57735-660-8.

A. Globus, J. Crawford, J. Lohn, and A. Pryor. Scheduling Earth Observing Satellites with Evolutionary Algorithms. In *Proceedings of International Conference on Space Mission Challenges for Information Technology*. NASA Ames Research Center, 01 2003.

F. Glover and K. Sörensen. Metaheuristics. *Scholarpedia*, 10(4):6532, 2015. doi: 10.4249/scholarpedia.6532.

J. J. Grefenstette. Optimization of control parameters for genetic algorithms. volume 16, pages 122–128, 1986. doi: 10.1109/TSMC.1986.289288.

M. Griffin and J. French. *Space Vehicle Design*. AIAA Education. American Institute of Aeronautics & Astronautics, 2004. ISBN 978-1563475399.

M. Hamdy, A-T. Nguyend, and J.L.M. Hensen. A performance comparison of multi-objective optimization algorithms for solving nearly-zero-energy-building design problems.

P. J. B. Hancock. An empirical comparison of selection methods in evolutionary algorithms. *Evolutionary Computing*, 865:80–94, 1994. doi: 10.1007/3-540-58483-8_7.

N. Hansen and A. Ostermeier. Convergence properties of Evolution Strategies with the Derandomized Covariance Matrix Adaptation: the $(\mu/\mu i, \lambda)$–CMA-ES. In *Proceedings of the 5th European Conference on Intelligent Techniques and Soft Computing*, 1997.

B. Henehan and M. Johas-Teener. 1394 Standards and Specifications Summary. Technical report, 2006. URL http://www.1394ta.org/developers/specifications/StandardsOrientationV5.0.pdf. Accessed 27-3-2019.

K. Holmström. The TOMLAB optimization environment in MATLAB. *Advanced Modeling and Optimization*, 1:47–69, 1999.

H. H. Hoos. *Autonomous Search: Automated Algorithm Configuration and Parameter Tuning*, pages 37–71. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-21434-9.

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.*, 36:267–306, 2009. doi: 10.1613/jair.2861.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pages 507–523, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25565-6. doi: 10.1007/978-3-642-25566-3_40.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. An evaluation of sequential model-based optimization for expensive blackbox functions. In *GECCO 2013 - Proceed-*

ings of the 2013 Genetic and Evolutionary Computation Conference Companion, pages 1209–1216, 07 2013.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *31st International Conference on Machine Learning, ICML 2014*, volume 2, pages 1130–1144, 01 2014.

W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *Journal of Global Optimization*, 14:331–355, 1999.

A. K. Hyder. *Spacecraft Power Technologies*. Imperial College Press, 2000. ISBN 1860941176.

C. Iacopino. Highly Responsive MPS for Dynamic EO Scenarios. American Institute of Aeronautics and Astronautics, June 2012. doi: 10.2514/6.2012-1275545.

C. Iacopino and P. Palmer. How ants can manage your satellites. In *Twenty-Third International Joint Conferences on Artificial Intelligence*, 2013.

C. Iacopino, P. Palmer, and N. Policella. A stigmergy-based paradigm for mission planning and scheduling of multiple spacecraft. *AI in Space: Intelligence beyond planet Earth, Barcelona*, 2011.

C. Iacopino, P. Palmer, A. Brewer, N. Policella, and A. Donati. EO constellation MPS based on Ant Colony Optimization algorithms. In *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, pages 159–164, June 2013. doi: 10.1109/RAST.2013.6581192.

J. Johnson, L. Bogovich, A. Tuchman, A. Kispert, B. Page, C. Burkhardt, R. Littlefield, D. Mclean, W. Potter, and W. Ochs. An intelligent planning and scheduling system for the HST servicing missions. In *Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations*, pages 875–880, 1993.

M. D. Johnston and G. Miller. Spike: Intelligent scheduling of Hubble Space Telescope Observations. *Intelligent Scheduling*, pages 391–422, 1994.

E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001. URL http://www.scipy.org/. Accessed 27-3-2019.

G. K. Kanji. *100 Statistical tests*. SAGE Publications, 2006. ISBN 9781412923767.

S. Karagöz and A.R. Yildiz. A comparison of recent metaheuristic algorithms for crashworthiness optimisation of vehicle thin-walled tubes considering sheet metal forming effects. *International Journal of Vehicle Design*, 2017.

D. C. Karnopp. Random search techniques for optimization problems. *Automatica*, 1(2-3):111–121, Aug 1963. doi: 10.1016/0005-1098(63)90018-9.

G. Keppel and T.D. Wickens. *Design and Analysis: A Researcher's Handbook*. Prentice Hall, 2004. ISBN 9780135159415.

G. Khrypunova, A. Romeob, F. Kurdesauc, D. L. Bätznerd, H. Zogge, and A. N. Tiwarie. Recent developments in evaporated CdTe solar cells. *Solar Energy Materials & Solar Cells*, 90:664 – 677, 2006. doi: 10.1016/j.solmat.2005.04.003.

H. Y. Kim. Statistical notes for clinical researchers: post-hoc multiple comparisons. *Restorative Dentistry & Endodontics*, 40:172–176, 2015. doi: 10.5395/rde.2015.40.2.172.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

T. Kluyver, F. Ragan-Kelley, B. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, Abdalla S. Avila, D., C. Willing, and Jupyter Development Team. Jupyter Notebooks - a publishing format for reproducible computational workflows. pages 87–90, 2018. doi: 10.3233/978-1-61499-649-1-87.

E. Komninou, M. Vasile, and E. Minisci. Optimal power harness routing for small-scale satellites. In *Proc. 62nd International Astronautical Congress, International Astronautical Federation*, 2011.

E. Komninou, M. Vasile, and E. Minisci. Optimal dynamic operations scheduling for small-scale satellites. In *Proc. 63rd International Astronautical Congress, International Astronautical Federation*, 10 2012a.

E. Komninou, M. Vasile, and E. Minisci. An integrated system-operations approach to the optimal design of small-scale satellites. In *Proc. 5th Nano-satellite Symposium, University of Tokyo*, 10 2012b.

Kongsberg Satellite Services. Technical report, 1997. URL http://www.ksat.no/en/ServicesKSAT/SvabardSatelliteStationSvalSatpage. Accessed 27-3-2019.

H. J Kramer. PROBA2 (Project for On-Board Autonomy-2), 2002. URL https://directory.eoportal.org/web/eoportal/satellite-missions/p/proba-2. Accessed 27-3-2019.

W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. 47:583–621, 1952.

T. A. Lavigna and F. L. Hornbuckle. The ATS-6 power system: Hardware implementation and orbital performance. Technical report, 1977. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770025285.pdf. Accessed 27-3-2019.

J. D. Leeper. Choosing the correct statistical test in SAS, Stata, SPSS and R, 2011. URL https://stats.idre.ucla.edu/other/mult-pkg/whatstat/. Accessed 27-3-2019.

J. Levine and F. Ducatelle. Ant colony optimization and local search for bin packing and cutting stock problems. *J Oper Res Soc*, 55, 2004.

M. Lindauer and F. Hutter. Warmstarting of model-based algorithm configuration. *Journal of Artificial Intelligence Research*, abs/1709.04636, 2017.

F. J. Lobo, C. F. Lima, and Z. Michalewicz. *Parameter Setting in Evolutionary Algorithms*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2007.

M. López-Ibáñez and T. Stützle. Automatic configuration of multi-objective aco algorithms. In *Swarm Intelligence*, pages 95–106, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15461-4.

S. Luke. *Essentials of Metaheuristics*. Lulu, Second edition, 2013. URL http://cs.gmu.edu/$\sim$sean/book/metaheuristics/. Accessed 27-3-2019.

Lund Research Ltd. Friedman Test in SPSS Statistics, 2012. URL https://statistics.laerd.com/spss-tutorials/friedman-test-using-spss-statistics.php. Accessed 27-3-2019.

D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003. ISBN 9780521642989.

E. Marshall and B. Marquier. Friedman test in R, 2014. URL https://www.sheffield.ac.uk/polopoly_fs/1.714578!/file/stcp-marquier-FriedmanR.pdf. Accessed 27-3-2019.

J. H. McDonald. *Handbook of Biological Statistics (3rd ed.)*. Sparky House Publishing, Baltimore, Maryland, 2014. URL http://www.biostathandbook.com/. Accessed 27-3-2019.

B. McKissock, P. Loyselle, and E. Vogel. Guidelines on Lithium-ion Battery Use in Space Applications. Technical report, 2009. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20090023862.pdf. Accessed 27-3-2019.

D. R. McLean, B. J. Page, A. Tuchman, A. V. Kispert, W. L. Yen, and W. J. Potter. Emphasizing conflict resolution versus conflict avoidance during schedule generation. *Expert Systems with Applications*, 5(3-4):441–446, 1992. ISSN 0957-4174. doi: 10.1016/0957-4174(92)90028-Q.

M. Mitchell. Genetic algorithms: An overview. *Complexity*, 1(1):31–39, 1995. ISSN 1099-0526. doi: 10.1002/cplx.6130010108.

M. Mitchell. *An Introduction to Genetic Algorithms*. Complex Adaptive Systems. The MIT Press, 1998. ISBN 9780262631853.

C. G. Moles, P. Mendes, and J. R. Banga. Parameter Estimation in Biochemical Pathways: A Comparison of Global Optimization Methods. *Genome Research*, 13(11):2467–2474, January 2003. ISSN 1088-9051, 1549-5469. doi: 10.1101/gr.1262503.

M. D. Morris. Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics*, 33:161–174, 1991. doi: 10.1080/00401706.1991.10484804.

V. Nannen and A.E. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 183–190, 2006.

NASA. *CubeSat 101 Basic Concepts and Processes for First-Time CubeSat Developer*. NASA CubeSat Launch Initiative. National Aeronautics and Space Administration, 2017. URL https://www.nasa.gov/sites/default/files/atoms/files/nasa_csli_cubesat_101_508.pdf. Accessed 27-3-2019.

National Aeronautics and Space Administration. NASA Technical Reports Server, 2014. URL http://ntrs.nasa.gov. Accessed 27-3-2019.

National Imagery and Mapping Agency. World Geodetic System 1984: Its definition and relationships with Local Geodesic Systems. Technical Report NIMA TR8350.2, Department of Defense, January 2000. URL http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf. Accessed 27-3-2019.

National Institute of Standards and Technology. McNemar test, 2008. URL https://itl.nist.gov/div898/software/dataplot/refman1/auxillar/mcnemar.htm. Accessed 27-3-2019.

P. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.

S. Panda and N. P. Padhy. Comparison of Particle Swarm Optimization and Genetic Algorithm for FACTS-based controller design. *Applied Soft Computing*, 8(4):1418–1427, September 2008. ISSN 1568-4946. doi: 10.1016/j.asoc.2007.10.009.

M. R. Patel. *Spacecraft Power Systems*. Taylor & Francis, 2004. ISBN 9780849327865.

R. Pierce. Expert mission planning and replanning scheduling system for NASA KSC payload operations. In *First Annual Workshop on Space operations Automation and Robotics*, pages 299–306, oct 1987. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19880007863.pdf. Accessed 27-3-2019.

F. Pinel, G. Danoy, and P. Bouvry. *Evolutionary Algorithm Parameter Tuning with Sensitivity Analysis*, pages 204–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

N. Policella. From Scheduling to Planning with Timelines: A history of successful applications in Space. In *International Conference on Automated Planning and Scheduling (ICAPS) Summer School 2013*, 2013.

C. Pralet and G. Verfaillie. AIMS: A Tool for Long-term Planning of the ESA INTEGRAL Mission. In *Proceedings of the 6th International Workshop on Planning and Scheduling for Space, IWPSS09*, 2009.

C. Pralet, G. Verfaillie, X. Olive, S. Rainjonneau, and I. Sebbag. Allocation of Downlink Windows for a Constellation of Satellites. In *i-SAIRAS 2012 (International Symposium on Artificial Intelligence, Robotics, and Automation in Space)*, 2012.

K. Price and R. Storn. Differential evolution (DE) for Continuous Function Optimization, 1997. URL https://www1.icsi.berkeley.edu/~storn/code.html. Accessed 27-3-2019.

K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005. ISBN 978-3-540-20950-8.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org/. Accessed 27-3-2019.

M. D. R-Moreno, J. Kurien, A. Cesta, and V. S. M. della Battaglia. Innovative AI Technologies for Future ESA Missions. In *Proceedings of 10th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, November 2008.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations using the ASPEN system. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, page 99, 1999.

G. Rabideau, S. Chien, R. Sherwood, D. Tran, B. Cichy, D. Mandl, S. Frye, S. Shulman, R. Bote, J. Szwaczkowski, et al. Mission operations with autonomy: a preliminary report for Earth Observing-1. 2004.

G. Rabideau, C. Polanskey, J. Doubleday, S. Chien, and S. P. Joy. A Data Management Tool for Dawn Science Planning. In *i-SAIRAS 2014 (International Symposium on Artificial Intelligence, Robotics, and Automation in Space)*, 2014.

Redu Space Services. Technical report, 2007. URL http://www.reduspaceservices.com/operations-maintenance-infrastructure/proba-operations. Accessed 27-3-2019.

V. Roberge, M. Tarbouchi, and G. Labonté. Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning. *IEEE Transactions on Industrial Informatics*, 9:132–141, 2013. doi: 10.1109/tii.2012.2198665.

D. Roddy. *Satellite Communications 4rd ed.* McGraw-Hill Education, 2006. ISBN 0071462988.

C. J. Rodrigues Capela. Protocol of communications for VORSat satellite – link budget. Master's thesis, Universidade do Porto – Faculdade de Engenharia, 2012.

H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *Comput. J.*, 3:175–184, 1960a.

H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, January 1960b. ISSN 0010-4620, 1460-2067. doi: 10.1093/comjnl/3.3.175.

T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4:284–294, 2000.

T. J. Santner, B. J. Williams, and W. I. Notz. The Design and Analysis of Computer Experiments. In *Springer series in statistics*, 2003. doi: 10.1007/978-1-4939-8847-1.

H. P. Schwefel. *Evolution and Optimum Seeking*. Sixth Generation Computer Technologies. Wiley, 1995. ISBN 9780471571483.

PROBA2 Science Center (P2SC). PROBA2 Ancillary Data, 2009. URL http://proba2.sidc.be/aux/data/spice/kernels/. Accessed 27-3-2019.

R. S. Sexton, R. E. Dorsey, and J. D. Johnson. Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. *Decision Support Systems*, 22(2):171–185, February 1998. ISSN 0167-9236. doi: 10.1016/S0167-9236(97)00040-7.

N. F. Shepard, C. V. Stahle, A. Schneider, and K. L. Hanson. Feasibility study of a 110 Watt per kilogram lightweight solar array system. Technical report, 1972. URL https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19720022383.pdf. Accessed 27-3-2019.

R. Shi, C. Cui, K. Su, and Z. Zain. Comparison Study of Two Meta-heuristic Algorithms with their Applications to Distributed Generation Planning. *Energy Procedia*, 12:245–252, 2011. doi: 10.1016/j.egypro.2011.10.034.

S. S. Skiena. *The Algorithm Design Manual*. Springer London, London, 2008. ISBN 978-1-84800-069-8.

M. Soltero, J. Zhang, and C. Cockril. RS-422 and RS-485 Standards Overview and System Configurations. Technical report, 2010.

W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, 1962. ISSN 00401706.

United States Department of Defense. MIL-STD-1553, Digital Time Division Command/Response Multiplex Data Bus. Technical report, 2018.

R. Steel, M. Niezette, A. Cesta, S. Fratini, A. Oddi, G. Cortellessa, R. Rasconi, G. Verfaillie, C. Pralet, M. Lavagna, and others. Advanced planning and scheduling initiative: MrSPOCK AIMS for XMAS in the space domain. In *The 6th*

International Workshop on Planning and Scheduling for Space, IWPSS-09, July 19th-July 21st, 2009.

R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11 (4):341–359, Dec 1997. ISSN 1573-2916. doi: 10.1023/A:1008202821328.

J. Sun and M. Q. Chester. Optimising Ground Stations Scheduling with a Genetic Algorithm. In *i-SAIRAS 2010 (International Symposium on Artificial Intelligence, Robotics, and Automation in Space)*. JAXA, 2010.

B. Teixeira de Sousa, A. Accomazzo, I. Shaw, and R. Steel. VEX MPS – supporting the Venus Express Mission Planning Concept. Space Telescope Science Institute, International Workshop on Planning and Scheduling for Space, 2006.

United States Naval Observatory. *Astronomical Almanac for the Year 2009 and Its Companion, the Astronomical Almanac Online: Data for Astronomy, Space Sciences, Geodesy, Surveying, Navigation, and Other Applications*. ASTRONOMICAL ALMANAC FOR THE YEAR. U.S. Government Printing Office, 2008. ISBN 9780118873420.

M. Vallati, C. Fawcett, A. Gerevini, H. H. Hoos, and A. Saetti. Automatic generation of efficient domain-optimized planners from generic parametrized planners. 05 2013.

M. Vasile, E. Minisci, F. Zuiani, E. Komninou, and Q. Wijnands. Fast evidence-based space system engineering. In *Proc. 62nd International Astronautical Congress, International Astronautical Federation*, 10 2011.

M. Vasile, E. Minisci, and Q. Wijnands. Approximated computation of belief functions for robust design optimization. In *Proc. 14th AIAA Non-Deterministic Approaches Conference*, 4 2012.

VEGA, Institute of Cognitive Sciences, Technologies, Office National d'Études et de Recherches Aérospatiales, Politecnico di Milano, and European Space Agency. Advanced Planning and Scheduling Initiative (APSI), December 2006 - December 2008. URL http://www.esa.int/Our_Activities/Operations/APSI_br_Advanced_Planning_Scheduling_Initiative. Accessed 27-3-2019.

G. Verfaillie. Models and algorithms for planning activities of Earth surveillance and Observation satellites. In *International Conference on Automated Planning and Scheduling (ICAPS) Summer School 2013*, 2013.

L. Wei-Cheng, L. Da-Yin, and L. Chung-Yang. Daily imaging scheduling of an Earth Observation satellite. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, (2):213–223, 03 2005. ISSN 1083-4427. doi: 10.1109/TSMCA.2005.843380.

E. W. Weisstein. Graph Distance Matrix. URL http://mathworld.wolfram.com/GraphDistanceMatrix.html. Accessed 27-3-2019.

E. W. Weisstein. Traveling Salesman Problem. From Mathworld – A Wolfram Web Resource, 2013. URL http://mathworld.wolfram.com/TravelingSalesmanProblem.html. Accessed 27-3-2019.

J. R. Wertz and W. J. Larson. *Space Mission Analysis and Design*. Space Technology Library. Springer Netherlands, 1999. ISBN 9780792359012.

M. Wetter and J. Wright. A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment*, 39(8):989–999, August 2004. ISSN 0360-1323. doi: 10.1016/j.buildenv.2004.01.022.

M. H. Wright. Nelder, Mead, and the other simplex method. *Documenta Mathematica*, 7, 2010.

X. S. Yang, S. Deb, M. Loomes, and M. Karamanoglu. A framework for self-tuning optimization algorithm. 23:2051–2057, 2013. doi: 10.1007/s00521-013-1498-4.

Z. Yuan, M. A. Montes de Oca, M. Birattari, and T. Stützle. Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. 6:49–75, 2012. doi: 10.1007/s11721-011-0065-9.

D. Zaharie. A comparative analysis of crossover variants in differential evolution. In *Proceedings of the International Multiconference on Computer Science and Information Technology*, pages 171–181, 2007.

J. Zender, 2012 - 15. personal communication.

N. Zufferey, P. Amstutz, and P. Giaccari. Graph colouring approaches for a satellite range scheduling problem. *Journal of Scheduling*, 11(4):263–277, Aug 2008. ISSN 1099-1425. doi: 10.1007/s10951-008-0066-8.

M. Zweben, E. Davis, B. Daun, and M.J. Deale. Scheduling and rescheduling with iterative repair. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1588–1596, November 1993. ISSN 0018-9472. doi: 10.1109/21.257756.

A
Solar cell performance data

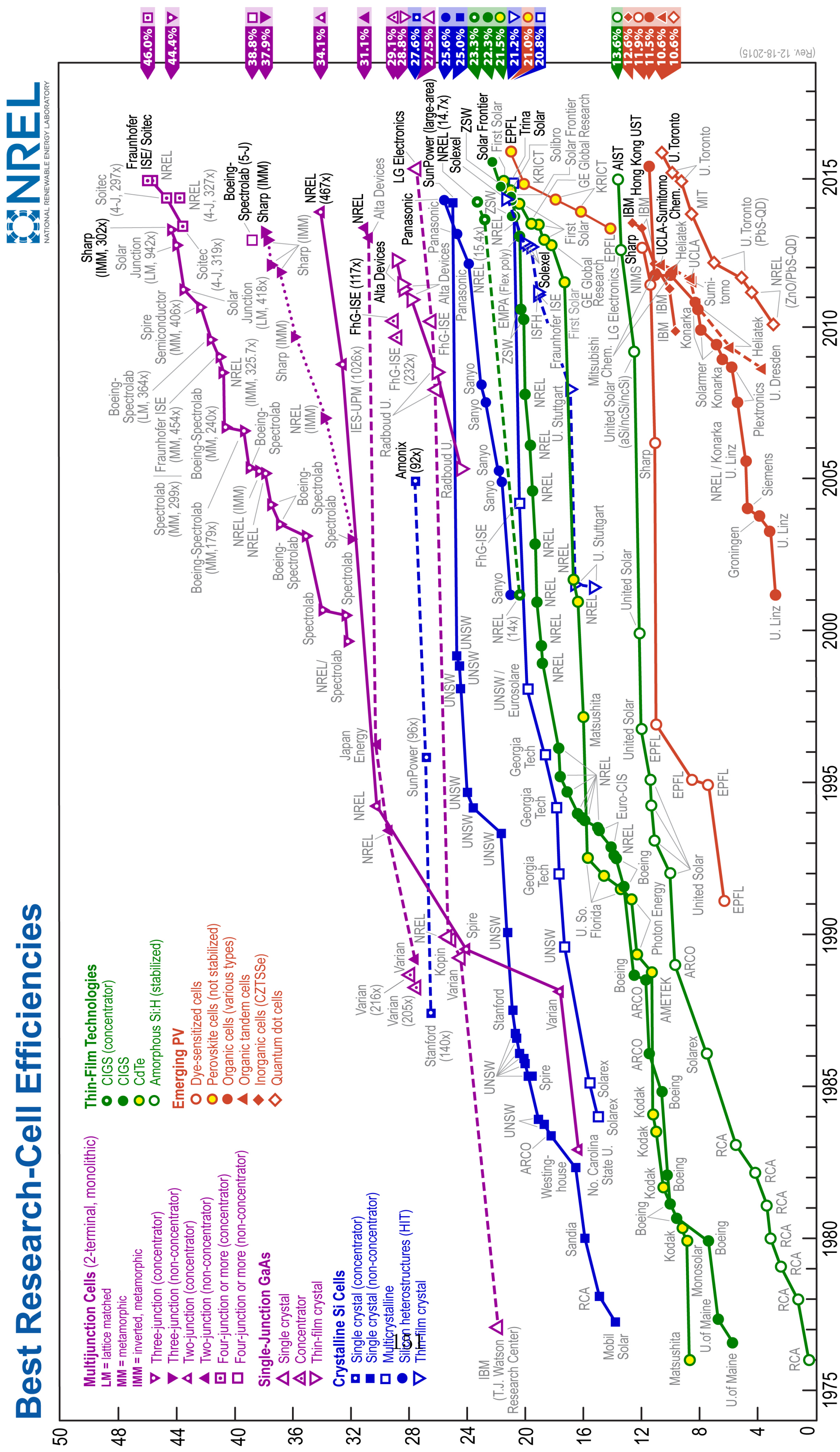


Figure A.1: Timeline of solar cell energy conversion efficiencies

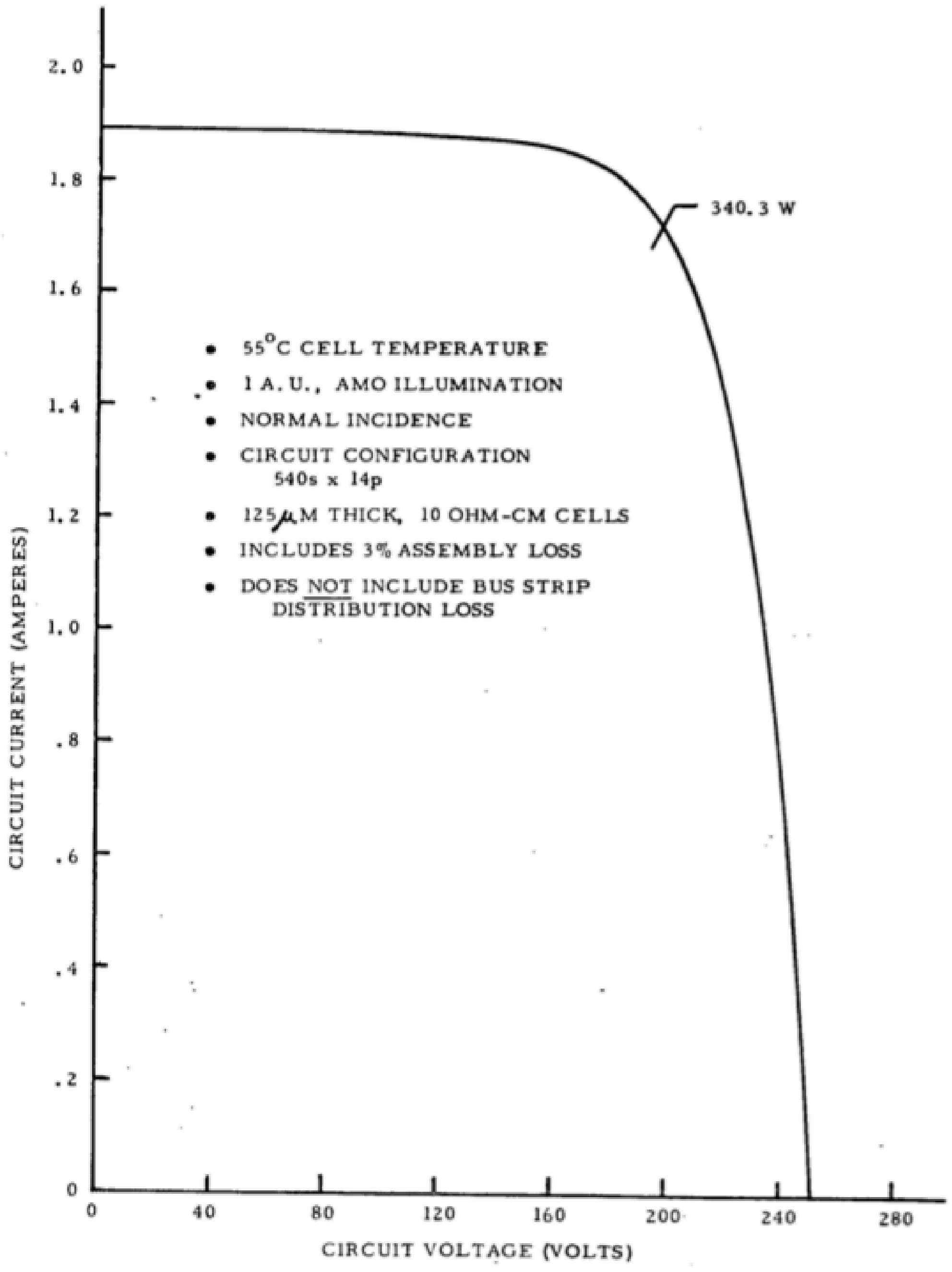


Figure A.2: Solar cell I-V characteristics [Shepard et al., 1972]. Notice that operating temperature (see Figure A.3), distance from the Sun and angle of incidence as well as air mass (AM0) influence its power output.

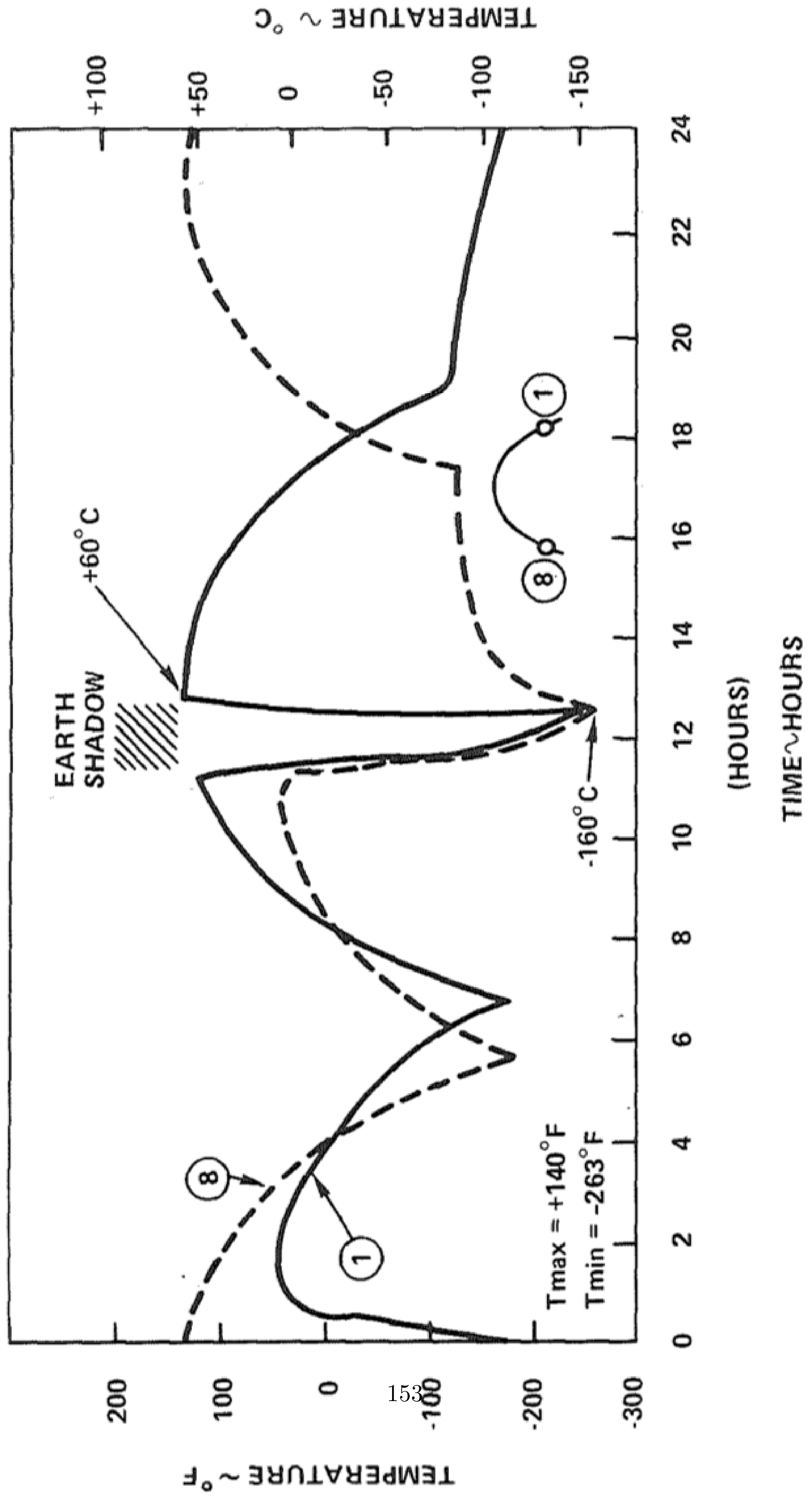


Figure A.3: ATS-6 solar panel temperature profile [Lavigna and Hornbuckle, 1977] for a 23.53hr Geosynchronous Orbit.

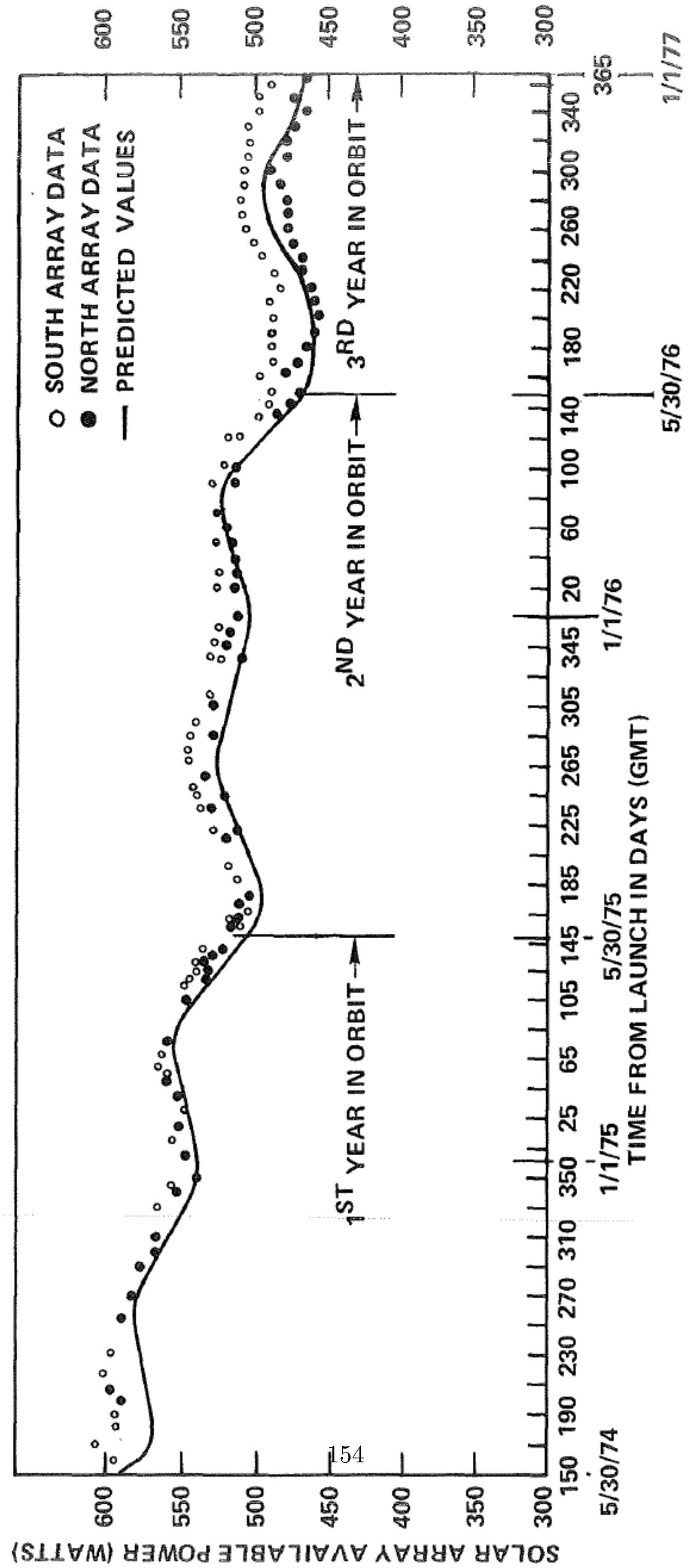


Figure A.4: ATS-6 measured solar array power output [Lavigna and Hornbuckle, 1977]. Notice power output degradation over time.

B
PROBA-2 Science Planning Automation

European Space Agency
Research and Science Support Department
Solar System Science Operations Division

PROBA2 Science Planning Automation

PROBA2-EST-TN-003

Issue 1.0

20 February 2013

Distribution List

This technical note will be distributed to:

Change Log

Draft	1	20 February 2013

Table of Content

A. Introduction

PROBA2 is a small satellite in sun-synchronous Earth orbit since November 2009. The Mission Operations Centre (MOC) is located at the Redu Ground Station in Redu, Belgium. The PROBA2 Science Centre (P2SC) is co-located with the Solar Influence Data Centre (SIDC), located at the Royal Observatory of Belgium, Brussels, Belgium.

The mission uses S-band up- and downlink using one of the S-band antennae in Redu and one S-band antenna in Svalbard, operated by Kongsberg Satellite Services AS (KSAT).

B. Scope

This document describes the activities around the science planning executed at the Royal Observatory of Belgium: it gives the process from the constraints and the science ideas to the final commanding of the instruments SWAP and LYRA. From these activities, potential system requirements for a further automation of the science planning tools are derived.

C. S-band Pass Planning

The S-band antenna in Redu is only visible above the horizon during a few orbits on a day for a reasonable duration. The angle upon the horizon, above which a good communication is guaranteed, is typically 10 degrees (tbc) and the duration acceptable shall be at least 9 minutes (tbc).

The Svalbard S-band antenna is visible each orbit for the PROBA2 satellite. As each pass has to be paid, the passes are selected carefully with a two weeks horizon every week (so, every week there is a two weeks planning, of which the first week is fixed and the second week might be updated – if necessary – in the next planning in the week afterwards). ESA has purchased on a average 5 passes per day from KSAT.

The passes are selected such that the overall downlink volume is maximized. Of course, data must exist on-board to fit to the downlink estimations.

To compute the geometrical parameters (visibility, angles, duration) there might be several capabilities; one would be the usage of the SPICE toolkit from the NAIF team at JPL. SPICE kernels are available.

PASS-01	Pass selection	The tool shall select the Redu S-band passes and the Svalbard S-band passes such that the overall transmission volume is maximized.
PASS-02	Pass configuration	The tool shall allow the parameterization of all variable parameters, e.g. the angle upon the horizon, the minimal duration of a pass, the minimal and maximal number of passes per day per station.
PASS-03	Pass constraints	The tool shall accept pass availability constraints, especially that there are periods in which no passes of a station are available, e.g. when already scheduled for other satellites.

D. SWAP Planning

The SWAP Instrument Operations Sheet is typically prepared once a week (example in ANNEX 1). The plan does include calibration requests, scientific requests, adaptations for other instruments (e.g. the weekly ESP jump), it takes care of Large Angle Rotations (LARs), eclipses during the eclipse season, etc.

- The SWAP imager when in imaging mode makes an image every n seconds. This is called the imaging cadence n, e.g. 120 seconds cadence.
- Weekly ESP jump: once a week, the ESP instrument is operated for 20 minutes. During the ESP operations, SWAP shall not start an image acquisition. As a consequence, the operator select two instances of neighborhood LARs, ensures that the SWAP imaging stops well before the LAR and restarts well after the LAR. The operator does this using the TABLE functionality on-board PROBA2; basically the operator requests a 10 second integration time using a 30 minute cadence and then changes the cadence before the first image is acquired after the LAR.
- None of the following calibration campaigns shall be executed when flying over the South Atlantic Anomaly.
- Every two weeks, a LED calibration routine is executed. The calibration takes about 50 minutes and typically looks like the following example: (details to come)

```
2013.02.26T04:00:00.000 idle
2013.02.26T04:00:10.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1 led_a_on 60 30 12bits 0.02617 0.02617
2013.02.26T04:00:20.000 data_management off 10 off off 10 3600 off 0 off float 128 8 off off 255 off
2013.02.26T04:00:53.000 specific_acquisition
2013.02.26T04:02:30.000 data_management off 10 off off 0 0 off 0 on float 128 8 off off 0 off
2013.02.26T04:07:30.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1 led_b_on 60 30 12bits 0.02617 0.02617
2013.02.26T04:12:30.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1 led_off 60 30 12bits 0.02617 0.02617
2013.02.26T04:17:30.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1 led_off 30 30 12bits 0.02617 0.02617
2013.02.26T04:27:40.000 data_management on 10 off fixed 10 3600 jpeg 3 on float 128 8 off off 255 off
2013.02.26T04:27:50.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1 led_a_on 100 30 12bits 0.0 0.0
2013.02.26T04:28:40.000 data_management off 10 off off 10 3600 off 0 off float 128 8 off off 0 off
2013.02.26T04:28:50.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1 led_off 100 30 12bits 0.0 0.0
2013.02.26T04:52:10.000 data_management on 10 off fixed 10 3600 jpeg 0 on float 128 8 off off 255 off
```

- From November to February every year, the satellite is exposed to eclipses. This means that the tangential altitude (take the nearest line from the line of sight from the satellite to the sun to the earth) is approaching 0 meters and goes into negative for visual eclipses. When the tangential altitude is between 0 and 400 (tbc) meters, then the LYRA and SWAP instruments are exposed to a reduction of EUV light and acquire less photons on the detectors. As the SWAP team is not interested to obtain images during the occultations, the imager is programmed such that no images are acquired between a pair of tangential altitudes, e.g. 400 meters. That means when ever the tangential altitude is less then 400 meters, no images are obtained.

- Once upon a while, the satellite is going into SAFE mode. Then the imager has to be 'reset' and restarted. See example.

```
2013.02.18T13:45:00.000 data_management on 10 off fixed 10 3600 jpeg 0 on float 128 8 off off 255 off
```

- The SWAP imager has on-board a large image buffer. This buffer is configurable, but since a long time fixed to be able to store onboard 280 images. An image is 1kx1k in size and is compressed with a compression rate up to 2.7 (tbc). During a 10 minute downlink, one can typically transfer 45Mbytes of data, that corresponds to about 60 images (JZ to check all numbers!). In case the image buffer is full, the images with the lowest priorities are overwritten. Priorities are set from ground within the SWAP commanding.
- Off-pointing campaigns

SWAP-01	ESP Support	The tool shall schedule a timeslot for the 20 minutes operations of ESP once a week (currently Thursday a.m.) taking care of the LARs.
SWAP-02	LED calibration	The tool shall schedule every two weeks a LED calibration campaign. The execution day of the LED calibration campaign shall be configurable.
SWAP-03	SAA	The tool shall take into account the position of the satellite and not plan any calibration campaign when over the South Atlantic Anomaly.
SWAP-04	Occultations	The tool shall take into account the tangential altitude of the SWAP imager line of sight and avoid the imaging when less than settable tangential altitude.
SWAP-05	SAFE mode	The tool shall initialize the SWAP imager after returning from satellite SAFE mode.
SWAP-06	Cadence	The tool shall ensure that the maximum cadence possible is used.

E. LYRA Planning

The LYRA Instrument Operations Sheet is typically prepared once a week (example in ANNEX 2). The plan does include calibration requests, occultation/eclipse campaigns, etc.

- Every two weeks a LED calibration campaign is executed using unit1 and unit3. The campaign takes about 9 hours. One does not have to take the LARs into account. See example.

```
2013.02.27T09:00:00.000 set_cover 2 close
2013.02.27T09:01:00.000 acquisition 50ms unit_2 unit_1 100 off 0
2013.02.27T09:02:00.000 acquisition 50ms unit_2 unit_1 200000 off 0
2013.02.27T09:42:00.000 acquisition 50ms unit_2 unit_1 200000 vis 255
2013.02.27T11:22:00.000 acquisition 50ms unit_2 unit_1 200000 uv 255
2013.02.27T13:02:00.000 acquisition 50ms unit_2 unit_1 200000 off 0
2013.02.27T13:42:00.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.27T13:43:00.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.27T14:23:00.000 acquisition 50ms unit_2 unit_3 200000 vis 255
2013.02.27T16:03:00.000 acquisition 50ms unit_2 unit_3 200000 uv 255
2013.02.27T17:43:00.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.27T18:23:00.000 acquisition 50ms unit_2 off 100 off 0
2013.02.27T18:24:00.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.27T18:25:00.000 set_cover 2 open
```

- Once upon a time a ASIC reload is executed (is done via Flight Control Procedures by MOC when told by P2SC via email). Before and after the ASIC reload a 2 hour period of dark current is sometimes requested. See example.

```
2013.02.26T02:37:00.000 set_cover 2 close
2013.02.26T02:38:00.000 acquisition 50ms unit_2 unit_3 200000 off 0    ;;; ASIC reload at 03:38
2013.02.26T04:38:00.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.26T04:39:00.000 set_cover 2 open
```

- From November to February every year, the satellite is exposed to eclipses. This means that the tangential altitude (take the nearest line from the line of sight from the satellite to the sun to the earth) is approaching 0 meters and goes into negative for visual eclipses. When the tangential altitude is between 0 and 400 (tbc) meters, then the LYRA and SWAP instruments are exposed to a reduction of EUV light and acquire less photons on the detectors. For LYRA, this period is used to obtain data from unit1 or unit3 on each day of the week or on one day of the week. (see example)

```
2013.02.21T08:57:32.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.21T08:58:32.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.21T09:02:32.000 set_cover 3 open                        ⇐- occultation start
2013.02.21T09:39:53.000 set_cover 3 close                        ⇐ occultation end
2013.02.21T09:44:53.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.22T09:46:14.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.22T09:47:14.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.22T09:51:14.000 set_cover 3 open
2013.02.22T10:27:33.000 set_cover 3 close
2013.02.22T10:32:33.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.23T08:55:44.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.23T08:56:44.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.23T09:00:44.000 set_cover 3 open
2013.02.23T09:35:59.000 set_cover 3 close
2013.02.23T09:40:59.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.24T09:44:34.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.24T09:45:34.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.24T09:49:34.000 set_cover 3 open
2013.02.24T10:23:32.000 set_cover 3 close
```

- Once upon a while, the satellite is going into SAFE mode. Then the imager has to be ‘reset’ and restarted. See example.

```
2013.02.18T13:45:00.000 warm_up 50ms unit_2 off 100 off 0 open close
2013.02.18T13:52:00.000 set_heater ab 1 off
2013.02.18T13:52:05.000 set_heater ab 2 off
2013.02.18T13:52:10.000 set_heater ab 3 off
2013.02.18T13:53:00.000 acquisition 50ms unit_2 off 200000 off 0
```

- Typically once a month, a comparison observation with Unit2/Unit1 and Unit2/Unit3 is acquired. This calibration campaign takes about 4 hours and 30minutes.

```
2013.02.15T05:00:00.000 acquisition 50ms unit_2 unit_1 100 off 0
2013.02.15T05:01:00.000 acquisition 50ms unit_2 unit_1 200000 off 0
2013.02.15T05:05:00.000 set_cover 1 open
2013.02.15T05:55:00.000 set_cover 1 close
2013.02.15T06:00:00.000 acquisition 50ms unit_2 off 200000 off 0
2013.02.15T09:07:42.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.02.15T09:08:42.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.02.15T09:12:42.000 set_cover 3 open
2013.02.15T09:32:42.000 set_cover 3 close
2013.02.15T09:37:42.000 acquisition 50ms unit_2 off 200000 off 0
```

- whenever there is an active region on the disk that is likely to produce large flares, the LYRA PI might request a flare hunting campaign with unit 1 or unit 3. See example

```
2013.01.11T18:10:00.000 acquisition 50ms unit_2 unit_3 100 off 0
2013.01.11T18:11:00.000 acquisition 50ms unit_2 unit_3 200000 off 0
2013.01.11T18:15:00.000 set_cover 3 open
2013.01.12T20:45:00.000 set_cover 3 close
2013.01.12T20:50:00.000 acquisition 50ms unit_2 off 200000 off 0
```

LYRA-01	LED calibration	The tool shall schedule every two weeks a LED calibration campaign. The preferred execution day(s) shall be configurable.
LYRA-02	ASIC Dark current	The tool shall schedule dark current acquisitions whenever an ASIC reload is executed (tbc by PI)
LYRA-03	Occultation campaigns	The tool shall schedule occultation campaigns. Unit used and repetition frequency shall be parameters.
LYRA-04	SAFE mode	The tool shall initialize the LYRA radiometer after returning from satellite SAFE mode.
LYRA-05	Calibration campaign	The tool shall execute a unit calibration campaign on a frequency to be parameterized, e.g. once a month on a specify day, every two weeks on a specify day, …
LYRA-06	Flare hunting	The tool shall schedule a flare hunting campaign whenever the SIDC declares the probability of a X-flare larger than x %.
LYRA-07	Campaign exclusion	The tool shall never schedule the following campaigns in parallel: - flare hunting and LED calibration - tbw

F. Planning Interactions

- Sometimes, occultation campaigns are executed with both SWAP and LYRA acquiring data through several occultation periods a day.

- Sometimes, campaigns with other satellites are executed. Then it must be possible to freeze a part of the science planning towards the timeline; actually give the command period as input.

COM-01	Occulation campaigns	The tool shall schedule occultation campaigns using both the SWAP and LYRA instrument.
COM-02	Other satellites	The tool shall allow the settings of execution periods for both SWAP and LYRA in case observations are done with other spacecraft. Twice a year, the SDO satellite undergoes eclipses. SWAP imaging shall take place at a higher cadence (60 sec, TBD) during these periods, non-interrupted with other scientific campaigns or calibration campaigns

```
SWAP
00456
2013.02.25T10:29:08.000
2013.02.26T00:00:00.000
# generated on 2013-02-25T10:29:08Z by ios.xsl version 1.1
2013.02.26T04:00:00.000 idle
2013.02.26T04:00:10.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1
led_a_on 60 30 12bits 0.02617 0.02617
2013.02.26T04:00:20.000 data_management off 10 off off 10 3600 off 0 off float 128 8 off off 255 off
2013.02.26T04:00:53.000 specific_acquisition
2013.02.26T04:02:30.000 data_management off 10 off off 0 0 off 0 on float 128 8 off off 0 off
2013.02.26T04:07:30.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1
led_b_on 60 30 12bits 0.02617 0.02617
2013.02.26T04:12:30.000 acquisition_configuration correlated_double_sampling 3 0 0 1023 1023 59 1
led_off 60 30 12bits 0.02617 0.02617
2013.02.26T04:17:30.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1
led_off 30 30 12bits 0.02617 0.02617
2013.02.26T04:27:40.000 data_management on 10 off fixed 10 3600 jpeg 3 on float 128 8 off off 255 off
2013.02.26T04:27:50.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1
led_a_on 100 30 12bits 0.0 0.0
2013.02.26T04:28:40.000 data_management off 10 off off 10 3600 off 0 off float 128 8 off off 0 off
2013.02.26T04:28:50.000 acquisition_configuration correlated_double_sampling 10 0 0 1023 1023 59 1
led_off 100 30 12bits 0.0 0.0
2013.02.26T04:52:10.000 data_management on 10 off fixed 10 3600 jpeg 0 on float 128 8 off off 255 off
2013.02.26T04:52:30.000 table_acquisition 1 9
2013.02.28T09:46:00.000 table_acquisition 0 1
2013.02.28T10:16:00.000 table_acquisition 1 9
```

Annex 1:

ANNEX 2:

C

ESA/ITI Robust Design Optimisation of Space Missions. Reduced Model Definition Report

Below is the most recent draft available to the thesis author. This document was finalised by Dr. Minisci and Prof. Vasile and submitted to ESA in 2012, where the Mechanical & Aerospace Engineering department of the University of Strathclyde was awarded an ESA ITI[1] grant.

[1] https://iti.esa.int/iti/index.jsp

Equation Chapter 1 Section 1

Robust Design Optimisation of Space Missions

....

Reduced Model Definition Report

Ref: Robust_ RMDR.doc
Issue: 3
Date : 2011/04/12

Prepared by: Eirini Komninou
Edmondo Minisci
Massimiliano Vasile

Date: 2010/07/1

Checked by: Edmondo Minisci and Massimiliano Vasile

Date: 2010/07/2

Approved by: Massimiliano Vasile

Date: 2010/07/12

Space Advanced Research Team
University of Strathclyde, James Weir South Building, G1 1XJ, Glasgow, UK
E-mail: massimiliano.vasile@strath.ac.uk

Document Change Record

Issue	**Date**	**Changes**	**Name of person**
1	*2010/07/12*	LISA Pathfinder & models description	Eirini Komninou
2	*2010/08/04*	EPS UML diagram & description update	Eirini Komninou
3	*2011/02/02*	ADS and TT&C model update	Edmondo Minisci
4	*2011/04/02*	Prop, System, ACS model updates	Massimiliano Vasile
5	*2011/04/15*	UML diagram update	Edmondo Minisci

Scope

This document is the Reduced Model Definition Report, output of WP1500. It contains a detailed description of the models developed within the present study and their adaptation to the specific case study. The models have been developed with in mind a general application and then adapted to the specific case of LISA Path Finder the reference test case of this study. The adaptation of the generic models to the specific test case contains also elements of the uncertainty model definition.

Reference documents

[P1] Introduction to LISA Pathfinder [Ref: LISA-LPF-RP-0002], ESA 2009, Paul McNamara & Giuseppe Racca

[P2] LISA Pathfinder mission ESA homepage http://sci.esa.int/science-e/www/area/index.cfm?fareaid=40

[P3] Drag-Free Attitude Control System description http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=39981

[P4] Overview of LISA Pathfinder [Ref: LISA-LPF-RP-0001], ESA 2009, Paul McNamara

Terms, definitions and abbreviated terms

Symbols

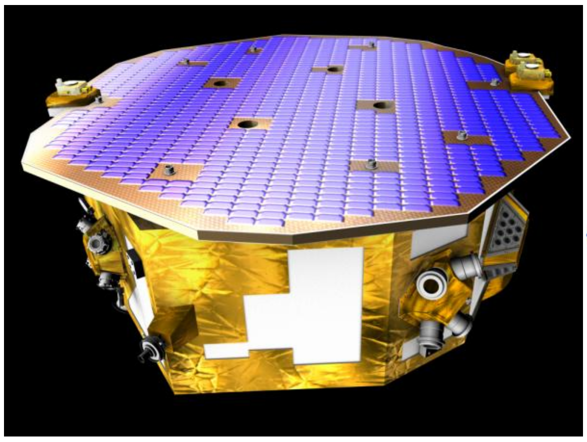
a	Semi-major axis

Abbreviations

ESA	European Space Agency
SpaceART	University of Strathclyde Space Advanced Research Team

Contents

1. Test Case: LISA Pathfinder (LPF)



1.1. Introduction

LISA Pathfinder, the second of the European Space Agency's Small Missions for Advanced Research in Technology (SMART), is a dedicated technology demonstrator for the joint ESA/NASA Laser Interferometer Space Antenna (LISA) mission. The technologies required for LISA are many and extremely challenging. This coupled with the fact that some flight hardware cannot be fully tested on ground due to Earth-induced noise, led to the implementation of the LISA Pathfinder mission to test the critical LISA technologies in a flight environment.

LISA Pathfinder essentially mimics one arm of the LISA constellation by shrinking the 5 million kilometre armlength down to a few tens of centimetres, giving up the sensitivity to gravitational waves, but keeping the measurement technology: the distance between the two test masses is measured using a laser interferometric technique similar to one aspect of the LISA interferometry system. The scientific objective of the LISA Pathfinder mission consists then of the first in-flight test of low frequency gravitational wave detection metrology.

LISA Pathfinder was first proposed in 1998 as ELITE (European LIsa TEchnology Experiment). This mission consisted of a single spacecraft in geostationary orbit with a differential acceleration goal of 10^{-14} ms^{-2} / $\sqrt{\mathrm{Hz}}$ over a frequency range of 1-100 mHz. This original proposal was refined and proposed to ESA in 2000 in response to the SMART-2 announcement of opportunity. At the time, the proposal called for a joint LISA and Darwin[1] pathfinder mission, consisting of two free-flying spacecraft, with three payloads (LISA Technology Package, Darwin Technology Package, and a US provided LISA Technology Package). The goal of the mission was to demonstrate drag-free control (for LISA) and formation flying (for Darwin). The mission was approved by the Science Programme Committee (SPC) in November 2000. After an initial industrial study, the mission was descoped to a single spacecraft (the Darwin Pathfinder was cancelled) and renamed LISA Pathfinder (LPF). At the time, LPF carried two payloads, the European built LISA Technology Package (LTP), and the US provided Disturbance Reduction System (DRS). Both payloads consisted of two inertial sensors, a laser metrology system, micro-Newton thrusters and drag free control software. However, the DRS was descoped and now consists of micro-Newton thrusters and a dedicated processor running the drag-free and attitude control software, and will use the LTP inertial sensors.

1 Darwin is a proposed mission consisting of a flotilla of four or five free-flying spacecraft that will search for Earth-like planets around other stars and analyse their atmospheres for the chemical signature of life.

LISA Pathfinder is due to be launched in 2011 on-board a dedicated small launch vehicle. The launcher selected is the new Arianespace VEGA launcher, which will launch LPF from the European spaceport of Kourou (French Guyana) into a parking orbit with perigee at 200 km, apogee at 1620 km, and an inclination to the equator of 5.3º . Since the VEGA launcher is still under development a back-up launch possibility is maintained with the Russian vehicle, Rockot, which could launch LISA Pathfinder from Plesetsk (Russia) into a parking orbit with perigee at 200 km, apogee at 900 km and an inclination to the equator of 63º . After a series of apogee raising manoeuvres using an expendable propulsion module, LISA Pathfinder will enter a transfer orbit towards the first Sun-Earth Lagrange point (L1). After separation from the propulsion module, the LPF spacecraft will be stabilised using the micro-Newton thrusters, entering a 500,000 km by 800,000 km Lissajous orbit around L1. Following the initial on-orbit check-out and instrument calibration, the in-flight demonstration of the LISA technology will then take place. The nominal lifetime of the science operations is 180 days; this includes the LTP and DRS operations, and a period of operations when the LTP will control the DRS thrusters.

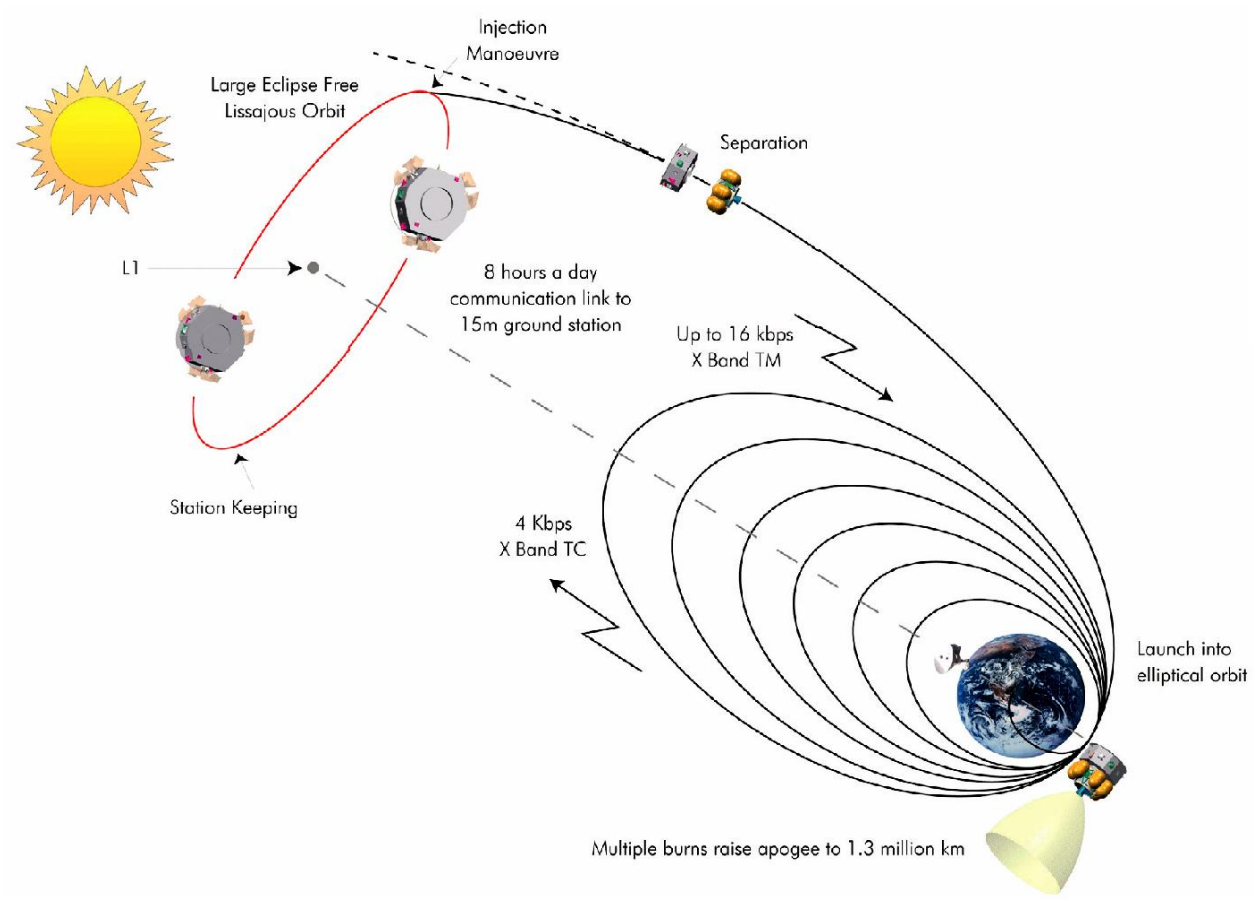

Injection Manoeuvre
Large Eclipse Free Lissajous Orbit
Separation
L1
8 hours a day communication link to 15m ground station
Up to 16 kbps X Band TM
Station Keeping
4 Kbps X Band TC
Launch into elliptical orbit
Multiple burns raise apogee to 1.3 million km

Figure 1: LPF's orbital scheme - From launch to operation

1.2. Payload

Unlike traditional observatory or planetary missions, the payload in LISA Pathfinder cannot be considered as a discrete piece of hardware carried by the spacecraft. Instead, during science operations the payload and spacecraft act as a single unit: the attitude control of the spacecraft is driven by the payload. The LISA Pathfinder spacecraft will carry two payloads, or test packages: the LISA Technology Package (LTP), provided by European institutes and industry, and the Disturbance Reduction System (DRS), provided by NASA.

LISA technology package

The LTP represents one arm of the LISA interferometer, in which the distance between the two test masses is reduced from 5 million kilometres to 35 centimetres. As in LISA, the test masses fulfil a double role: they serve as mirrors for the interferometer and as inertial references for the drag-free control system. The main role of the LTP is to house the test masses and to provide the position information of the test masses to the Drag-Free and Attitude Control System (DFACS).

The LTP DFACS consists of an inertial sensor, a proportional micro-propulsion system and a control loop. The inertial sensor subsystem is designed such that a cubic test mass located at the centre is free from all external forces except inertial gravity. The two identical test masses, each one a 46 mm cube composed of Gold:Platinum alloy, are housed in individual evacuated enclosures. The displacement of the cubes with respect to their housing is measured by capacitive sensing in three dimensions. These position signals are ed in a feedback loop to command proportional micro-propulsion thrusters to enable the spacecraft to remain centred on the proof mass (see [P3] for further details). Field Emission Electric Propulsion (FEEP) thrusters will be used as the micro-propulsion elements.

The LISA Technology Package is the European-provided payload onboard LISA Pathfinder. The instrument is being built by a consortium of European National Agencies and ESA (see table below for further information). The subsystems will be integrated and tested under the control of Astrium GmbH, Germany (the LISA Pathfinder Architect). The fully integrated technology package will then be integrated into the LISA Pathfinder spacecraft under the control of Astrium UK Ltd., the Industrial Prime Contractor.

European contributions to the LISA Technology Package

Country	Institute/Industry	Responsibility
France	APC Université Paris, with Oerlikon (CH)	Laser Modulator
Germany	AEI, Hannover	Co-PI, Interferometer design
	Astrium GmbH	LTP Architect
	Tesat (D) / Max Planck Albert Einstein Institute	Reference Laser Unit
Italy	University of Trento	PI, Inertial Sensor Design
	Carlo Gavazzi Space	Inertial Sensor Subsystem
	Thales Alenia Space	Test Mass Electrode Housing
The Netherlands	SRON	ISS Check Out Equipment
Spain	University of Barcelona	Data Management Unit
		Data Diagnostic System
Switzerland	ETH Zurich/ Oerlikon (CH)	ISS Front End Electronics
United Kingdom	University of Birmingham	Phasemeter Assembly
	University of Glasgow	Optical Bench Interferometer
	Imperial College London	Charge Management System
ESA	Thales Alenia Space (IT)	Caging Mechanism
	Astrium GmbH (DE)	LTP Architect

1.3. Disturbance Reduction System

The Disturbance Reduction System (DRS) will validate system-level technologies required for use on 'drag-free' spacecrafts, that is, spacecrafts that are controlled to follow a trajectory determined only by external gravitational forces (a geodesic). The DRS eliminates other forces, such as solar radiation pressure, that would disturb the trajectory.

The DRS is a NASA-supplied system, which contributes to the LISA Pathfinder mission goals and uses the European LTP as a gravitational sensor. The DRS actuator consists of two clusters of colloidal thrusters that use ionised droplets of a colloidal solution accelerated in an electric field to provide micro-propulsion, and drag-free control software residing on a dedicated computer. The DRS will use the sensor information from the LTP (test mass positions and attitude) to control the spacecraft position and attitude.

1.4. Subsystems

1.4.1. Structure

The Science Spacecraft platform structure provides the mechanical support for the hardware of the other spacecraft subsystems. The spacecraft has a shape of an octagonal prism. The outer diameter is 2.31 m and the height 0.96 m. One of the two bases is covered by a sunshield panel supporting an array of triple junction GaAs solar cells of 2.8 m^2 , providing at end-of-life 650 W of power, while the other base interfaces with the propulsion module. A large central cylinder accommodates the LTP Core Assembly, while the rest of the payload equipment and the spacecraft units are mounted as far away as possible on shear walls connecting the central cylinder to the outer panel forming the octagonal structure. The cylinder and all structural panels are constructed from sandwich panels or shells with carbon fibre laminate skins bonded to aluminium honeycomb core. Aluminium items are limited to structural rings, cleats, inserts and minor brackets.

1.4.2. Thermal Control

The Thermal Control Subsystem must guarantee the very stable thermal environment required by the science measurements. Together with the stringent thermal stability required at LTP level, a stable thermal environment of 10^{-3} K/ $\sqrt{\text{Hz}}$ is also required at the LTP interface, in order to minimise the thermoelastic distortions. Passive means are used to control the upper temperatures of sensitive equipments, with electrical heaters to control the lower temperatures. The entire module is wrapped in Multi-Layer Insulation (MLI) except for designated radiator areas designed to reject to space the excessive heat. The minimum necessary heater power is applied in the cold cases so that the lower temperature of each unit is maintained towards the bottom of their allowable range. By using the full design temperature range of each unit in this way, the heater power requirement is minimised. Heater switching is not permitted during the nominal science operations as the transient variations in temperature that happen as heaters switch can interfere with the payload measurements. On the sensitive equipment, different combinations of trimming heaters are used to obtain the required temperatures. On the micropropulsion systems (both FEEP and CMNT) a high frequency pulse width modulation control of the heaters is used. Heat pipes are avoided as they too would interfere with the measurements due to gravitational disturbance caused by the transfer of mass through the pipes.

1.4.3. Attitude and Orbit Control Subsystem

The Science Module is a three-axis controlled spacecraft. Apart from the DFACS -described right below- an Attitude and Orbit Control Subsystem (AOCS) is needed in order to control the Launch Composite (when the propulsion module is attached to the science module) and for the on-station phases when the drag-free conditions cannot be maintained. The AOCS uses as sensors two Autonomous Star Trackers, two Digital Sun Sensors (DSS) for sun acquisition and in safe mode and two Fibre-Optic Gyroscopes Units, required for certain phases where the optical sensors are inoperable due to high rates or eclipse. They are also needed to provide high bandwidth data during the main engine burn. The AOCS actuators are 4 pairs of 10 N bi-propellant thrusters located on the propulsion module and three clusters of 4 FEEP thrusters each located on the sides of the science module.

1.4.4. Drag-Free Attitude Control System

The Drag-Free Attitude Control System (DFACS) is probably the single most important subsystem on board. The main objective of the DFACS is to control the spacecraft dynamics is such a way that the main requirement on the residual acceleration is met. Like any control system, DFACS makes use of sensors and actuators. The spacecraft attitude is sensed by means of a pair of star trackers with a measurement error of 32 arcsec/ $\sqrt{\text{Hz}}$.

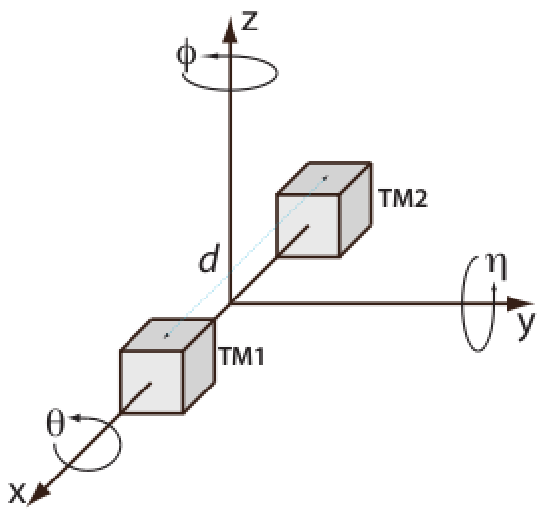


Figure 2: LTP Test Masses layout schematic and axis rotation angles notation

With reference to Image 6, the test masses position (x_i , y_i , z_i with i = 1, 2) and attitude (θ_i , η_i , φ_i with i = 1, 2) with respect to their housing inside the inertial sensor, are sensed through two different means:

electrostatic readout based on capacitance electronic measurement with a measurement noise of 1.8 nm/ √Hz for x, y, z and 200 nrad/ √Hz for θ, η, φ over the measurement bandwidth.

optical readout based on laser interferometric measurement, with a noise of 9 pm/ √Hz for x_1 and $x_1 - x_2$ and 20 nrad/ √Hz for η_1 , φ_1 , η_{1-2} , φ_{1-2} over the measurement bandwidth.

The actuation on the spacecraft attitude is performed by the set of micropropulsion thrusters with a noise of 0.1 μN/ √Hz.

The forces and torques on the test masses are provided by the inertial sensor electrostatic actuation. During the science modes, the maximum differential acceleration noise allowed on the TM's is 10^{-14} ms^{-2} / √Hz.

The DFACS tasks is to control the 15 degrees of freedom (DOF) present on board (6 DOF per test mass and 3 DOF for the attitude of the spacecraft) to fulfil the following objectives:

to shield one test mass - the drag-free or free floating test mass - from external disturbances along its sensitive axis in the measurement bandwidth [1 mHz to 30 mHz]. The spacecraft is therefore controlled to follow the drag-free test mass, which is free to float in its housing, only subject to the forces that directly impact on the test mass (e.g. thermal radiation pressure, magnetic field interaction with TM), called internal forces fi and to the non contact coupling with the electrodes housing (e.g. due to electrostatic field and gravity gradient), called parasitic stiffness ω_i^2 .

to measure by laser interferometer the differential acceleration between the drag-free test mass and the second test mass maintained centred in its housing using electrostatic capacitive forces.

to keep the spacecraft pointed to the sun and the fixed communication antenna pointed to Earth.

This is realised by careful selection of the drag-free degrees of freedom and frequency band separation.

1.4.5. Micropropulsion:

The LISA Pathfinder Micro-Propulsion Subsystem (MPS) is based on Field Emission Electric Propulsion (FEEP) technology.

In field emission electrical propulsion, positive ions are directly extracted from liquid metals (for LISA Pathfinder, Caesium has been chosen as the liquid metal source, however a back-up option of using Indium is also being developed in Europe) and accelerated by means of electrostatic force in high vacuum. This function is carried out by applying a very high voltage to a suitable electrode configuration, which is able to create and enhance very high electrical fields (up to 10^9 V/m).

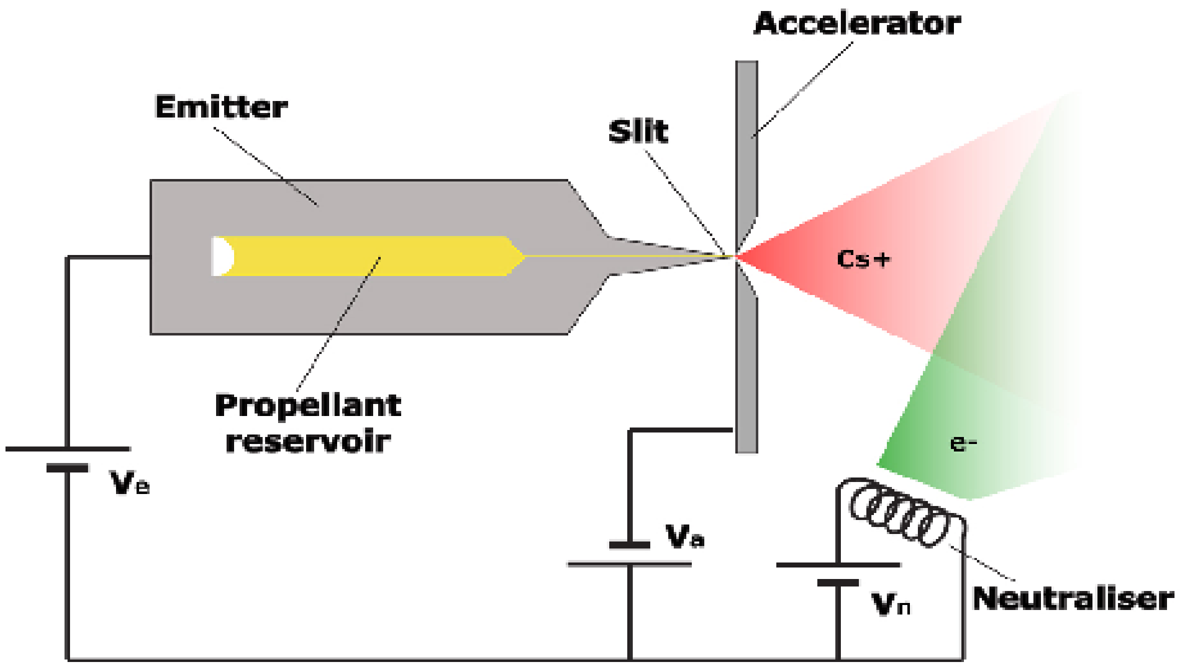


Figure 3: FEEP propulsion concept

The FEEP working principle is given in Image 7. An additional external source of electrons, the neutraliser, needs to be included to maintain the balance of the overall electrical charge of the system (ions$^+$ = e$^-$).

The LISA Pathfinder MPS is composed of three main parts, called Micro Propulsion Assembly (MPA): each one consisting of one FEEP Cluster Assembly, one Power Control Unit (PCU) and one Neutraliser Assembly (NA). The FEEP Cluster Assembly (Image 8) consists of a self-contained unit of 4 FEEP Thruster Assemblies, which include propellant reservoir, mounted on a support structure. The four thrusters are devoted to provide thrust to the required vector directions and are commanded individually and work in hot redundancy.

1.4.6. Communications system

The communications subsystem works at X-band frequency (7230 MHz uplink and 8495 MHz downlink) and provides for commanding and housekeeping telemetry during LEOP, transfer and on-station and also transmits science data telemetry whilst on-station. For LEOP, some phases of transfer and during on-station anomalies, omni directional coverage is required. Two hemispherical antennas are used for the omnidirectional coverage allowing a maximum data rate of 60 ksymbols/s during LEOP and 1 ksymbols/s at L1. However to achieve the required telemetry data rate on-station at L1 distance, a medium gain horn antenna is used capable to transmit 120 ksymbols/s. Two X-band transponders are present for redundancy, with coherent ranging. Additional amplification is required at the output of the transponders to achieve the required telemetry margins.

1.4.7. On-Board Computer

The On-Board Computer (OBC) is the central control unit for all on board data handling activities, the attitude and orbit control subsystem (AOCS), the DFACS and the management of the platform and payload equipments. Data Handling functions mainly constitute of command distribution, telemetry acquisition and timing facilities during all phases of the mission. Furthermore the OBC performs monitoring functions and depending the detection of failures provides safe system reconfiguration capabilities. The OBC consists of several independent redundant modules: two processor modules, each based on a single chip ERC32 central processing unit working at 22.5 MHz with 6MB RAM, 1.5 MB EEPROM and 64 KB PROM; two telecommand, telemetry and reconfiguration units containing 400 KB safeguard memory; four actuator and sensors interface modules and two mass memory units of 12.5 Gb.

The OBC interfaces with the spacecraft units and the LTP through a MIL-bus 1553B, while the internal backplane link uses the Space-Wire standard. The OBC hosts the On-Board Software (OBSW) which performs all the spacecraft and most of the LTP functions.

1.4.8. Power System

The power subsystem provides a stable regulated bus, with voltage regulation at 28 VDC. During the technology demonstration phase the spacecraft will be sun pointing, therefore all electrical power can be generated by the solar array. Battery power is only required during launch and early operations (LEOP), for eclipse periods during the transfer, during slews to and from engine firing attitude, while firing the engine, and for any anomalies during the on-station phase. For a nominal mission the battery is only required during the initial phases of the mission, therefore because of its low mass, and simple management, a Li-ion battery providing 400 Wh is used.

2. SpaceART subsystem model definition

2.1. Introduction

The Space ART (Advanced Research Team) of the University of Strathclyde is currently developing subsystem software models (using Matlab®) of all the fundamental subsystems on board an unmanned spacecraft.

ACS (Attitude Control System), EPS (Electrical Power System)&HARN (Harness), TT&C (Telemetry Tracking & Commanding) system, S&M (Structures & Mechanisms) system, C&DH (Command & Data Handling) system, Thermal system and PROP (Propulsion) system are found in all different kinds of unmanned spacecrafts, thus considered fundamental for every mission.

The SpaceART's current work aims at implementing a set of subsystem models which are able to produce results (model outputs) accurate enough to allow a designer to predict her/his subsystem's mass and power budget accurately enough for a pre-phase A study.

Input parameters can be divided into the following categories:

Design parameters

Fixed parameters

Uncertain parameters

Design parameters are defined as input parameters which can be decided by the designer(s).

Fixed parameters are defined as input parameters which are already known or given, therefore are considered constants through the design process.

Uncertain parameters are defined as input parameters which cannot be deterministically defined. Such parameters can either be design or fixed ones which cannot be defined with absolute certainty.

Each input triggers a specific function, thus retrieving essential data like sensor technical characteristics for example. The retrieved data is then processed using mathematical models simulating various aspects of the subsystem model in question. The outputs produced by the software model are able to predict the physical attributes of the subsystem the designer(s) is dealing with. As seen in the following chapters, the common outputs produced by all subsystem software models are

the overall estimated mass

the overall power consumption

Data handling systems produce an extra output

the overall data output or data rate output.

Some output parameters will actually remain internal, meaning that they will be fed as inputs to other systems, thus contributing to sizing some or all interfacing systems.

For example, every satellite subsystem model produces a power consumption output parameter among others. All subsystems' power consumption output parameters are fed to the EPS model, thus sizing its parameters (EPS weight, solar array area, solar cell connection topology, battery capacity). Therefore, the power consumption output of each individual system is consider an internal output.

In all cases (excluding the C&DH), Veps and Ieps (EPS outputs) which appear as input parameters passed to each individual system, are logical interconnections displaying the connection between the EPS system and every other spacecraft system.

In the C&DH system, Ieps is a sizing parameter of the C&DH Power Regulator block.

2.2. UML models

Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of software engineering. The standard is managed, and was created by, the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software-intensive systems.

SpaceART used the advantages of UML to create the blueprint of each subsystem's software model that is currently under development. There is a total of 10 UML models available, depicting the aforementioned fundamental subsystems' software structure plus a model describing how all subsystem models are linked overall.

In the remainder of this report, one can find the description of all the subsystem models developed within this study. At first we will present the overall system model with the links and connections among subsystems and a mathematical treatment of the integration between Thermal, Power and Propulsion.

Satellite subsystems general overview:

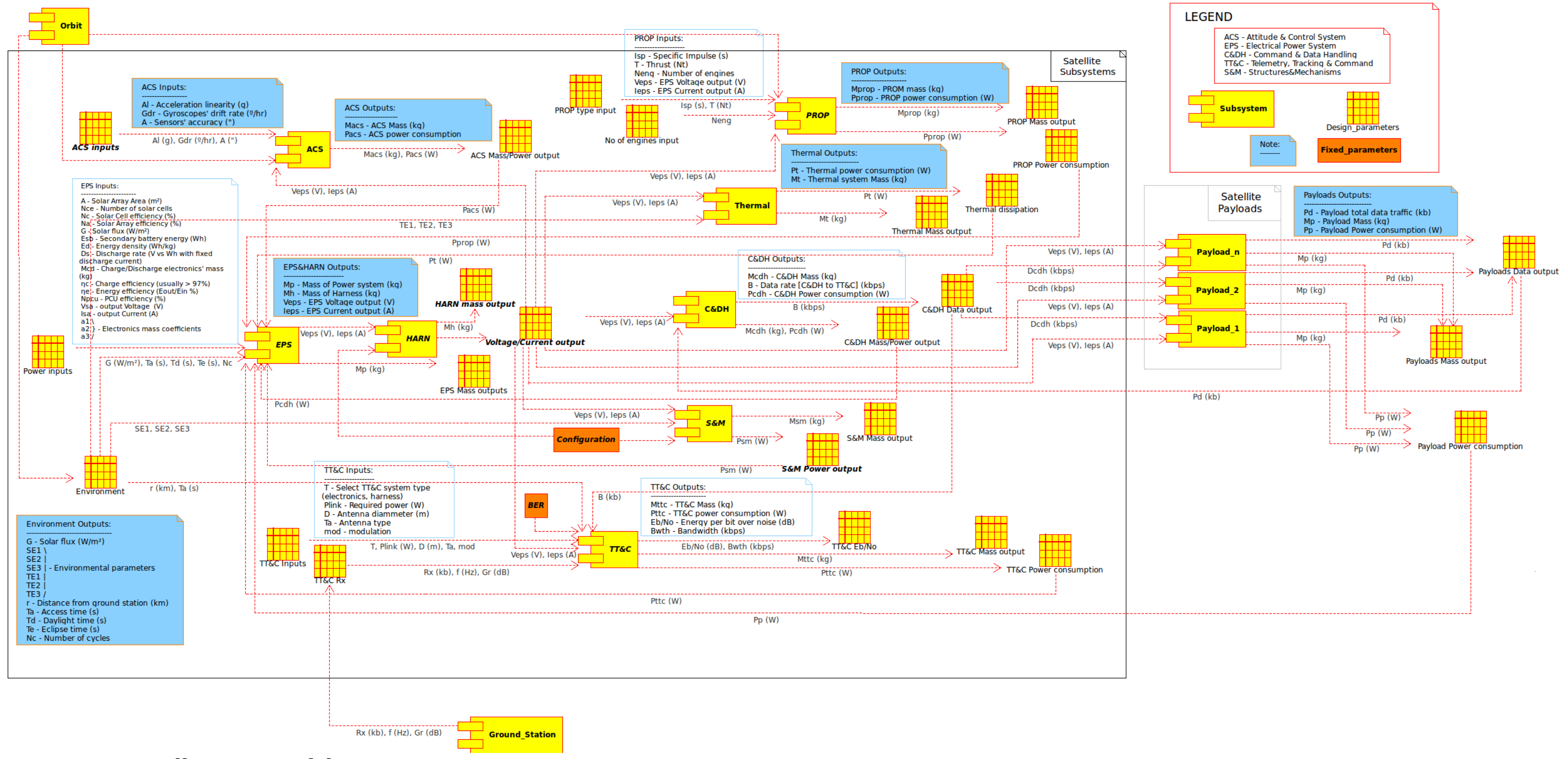

LEGEND
ACS - Attitude & Control System
EPS - Electrical Power System
C&DH - Command & Data Handling
TT&C - Telemetry, Tracking & Command
S&M - Structures&Mechanisms
Subsystem
Design_parameters
Note:
Fixed_parameters
Satellite Subsystems
Satellite Payloads
Orbit
ACS inputs
ACS Inputs:
Al - Acceleration linearity (g)
Gdr - Gyroscopes' drift rate (º/hr)
A - Sensors' accuracy (")
Al (g), Gdr (º/hr), A (")
ACS
ACS Outputs:
Macs - ACS Mass (kg)
Pacs - ACS power consumption
Macs (kg), Pacs (W)
ACS Mass/Power output
PROP type input
No of engines input
PROP Inputs:
Isp - Specific Impulse (s)
T - Thrust (Nt)
Neng - Number of engines
Veps - EPS Voltage output (V)
Ieps - EPS Current output (A)
Isp (s), T (Nt)
Neng
PROP
PROP Outputs:
Mprop - PROM mass (kg)
Pprop - PROP power consumption (W)
Mprop (kg)
Pprop (W)
PROP Mass output
PROP Power consumption
Thermal
Thermal Outputs:
Pt - Thermal power consumption (W)
Mt - Thermal system Mass (kg)
Pt (W)
Mt (kg)
Thermal Mass output
Thermal dissipation
EPS Inputs:
A - Solar Array Area (m²)
Nce - Number of solar cells
Nc - Solar Cell efficiency (%)
Na - Solar Array efficiency (%)
G - Solar flux (W/m²)
Esb - Secondary battery energy (Wh)
Ed - Energy density (Wh/kg)
Ds - Discharge rate (V vs Wh with fixed discharge current)
Mcd - Charge/Discharge electronics' mass (kg)
ηc - Charge efficiency (usually > 97%)
ηe - Energy efficiency (Eout/Ein %)
Npcu - PCU efficiency (%)
Vsa - output Voltage (V)
Isa - output Current (A)
a1 \
a2 } - Electronics mass coefficients
a3 /
Power inputs
G (W/m²), Ta (s), Td (s), Te (s), Nc
EPS
EPS&HARN Outputs:
Mp - Mass of Power system (kg)
Mh - Mass of Harness (kg)
Veps - EPS Voltage output (V)
Ieps - EPS Current output (A)
HARN
HARN mass output
Mh (kg)
Voltage/Current output
EPS Mass outputs
Mp (kg)
Pacs (W)
TE1, TE2, TE3
Pprop (W)
Pt (W)
Veps (V), Ieps (A)
Pcdh (W)
C&DH
C&DH Outputs:
Mcdh - C&DH Mass (kg)
B - Data rate [C&DH to TT&C] (kbps)
Pcdh - C&DH Power consumption (W)
B (kbps)
Mcdh (kg), Pcdh (W)
C&DH Data output
C&DH Mass/Power output
S&M
Configuration
Msm (kg)
Psm (W)
S&M Mass output
S&M Power output
Environment
SE1, SE2, SE3
r (km), Ta (s)
Environment Outputs:
G - Solar flux (W/m²)
SE1 \
SE2 |
SE3 | - Environmental parameters
TE1 |
TE2 |
TE3 /
r - Distance from ground station (km)
Ta - Access time (s)
Td - Daylight time (s)
Te - Eclipse time (s)
Nc - Number of cycles
TT&C Inputs:
T - Select TT&C system type (electronics, harness)
Plink - Required power (W)
D - Antenna diammeter (m)
Ta - Antenna type
mod - modulation
TT&C Inputs
TT&C Rx
T, Plink (W), D (m), Ta, mod
Rx (kb), f (Hz), Gr (dB)
BER
B (kb)
TT&C
TT&C Outputs:
Mttc - TT&C Mass (kg)
Pttc - TT&C power consumption (W)
Eb/No - Energy per bit over noise (dB)
Bwth - Bandwidth (kbps)
Eb/No (dB), Bwth (kbps)
Mttc (kg)
Pttc (W)
TT&C Eb/No
TT&C Mass output
TT&C Power consumption
Pp (W)
Ground_Station
Payload_n
Payload_2
Payload_1
Payloads Outputs:
Pd - Payload total data traffic (kb)
Mp - Payload Mass (kg)
Pp - Payload Power consumption (W)
Dcdh (kbps)
Pd (kb)
Payloads Data output
Payloads Mass output
Payload Power consumption

Figure 4. Overall system model

Figure 4 shows the general overview diagram depicting the subsystems' interconnections. The diagram contains 8 blocks inside the Satellite Subsystem area. The Satellite Subsystem area is what we consider the satellite design space. Everything that is outside that area is either a given input, or an environmental parameter or an output of the design process. In particular, in the following we will consider the ground segment and Mission Analysis (the orbit block) as external to the system design process. Most of the environmental parameters will come from the Orbit and Ground Station blocks.

This diagram aims in demonstrating the interconnection between various subsystems' inputs and outputs, leading to a complete generic spacecraft model. With a total of 90 input parameters (internal and external) and the mass, power consumption and data budget of each subsystem as outputs, this model is able to produce both a set of individual budgets as well as an overall budget of all spacecraft subsystems constituting the space segment.

The inputs of each subsystem may either be external factors such as Environmental, Orbital, Ground station parameters or other subsystems' outputs for example the EPS Voltage and Current outputs are fed as inputs to all other subsystems and possible payloads.

Like mentioned before, the outputs of each block i.e subsystem are the expected subsystem mass as well as the expected power consumption. In data handling systems an additional output is produced, representing the data rate or total data traffic of a subsystem.

3. Using LPF as test case for SpaceART general overview:

The LPF model, developed here, contains eight subsystems each one represented by one of the SpaceART subsystem models. In particular the LPF satellite model contains:

An AOCS system and a DFACS system both represented by the SpaceART ACS model.

A Power system and harnessing both represented by the SpaceART EPS and HARN model.

An OBC system represented by the SpaceART C&DH model

A Structure system represented by the SpaceART S&M model

A Communications system represented by the SpaceART TT&C model

A Thermal control system represented by the SpaceART Thermal model

A Propulsion (chemical) and two micropropulsion (electrical) systems, all represented by the SpaceART PROP model

All LISA Pathfinder payloads' mass and power consumption will be added to the SpaceART general overview model's outputs in order to estimate the overall satellite mass and power consumption.

4. Subsystem Integration

The mass of the spacecraft is made of the sum of the individual masses of all the subsystems. Therefore, if the mass of the subsystem was not interdependent, one could compute separately the mass of each subsystem. However, a number of subsystems are deeply interrelated and cannot be considered separately. This deep link would imply an iterative process over the value of the mass which can make the overall robust optimization more complicated.

It was then decided to adopt a different approach. The mass of the spacecraft is divided in two components: a normalised one, called Ψ, and an absolute one, called Φ. Each subsystem will produce a normalized and an absolute mass component, such that the overall mass of the spacecraft can be computed as follows (NOTE: this is an example only with thermal, power and propulsion system assuming an electric propulsion system).

The mass of some subsystems depends on the overall mass of the spacecraft, like the mass of propellant for example:

$$m_p = \left(1-e^B\right)m_{s/c} \tag{1}$$

$$m_T = \gamma_T m_p = \gamma_T\left(1-e^B\right)m_{s/c} \tag{2}$$

$$W_e = \frac{a_c I_{sp}}{2\eta_e} m_{s/c} \tag{3}$$

$$m_e = \gamma_e W_e = \gamma_e \frac{a_c I_{sp}}{2\eta_e} m_{s/c} \tag{4}$$

The mass of the propulsion system is therefore:

$$m_{prop} = \left(1-e^B\right)m_{s/c} + \gamma_T\left(1-e^B\right)m_{s/c} + \gamma_e \frac{a_c I_{sp}}{2\eta_e} m_{s/c} = \left[\left(1-e^B\right) + \gamma_T\left(1-e^B\right) + \gamma_e \frac{a_c I_{sp}}{2\eta_e}\right] m_{s/c} \tag{5}$$

The mass of the thermal control system is the sum of the mass of the radiator plus the sum of the mass of the thermal links (the mass of the heaters is neglected here).
If one takes the hot case, gets:

$$Q_i + (1-\eta_e)W_e + R_1\Delta T_1 + R_2\Delta T_2 - R_3\Delta T_3 = 0 \tag{6}$$

$$-\sigma A_R \varepsilon_R T_R^4 + R_3\Delta T_3 = 0 \tag{7}$$

from which the radiator is:

$$A_R = \frac{Q_i + R_1\Delta T_1 + R_2\Delta T_2}{\sigma\varepsilon_R T_R^4} + \frac{(1-\eta_e)W_e}{\sigma\varepsilon_R T_R^4} = \frac{Q_i + R_1\Delta T_1 + R_2\Delta T_2}{\sigma\varepsilon_R T_R^4} + \frac{(1-\eta_e)a_c I_{sp}}{\sigma\varepsilon_R T_R^4 2\eta_e} m_{s/c} \tag{8}$$

The mass of the thermal links is simply:

$$m_l = \sum_{i=1}^{3} \rho_l \frac{R_i l^2}{k} \tag{9}$$

Let's now call $K_d = Q_i + R_1\Delta T_1 + R_2\Delta T_2$ such that:

$$A_R = \frac{K_d}{\sigma\varepsilon_R T_R^4} + \frac{(1-\eta_e)a_c I_{sp}}{\sigma\varepsilon_R T_R^4 2\eta_e} m_{s/c} \tag{10}$$

wih mass:

$$m_R = \rho_R\left(\frac{K_d}{\sigma\varepsilon_R T_R^4} + \frac{(1-\eta_e)a_c I_{sp}}{\sigma\varepsilon_R T_R^4 2\eta_e} m_{s/c}\right) \tag{11}$$

In the cold case we have:

$$Q_H = -\left[Q_i + R_1\Delta T_1 + R_2\Delta T_2 - \sigma A_R \varepsilon_R T_R^4\right] = \\ -\left[Q_i + R_1\Delta T_1 + R_2\Delta T_2 - K_d - \frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c}\right] \tag{12}$$

And if one calls $K_e = Q_i + R_1\Delta T_1 + R_2\Delta T_2$ the heat required in the cold case is:

$$Q_H = -\left[K_e - K_d - \frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c}\right] \tag{13}$$

The mass of the power system is the mass of the solar arrays plus the one of the batteries:

$$P_{SA} = \frac{[P_d + W_e]}{X_d} + \frac{P_e + Q_H}{X_e}\frac{T_e}{T_d} = \frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{W_e}{X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d} = \\ \frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d} \tag{14}$$

The mass of the solar arrays is:

(15)

$$m_{SA} = \frac{\rho_{SA}}{P_{EOL}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) \tag{16}$$

The mass of the batteries comes from the total energy capacity:

$$C_B = \frac{(P_e + Q_H)T_e}{DOD\eta_B} \tag{17}$$

which gives:

$$m_B = \gamma_B \frac{P_e T_e}{DOD\eta_B} + \gamma_B \frac{Q_H T_e}{DOD\eta_B} \tag{18}$$

If one then adds the mass of the PPU and harness:

$$m_{PPU} = \frac{aP_{SA}}{\eta_{PPU}} = \frac{a}{\eta_{PPU}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) \\ m_H = \gamma_{PPU}\frac{P_{SA}}{\eta_{PPU}} = \frac{\gamma_{PPU}}{\eta_{PPU}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) \tag{19}$$

The total mass of the power system is therefore:

$$m_{PW} = \frac{\rho_{SA}}{P_{EOL}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) + \gamma_B \frac{P_e T_e}{DOD\eta_B} + \gamma_B \frac{Q_H T_e}{DOD\eta_B} + \\ \frac{a}{\eta_{PPU}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) + \frac{\gamma_{PPU}}{\eta_{PPU}}\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d} + \frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) \tag{20}$$

Which can be rewritten as:

$$m_{PW} = \left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d}\right) + \gamma_B \frac{P_e T_e}{DOD\eta_B} + \\ \left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{Q_H}{X_e}\frac{T_e}{T_d}\right) + \gamma_B \frac{Q_H T_e}{DOD\eta_B} \tag{21}$$

Let's now substitute the expression for Q_H:

$$m_{PW} = \left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d}\right) + \gamma_B \frac{P_e T_e}{DOD\eta_B} +$$
$$\left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{-K_e - K_d - \frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c}}{X_e}\frac{T_e}{T_d}\right) + \tag{22}$$
$$\gamma_B \frac{-\left[K_e - K_d - \frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c}\right] T_e}{DOD\eta_B}$$

$$m_{PW} = \left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{P_d}{X_d} + \frac{P_e}{X_e}\frac{T_e}{T_d}\right) + \gamma_B \frac{P_e T_e}{DOD\eta_B} +$$
$$\left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{-K_e - K_d}{X_e}\frac{T_e}{T_d}\right) + \left(\frac{a}{\eta_{PPU}} + \frac{\gamma_{PPU}}{\eta_{PPU}} + \frac{\rho_{SA}}{P_{EOL}}\right)\left(\frac{a_c I_{sp} m_{s/c}}{2\eta_e X_d} + \frac{-\frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c}}{X_e}\frac{T_e}{T_d}\right) + \tag{23}$$
$$\gamma_B \frac{-[K_e - K_d]T_e}{DOD\eta_B} + \gamma_B \frac{\frac{(1-\eta_e)a_c I_{sp}}{2\eta_e} m_{s/c} T_e}{DOD\eta_B}$$

Now if for every subsystem one calls Φ the components without $m_{s/c}$ and Ψ the components multiplying $m_{s/c}$ one gets:

$$m_{s/c} = \Phi + \Psi m_{s/c} \tag{24}$$

from which the mass of the spacecraft is:

$$m_{s/c} = \frac{\Phi}{1-\Psi} \tag{25}$$

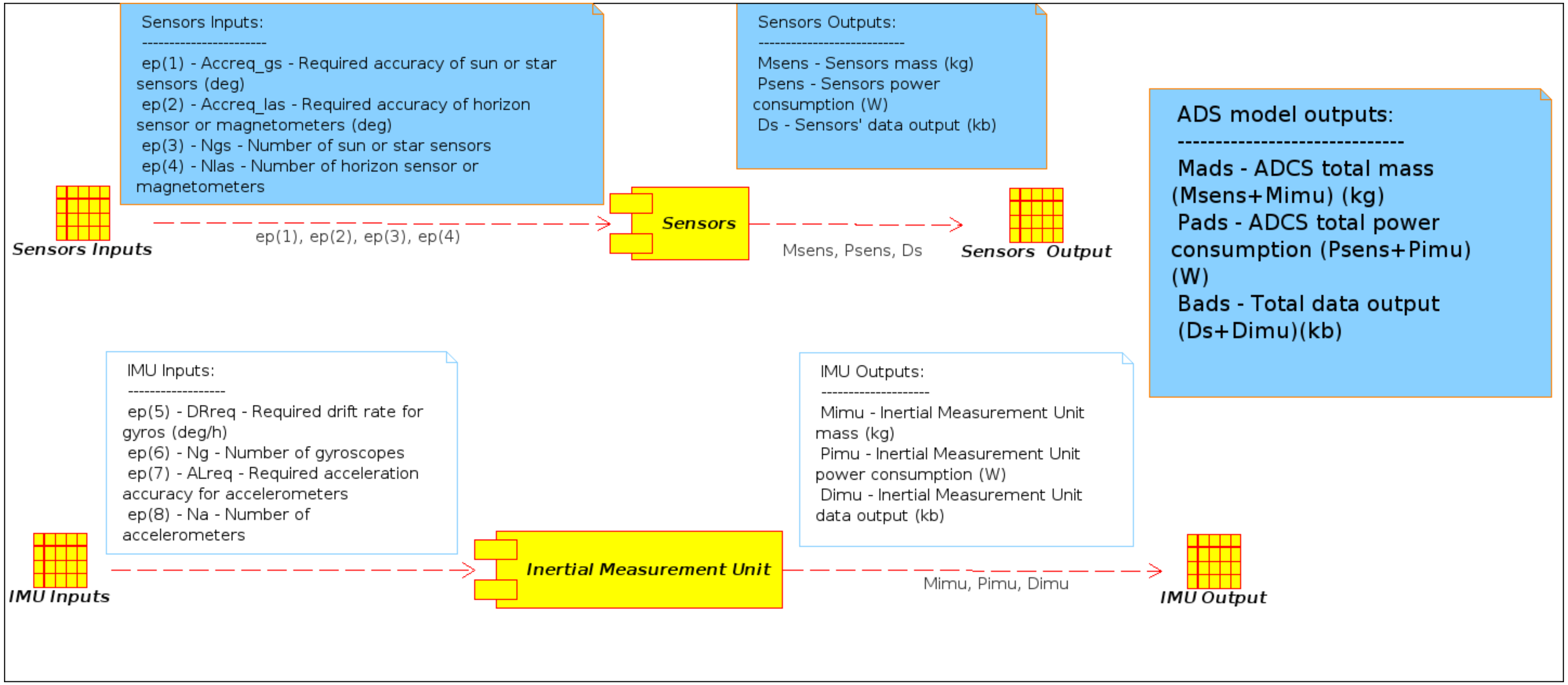
This equation states that for some combinations of parameters the mass of the spacecraft is not a real mass or it is infinite.

In the remainder of this document, each subsystem model will be described in details, following the sequence:

- ADS
- ACS
- C&DH
- Structure and Mechanisms
- Power+Harness
- Propulsion
- Thermal
- TT&C

For each one we will provide a general description, plus the adaptation to the case of LPF.

5. Attitude Determination Subsystem: UML diagram


Sensors Inputs:
----
ep(1) - Accreq_gs - Required accuracy of sun or star sensors (deg)
ep(2) - Accreq_las - Required accuracy of horizon sensor or magnetometers (deg)
ep(3) - Ngs - Number of sun or star sensors
ep(4) - Nlas - Number of horizon sensor or magnetometers
Sensors Outputs:

Msens - Sensors mass (kg)
Psens - Sensors power consumption (W)
Ds - Sensors' data output (kb)
ADS model outputs:

Mads - ADCS total mass (Msens+Mimu) (kg)
Pads - ADCS total power consumption (Psens+Pimu) (W)
Bads - Total data output (Ds+Dimu)(kb)
Sensors Inputs
ep(1), ep(2), ep(3), ep(4)
Sensors
Msens, Psens, Ds
Sensors Output
IMU Inputs:
----
ep(5) - DRreq - Required drift rate for gyros (deg/h)
ep(6) - Ng - Number of gyroscopes
ep(7) - ALreq - Required acceleration accuracy for accelerometers
ep(8) - Na - Number of accelerometers
IMU Outputs:

Mimu - Inertial Measurement Unit mass (kg)
Pimu - Inertial Measurement Unit power consumption (W)
Dimu - Inertial Measurement Unit data output (kb)
IMU Inputs
Inertial Measurement Unit
Mimu, Pimu, Dimu
IMU Output

Figure 5ADS UML diagram

According to the ADS UML diagram in Figure 5, the ADS model consists of 2 blocks, each one representing a separate physical unit. There is a total of 8 external input parameters for the ADS.

5.1. *Sensors:*

The Sensors inputs consist of:

Accreq_gs - Required accuracy of sun or star sensors (deg)

Accreq_las - Required accuracy of horizon sensor or magnetometers (deg)

Ngs - Number of sun or star sensors

Nlas - Number of horizon sensor or magnetometers

The output parameters are:

Msens – Sensors' total mass in kilograms (kg)

Psens – Sensors' total power consumption in Watts (W)

Ds – Sensors' total data output in kilobytes (kb)

5.2. *Inertial Measurement Unit (IMU):*

The IMU inputs consist of:

DRreq - Required drift rate for gyros (deg/h)
Ng - Number of gyroscopes
ALreq - Required acceleration accuracy for accelerometers
Na - Number of accelerometers

The output parameters are:

Mimu – IMU mass in kilograms (kg)
Pimu – IMU power consumption in Watts (W)
Dimu – IMU data output in kilobytes (kb)

ADS's external outputs:

The ADS's external outputs are:

Mads – Total ADS mass output i.e the sum of Msens+Mimu+Mpu (kg)
Pads – Total ADS power consumption i.e the sum of Psens+Pimu+Ppu (W)
Bads – Total ADS data output (kb)

5.3. *SpaceART ADS model function*

```
%% Attitude Determination and Control System
%
% [Mads,Pads,Bads] = ads(x,ep,flag)
%
%
%    DESCRIPTION
%    Attitude Determination System (ADS) model. It computes
%    the ADS mass, power consumption and data output based on the specified inputs found below.
%
%    INPUT
%    x - All the design and uncertain parameters
%
%      ep - All the environmental parameters (i.e inputs which are fixed, that are neither design nor uncertain
parameters)
%      ep(1) = Required accuracy of sun or star sensors (deg)
%      ep(2) = Required accuracy of horizont sensor or magnetometers (deg)
%      ep(3) = Ngs - Number of sun or star sensors
%      ep(4) = Nlas - Number of horizont sensor or magnetometers
%      ep(5) = DRreq - Required drift rate for gyros (deg/h)
%      ep(6) = Ng - Number of gyros
%      ep(7) = ALreq - Required acceleration accuracy for accelerometers (g)
%      ep(8) = Na - Number of accelerometers

%    flag - Flag used to change between system types
%
%
%    OUTPUT
%    Mads - ADS mass (kg)
%    Pads - ADS power consumption (W)
%    Bads - ADS data output (kb)
%
%    FUNCTION CALLS
%         none
%
%
%% REFERENCES
%
%      Based on: The ACS UML and UML description
% [1] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz,
%    Microcosm Press & Kluwer Academic Publishers
% [2] "Space System Design", lecture slides by Dr. Massimiliano Vasile


%% Internal functions called
[Msens,Psens,Ds] = Sensors(Accreq_gs,Accreq_las,Ngs,Nlas);

[Mimu,Pimu,Dimu] = IMU(DRreq,Ng,ALreq,Na);


%% Total Mass, Power, Data
```

Madcs = Msens+Mimu

Padcs = Psens+Pimu

Badcs = Ds+Dimu

As seen above, the ads.m is divided into individual functions, each one representing a logical block of the UML model I.e a hardware unit of the actual physical system.

5.4. *SpaceART ADS model description*

5.4.1. Sensors

[Msens,Psens,Ds] = Sensors(Accreq_gs,Accreq_las,Ngs,Nlas);

The "Sensors" routine models the mass, power consumption and output data-rate mass of different types of sensors based on their respective accuracy. Two groups of sensors are considered: a) sensors of general applicability, which are sun and star sensors, and b) sensors which can be used only for low orbits, such as horizon sensors and magnetometers. For each group, a data-base is built on the basis of current products available on the market. A polynomial regression function is used to obtain the mass, the power consumption and the output data rate on the basis of required accuracies.

In particular, a polynomial regression is performed on data shown in Table 1, to obtain the mass M_{Gsens}, the power consumption P_{Gsens}, and the output data-rate D_{Gsens} of general applicability sensors as a function of the required accuracy $Areq_{\text{GS}}$.

.

In a similar manner, a polynomial regression is performed on data shown in Table 2, to obtain the mass M_{LOsens}, the power consumption P_{LOsens}, and the output data-rate D_{LOsens} of low orbit sensors as a function of the required accuracy $Areq_{\text{LS}}$.

"polyfit" and "polyval" matlab functions are used to compute the polynomials' coefficients and to obtain budgets as a function of the required accuracy, respectively. Third order polynomial is used for the general applicability sensors, and second order one for the low orbit sensors.

Table 1 Characteristics of general applicability sensors

Accuracy [deg]	**Mass [kg]**	**Power Consumption [W]**	**Data-rate [kb/s]**
5	0.188	0.01	2
5	0.269	0.01	2
0.1	0.035	0.728	4
0.18	0.23	0.1	6
0.02	0.12	0.25	9
0.05	0.4	1.7	8
0.05	0.12	0.5	8
22/3600	1.42	3	16
90/3600	0.375	2	15
3/3600	3.16	11	30
18/3600	1.1	2.5	17

15/3600	1.8	2.8	18

Table 2 Characteristics of low orbit sensors

Accuracy [deg]	**Mass [kg]**	**Power Consumption [W]**	**Data-rate [kb/s]**
0.2	0.5	1.5	3
0.2	1.1	0.6	3
0.1	0.65	0.7	4
5	0.295	0.4	2

The total sensors' mass output M_{sens} is calculated as

$$M_{sens} = M_{Gsens} N_{Gsens} + M_{LOsens} N_{LOsens} \quad [kg] \tag{26}$$

where N_{Gsens} and N_{LOsens} are the number of general applicability sensors and the number of low orbit sensors, respectively. The total sensors' power consumption P_{sens} is calculated as follows:

$$P_{sens} = P_{Gsens} N_{Gsens} + P_{LOsens} N_{LOsens} \quad [W] \tag{27}$$

and the total sensors' data output D_{sens} is calculated as

$$D_{sens} = D_{Gsens} N_{Gsens} + D_{LOsens} N_{LOsens} \quad [kb/s] \tag{28}$$

At the moment the databases and polynomial regression processes are part of the routine, but when the routine will be used for automatic design, the regression process will be performed once for all and polynomials' coefficients will be directly hard-coded into the routine, in order to save computational time. In this case the absolute mass component Φ is equal to M_{sens} and the normalized component is 0.

5.4.1.1. Inertial Measurement Unit:

```
[Mimu,Pimu,Dimu] = IMU(DRreq,Ng,ALreq,Na);
```

The "IMU" routine models the mass, power consumption and output data-rate mass of the inertial measurement hardware, composed by gyroscopes and accelerometers, based on the accuracy of the components. Also in this case, a polynomial regression is performed on data shown in Table 3, to obtain the mass M_{Gyro}, the power consumption P_{Gyro}, and the output data-rate D_{Gyro} of gyro as a function of the required drift rate DR_{req}.

A further polynomial regression is performed on data shown in Table 4, to obtain the mass $\mathrm{M_{acc}}$, the power consumption $\mathrm{P_{acc}}$, and the output data-rate D_{acc} of low orbit sensors as a function of the required accuracy (linearity) A_{req}.

“polyfit” and “polyval” matlab functions are used, as well, to compute the polynomials’ coefficients and to obtain budgets as a function of the required accuracies, respectively. Second order polynomials are used both for gyros and accelerometers.

Table 3 Characteristics of gyros

Drift rate [deg/h]	**Mass [kg]**	**Power Consumption [W]**	**Data-rate [kb/s]**
0.009	2	12.5	9
0.009	2.8	12.5	8
0.002	2.6	13	10
0.009	6.3	13	7
1	1	10	4

Table 4 Characteristics of low orbit sensors

Linearity [g/g^2]	**Mass [kg]**	**Power Consumption [W]**	**Data-rate [kb/s]**
2.00E-06	0.3	5	2.3
3.00E-06	0.2	4	1.8
5.00E-06	0.16	2	1.3
5.00E-05	0.1	2	1

The total mass of the IMU, M_{imu} is calculated as

$$M_{imu} = M_{Gyro} N_g + M_{acc} N_a \quad [kg] \tag{29}$$

where N_g and N_a are the number of gyros and the number of accelerometers, respectively.

The total power consumption P_{imu} is

$$P_{imu} = P_{Gyro} N_g + P_{acc} N_a \quad [W] \tag{30}$$

and the total data output D_{sens} is obtained as

$$D_{sens} = D_{Gsens} N_{Gsens} + D_{LOsens} N_{LOsens} \quad [kb/s] \tag{31}$$

Also in this case the databases and polynomial regression processes are currently part of the routine, and will be substituted by direct hard-coding the polynomials’ coefficients.

5.5. Using LISA PathFinder (LPF) as test case for the SpaceART ADS model

The LPF Attitude and Orbit Control Subsystem (AOCS) uses as sensors two Autonomous Star Trackers, two Digital Sun Sensors for Sun acquisition and in safe mode and two Fibre-Optic Gyroscopes Units, required for certain phases where the optical sensors are inoperable due to high rates or eclipse. They are also needed to provide high bandwidth data during the main engine burn.

5.5.1. SpaceART ADS model LPF implementation

SpaceART is using the LPF as a test case in order to validate its ADS system model. Taking into account the mission facts and figures, each input can become a design, uncertain or fixed/environmental parameter depending on their definition respectively:

Design parameters are defined as input parameters which can be decided by the designer(s), after taking into account several mission factors.

Fixed parameters (including environmental parameters) are defined as input parameters which are already known or given, therefore are considered constants.

Uncertain parameters are defined as input parameters which cannot be deterministically defined. Such parameters can either be design or fixed ones which cannot be defined with absolute certainty.

Taking the above definitions into account, the SpaceART ADS model inputs are defined as follows:

5.5.2. Sensors LPF implementation:

The Sensors inputs consist of:

Accreq_gs - Required accuracy of sun or star sensors (deg) → $^{\text{Fixed}}$ parameter

Accreq_las - Required accuracy of horizon sensor or magnetometers (deg) → $^{\text{Fixed}}$ parameter

Ngs - Number of sun or star sensors → $^{\text{Fixed}}$ parameter

Nlas - Number of horizon sensors or magnetometers → $^{\text{Fixed}}$ parameter

5.5.3. Inertial Measurement Unit (IMU) LPF implementation:

The IMU inputs consist of:

DRreq - Required drift rate for gyros (deg/h) → $^{\text{Fixed}}$ parameter

Ng - Number of gyroscopes → $^{\text{Fixed}}$ parameter

ALreq - Required acceleration accuracy for accelerometers → $^{\text{Fixed}}$ parameter

Na - Number of accelerometers → $^{\text{Fixed}}$ parameter

5.6. References:

Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz, Microcosm Press & Kluwer Academic Publishers

"Space System Design", lecture slides by Dr. Massimiliano Vasile

6. Attitude Control System (ACS): UML Model

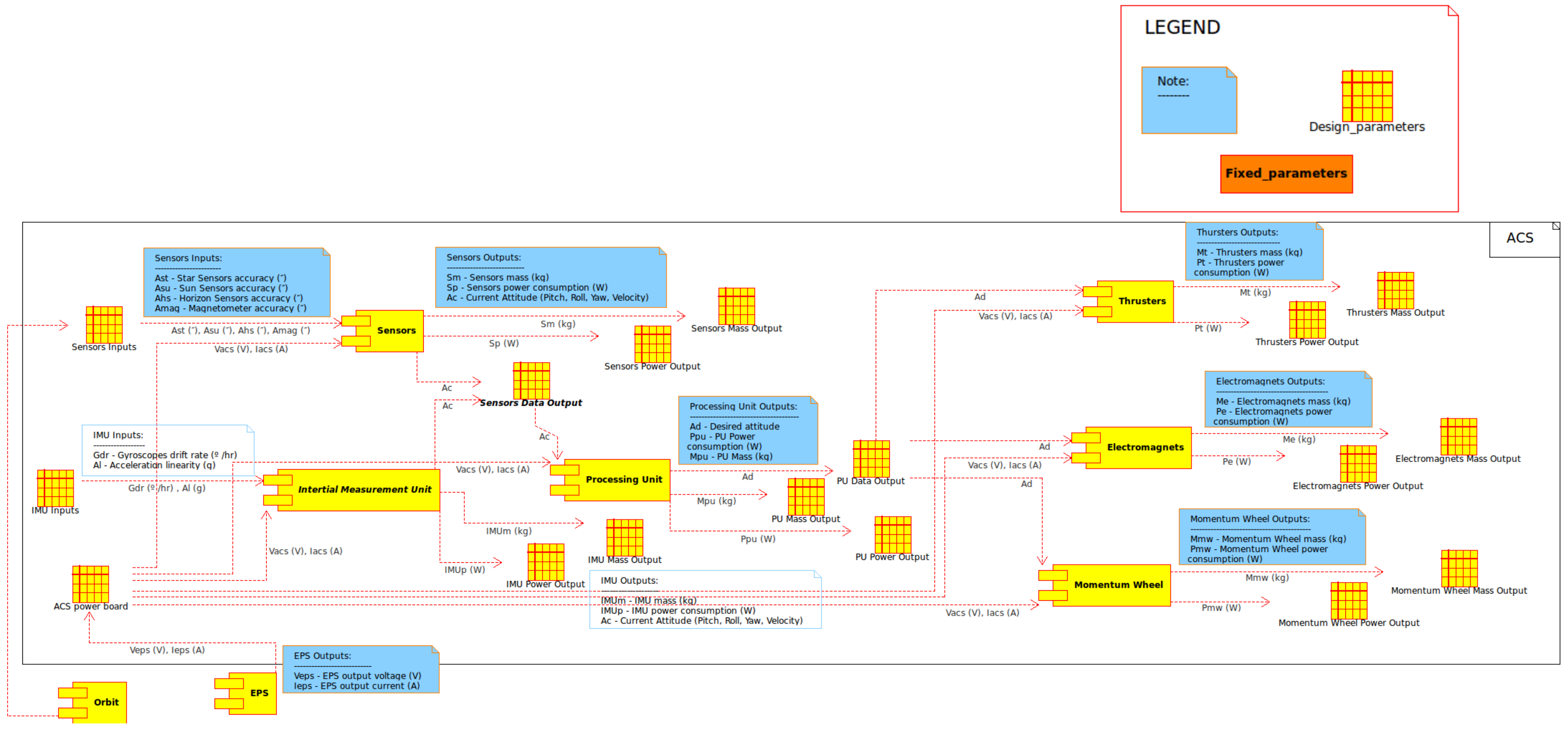

LEGEND
Note:
Design_parameters
Fixed_parameters
ACS
Sensors Inputs:
Ast - Star Sensors accuracy (")
Asu - Sun Sensors accuracy (")
Ahs - Horizon Sensors accuracy (")
Amag - Magnetometer accuracy (")
Sensors Outputs:
Sm - Sensors mass (kg)
Sp - Sensors power consumption (W)
Ac - Current Attitude (Pitch, Roll, Yaw, Velocity)
Sensors Inputs
Sensors
Ast ("), Asu ("), Ahs ("), Amag (")
Vacs (V), Iacs (A)
Sm (kg)
Sp (W)
Sensors Mass Output
Sensors Power Output
Ac
Sensors Data Output
IMU Inputs:
Gdr - Gyroscopes drift rate (º /hr)
Al - Acceleration linearity (g)
IMU Inputs
Gdr (º/hr) , Al (g)
Intertial Measurement Unit
IMUm (kg)
IMUp (W)
IMU Mass Output
IMU Power Output
IMU Outputs:
IMUm - IMU mass (kg)
IMUp - IMU power consumption (W)
Ac - Current Attitude (Pitch, Roll, Yaw, Velocity)
Processing Unit
Processing Unit Outputs:
Ad - Desired attitude
Ppu - PU Power consumption (W)
Mpu - PU Mass (kg)
Ad
Mpu (kg)
Ppu (W)
PU Data Output
PU Mass Output
PU Power Output
Thursters Outputs:
Mt - Thrusters mass (kg)
Pt - Thrusters power consumption (W)
Thrusters
Mt (kg)
Pt (W)
Thrusters Mass Output
Thrusters Power Output
Electromagnets Outputs:
Me - Electromagnets mass (kg)
Pe - Electromagnets power consumption (W)
Electromagnets
Me (kg)
Pe (W)
Electromagnets Mass Output
Electromagnets Power Output
Momentum Wheel Outputs:
Mmw - Momentum Wheel mass (kg)
Pmw - Momentum Wheel power consumption (W)
Momentum Wheel
Mmw (kg)
Pmw (W)
Momentum Wheel Mass Output
Momentum Wheel Power Output
ACS power board
Veps (V), Ieps (A)
EPS Outputs:
Veps - EPS output voltage (V)
Ieps - EPS output current (A)
EPS
Orbit

Figure 6. Attitude Control System Model UML Diagram

6.1. SpaceART ACS model description

The ACS model implements three types of operations: disturbance compensation, slewing manoeuvre, wheel desaturation . Three actuators are sized to be used in all three situations: momentum wheels, thrusters, magneto torquers. Magneto torquers are not considered if the spacecraft is far from the central body or the central body has not magnetic field.

The disturbance compensation model assumes that there are three types of disturbance acting on the spacecraft: solar pressure disturbance, magnetic filed disturbance, aerodynamic disturbance.
The wheels are assumed to store the momentum induced by the disturbance and therefore need to be desaturated using thrusters.
Magneto torquers are sized assuming a direct compensation of the disturbance. The thrusters are sized to desaturate the wheels and to be able to compensate for the disturbance. Currently the model implements only electric propulsion for the thrusters. Future versions will implement also other types of propulsion.
The disturbances are introduced as part of the environmental parameters and calculated outside the ACS model.

The integral of the momentum stored in the wheels:

$$H_p = \frac{T_{sp} + T_m + T_a}{4\theta_{acc}} \tag{32}$$

Angular momentum required for a slewing manoeuvre of an angle θ in time Δt around an axis with moment of inertia *I*:

$$h_s = \frac{1}{2}\frac{I\theta}{\Delta t} \tag{33}$$

The power required to actuate the wheels is estimated through a specific power coefficient α_{wheels} :

$$P_{wheels} = \alpha_{wheels} H_p \tag{34}$$

and the mass is estimated using a specific mass coefficient μ_{wheels}:

$$M_{wheels} = \mu_{wheels} P_{wheels} \tag{35}$$

The specific power required to actuate the wheels is estimated through a specific power coefficient α_{wheels} :

$$p_{wheels} = \alpha_{wheels} h_g \tag{36}$$

and the mass is estimated using a specific mass coefficient μ_{wheels}:

$$m_{wheels} = \mu_{wheels} p_{wheels} \tag{37}$$

The wheels can operate with magnetotorquers or with thrusters. In the former case the dipole required to dump a stored angular momentum H_{p} in a magnetic field *B* is:

$$D = \frac{H_p}{B} \tag{38}$$

with a required power:

$$p_{torque} = \frac{DV_{bus}}{\alpha I} \tag{39}$$

If the magneto torquers are used to compensate for a disturbance then the dipole is:

$$D = \frac{T_{sp} + T_m + T_a}{B} \tag{40}$$

$$p_{torque} = p_{torque} + \frac{DV_{bus}}{\alpha I} \tag{41}$$

$$m_{torque} = \mu_{eng} p_{torque} \tag{42}$$

the total mass and power of the magneto torquers are:

$$\Psi_{acs} = m_{torque} + m_{wheels} \quad (43)$$

$$p_{acs} = p_{wheels} + p_{torque} \quad (44)$$

$$P_{acs} = 0 \quad (45)$$

$$\Phi_{acs} = 0 \quad (46)$$

If thrusters are used to desaturate and compensate then the thrust level to desaturate in a time Δt is:

$$F_{dump} = \frac{H_p}{L\Delta t} \quad (47)$$

with propellant mass:

$$M_p = \frac{F_{dump}\Delta t}{I_{sp} g_0} \quad (48)$$

The thrust required to compensate for disturances is:

$$T = \frac{T_{sp} + T_m + T_a}{L} \quad (49)$$

with propellant mass:

$$M_p = M_p + \frac{TP_{orb}}{I_{sp} g_0} \quad (50)$$

If electric thrusters are used, then their power consumption and mass can be computed as follows :

$$P_{eps} = \frac{F_{dump} I_{sp} g_0}{2\eta} \quad (51)$$

$$M_{eng} = \mu_{eng} P_{eps} \quad (52)$$

$$M_{eps} = \frac{P_{eps}}{\alpha} \quad (53)$$

$$M_{\text{tanks}} = \varepsilon_{harn} M_p \quad (54)$$

$$\Phi_{acs} = M_{wheels} + M_{eng} + M_{\text{tanks}} + M_p \quad (55)$$

$$P_{acs} = P_{wheels} + P_{eps} \quad (56)$$

$$p_{acs} = 0 \quad (57)$$

$$\Psi_{acs} = 0 \quad (58)$$

6.2. SpaceART ACS function

```
%
%   [Psiacs,Phiacs,Pacs,pacs]=acs_norm(x,ep,flag)
%   Computes the mass fraction and power of the attitude control system
%
%
%
%   INPUT
%
%       Isp=x(1)   - specific impulse of the thrusters (s)
%       time=x(2)   -  required time to perform a manoeuvre (s)
%       theta_accuracy=x(3) - required pointing accuracy (rad)
%       theta=x(4) - required slew angle (rad)
%       mu_eng=x(5) - specific mass of the thruster/torquers (kg/W)
%       alpha=x(6) - specific power of the thrusters/torquers (W/kg)
%       eta=x(7) - efficiency of the thrusters/torquers
%       harn_fraction=x(8) - mass ratio of tanks and harness
%       alpha_wheels=x(9)- specific power of wheels (W/(N*m*s)
%       mu_wheels=x(10) - specific mass of wheels (kg/W)

%       ep(1)=tg - gravity gradient disturbance (m^2/s^2)
%       ep(2)=Tsp- solar radiation pressure disturbance (Nm)
%       ep(3)=Tm - magnetic field disturbance (Nm)
%       ep(4)=Ta - aerodynamic disturbance (Nm)
%       ep(5)=B   - magnetic field (Tesla)
%       ep(6)=theta - slew manoeuvre angle (rad)
%       ep(7)=Porb  - orbit period  (s)
%       ep(8)=theta_accuracy -  required angular accuracy (rad)
%       ep(9)=Vbus           - bus voltage (V)
%       ep(10)=time          - time allowed to perform the slewing manouvre
%       (s)
%       ep(11)=I             - specific moment of inertia (m^2)
%       flag                 - switch between magneto torquers and wheels
%
%
%    OUTPUT
%            Psiacs    - Psi mass function (normalised mass component) for the
ACS
%            Phiacs    - Phi mass function (absolute mass component) for the ACS
%            Pacs      - Power requirement for the ACS
%            pacs      - Specific power requirement for the ACS

%% REFERENCES
%
%       Based on: The ACS UML and UML description
% [1] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R.
%     Wertz,
%     Microcosm Press & Kluwer Academic Publishers
```

6.3. SpaceART ACS model applied to the LPF AOCS case

The ACS model applied to LPF is used to size the thrusters to compensate for disturbances. Given the current configuration of LPF, the sizing scenario is during the operational like around the Lagrangian point. The torques and disturbances are computed outside the satellite subsystem area in the orbit block and passed to the ACS subsystem block.

k.

7. Command & Data Handling

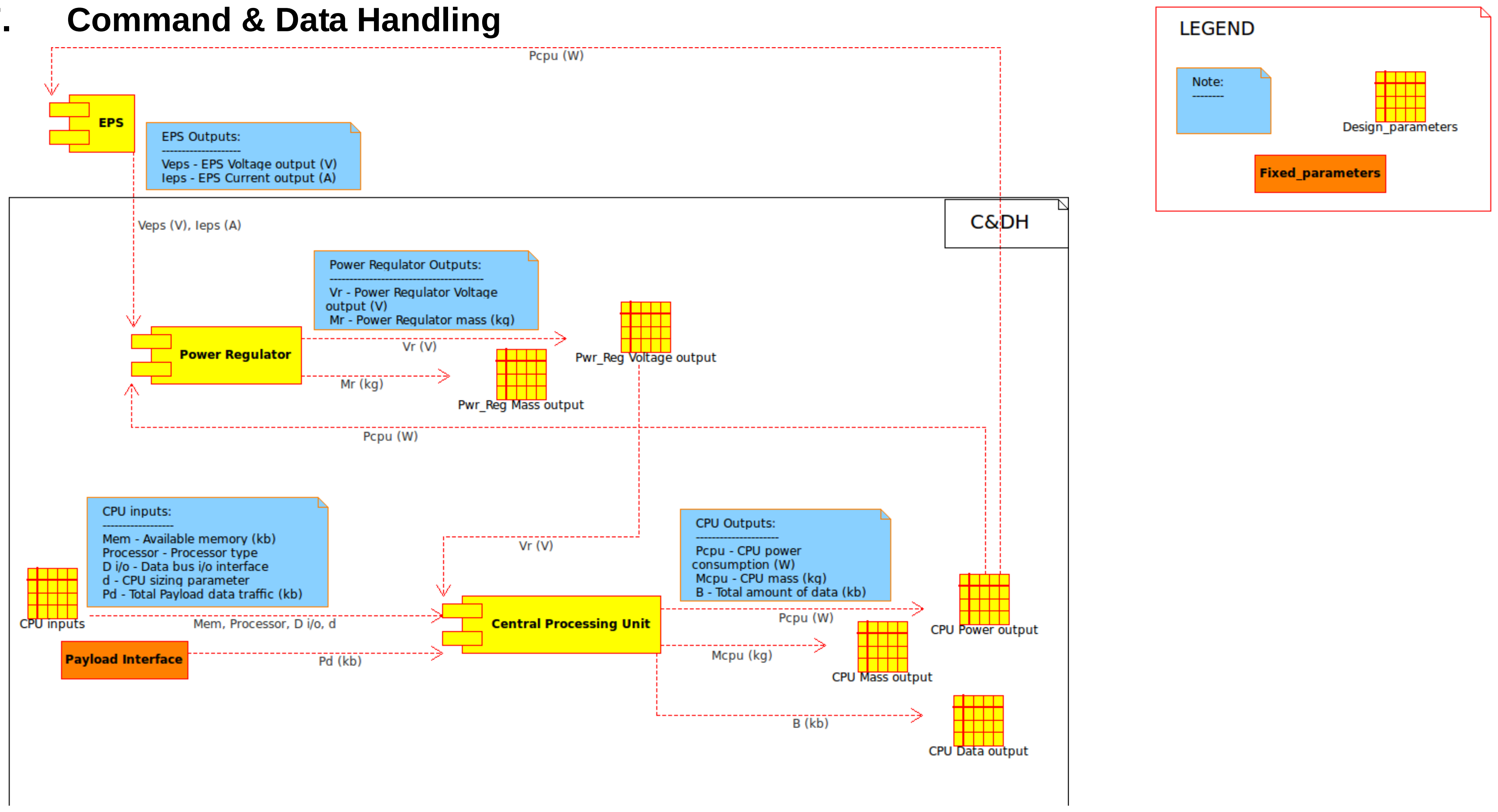


Figure 7. CD&DH Model UML Diagram

As seen above, the C&DH model consists of 6 blocks, each one representing a unit of the Command and Data Handling system. There is a total of 6 input parameters for the C&DH.

7.1. Power Regulator (PR):

The C&DH power regulator downgrades the spacecraft bus voltage to a compatible level for all the C&DH electronics.
The Power Regulator inputs consist of

Veps - EPS system voltage output in Volts (V) (EPS output)
Ieps - EPS system current output in Amperes (A) (EPS output) → Fixed/Design parameter (see Chapter 2, Introduction)
Pcpu – the CPU power consumption in Watts (W) (CPU output)

Using the provided inputs, the corresponding function uses three possible methods for calculating the corresponding output(s) depending on the available data. If enough data is available, the model uses analytical formulas for computing each output. Alternatively, regression/interpolation curves are used. If none of the aforementioned methods is available, discrete data is used for computing each output parameter.

The output parameters are:

Vr - Power Regulator voltage output in Volts (V)
Mr - Power Regulator mass output in kilograms (kg)

7.2. Central Processing Unit (CPU):

The C&DH CPU processes the commands and handles the spacecraft data traffic, acting as the maestro behind all spacecraft processes.

The CPU inputs consist of

Mem - the amount of available memory of the C&DH in kilobytes (kg) → Design parameter

Processor - the processor type → Design parameter

D i/o – Data bus i/o interface → Design parameter

Vr - Power Regulator voltage output in Volts (V) (PR output) → Design parameter

Pd - Total payload data traffic measured in kilobytes (kb)

Using the provided inputs, the corresponding function uses three possible methods for calculating the corresponding output(s) depending on the available data. If enough data is available, the model uses analytical formulas for computing each output. Alternatively, regression/interpolation curves are used. If none of the aforementioned methods is available, discrete data is used for computing each output parameter.
The output parameters are:

Pcpu – the CPU power consumption in Watts (W)

Mcpu – the CPU mass in kilograms (kg)

B – the total data traffic handled by the CPU in kilobytes (kb)

The C&DH physical size computation is one more output the SpaceART considers adding to the software model. This will add up to the overall complexity of the model, which will also reflect on the development

timeline. Nevertheless, it is considered an important addition which will lead to the C&DH model being a complete C&DH design tool.

7.3. SpaceART C&DH model function

```
%%            Command and Data Handling (C&DH) model
%
%
%       ooooooooooooooooooooooooooooooooooooo
%       8                                .d88
%       8  oooooooooooooooooooooooooooood8888
%       8  8888888888888888888888888P"   8888    oooooooooooooooo
%       8  8888888888888888888888P"      8888    8              8
%       8  8888888888888888888P"         8888    8             d8
%       8  8888888888888888P"            8888    8            d88
%       8  8888888888888P"               8888    8           d888
%       8  8888888888P"                  8888    8          d8888
%       8  8888888P"                     8888    8         d88888
%       8  8888P"                        8888    8        d888888
%       8  8888oooooooooooooooooooooocgmm8888    8       d8888888
%       8 .od88888888888888888888888888888888    8      d88888888
%       8888888888888888888888888888888888888    8     d888888888
%                                                8    d8888888888
%          ooooooooooooooooooooooooooooooo       8   d88888888888
%         d                       ...oood8b      8  d888888888888
%        d              ...oood888888888888b     8 d8888888888888
%       d     ...oood88888888888888888888888b    8d88888888888888
%      dood8888888888888888888888888888888888b
%
%
%     [Mcdh,Pcdh,Bcdh] = cdh (x,ep,flag)
%
%     DESCRIPTION
%     Command and Data Handling system model. It computes
%     the system's mass, power consumption and data output based on the
%     specified inputs found below.
%
%     INPUT
%    *x - All the design and uncertain parameters
%      x(1) = spec_mass - Conductor specific mass (kg/m)
%      x(2) = specific mass for data storage (kg/nbits)
%      x(3) = no_cpu number of cpu’s
%      x(4) = specific power for data storage (W/nbits)
%      x(5) = data volume fraction
%      x(6) = box mass fraction
%    *ep - All the environmental parameters (i.e inputs which are fixed, that
are neither design nor uncertain parameters)
%      ep(1) = datarate - Chosen bus datarate (kbits/sec)
%    *flag - Flag used to change between system types
%
%     OUTPUT
%      Mcdh - C&DH system's mass (kg)
%      Pcdh - C&DH system's power consumption (W)
%      Bcdh - C&DH system's data output (kb)
%
%
%     FUNCTION CALLS
%          none
%
%     CHANGE LOG
%
%  Created by: Eirini Komninou - 13/12/2010
%  Modified by:
%
%      REFERENCES
%       Based on: The C&DH UML
% [1] Space Vehicle Design, 2nd ed, Michael D. Griffin, James R. French, AIAA
education series, 2004
% [2] Elements of Spacecraft Design, Charles D. Brown, AIAA education series,
2002
% [3] MIL-STD 1553B,
% http://en.wikipedia.org/wiki/MIL-STD-1553#Physical_layer & MIL-HDBK-1553A
section40-42
% [4] IEEE-488, http://en.wikipedia.org/wiki/IEEE-488#Characteristics
% [5] RS-422, http://en.wikipedia.org/wiki/RS-422
% [6] RS-232, http://en.wikipedia.org/wiki/RS-232#Scope_of_the_standard &
http://www.arcelect.com/rs232.htm
% [7] IEEE-1394 (FireWire), http://en.wikipedia.org/wiki/IEEE_1394_interface
% [8] Serial Communication Cable - Tyco electronics datasheet
% [9] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R.
Wertz,
%     Microcosm Press & Kluwer Academic Publishers, 1999
%_______________________________________________________________________________
___________________________________________
```

7.4. SpaceART C&DH model description

7.4.1. Harness and CPU Sizing

Given the *max_datarate* = [20 1000 8000 10000 400000] in *kbit/s* and the corresponding *max_cable_length* = [15.2 60.9 20 1216 4.5] in *m*, for the data harness, one can compute the required length for a given datarate by interpolating the data and expressing the required harness length as a function of the datarate:

$$l_{harn} = f(datarate) \tag{59}$$

The corresponding mass is then computed as follows:

$$M_{data\_wire} = 1.2\mu_{cab} l_{harn} \tag{60}$$

The specific cable mass μ_{cab} refers to the conductor only therefore we add 20% extra mass for the wire jacket.

Table 5. Look-up table for mass and power of the CPU as a function of number of instructions

	Lower bound	Upper bound
$n_{\text{instruction}}$	200	300
M_{board}	0.3	0.5
P_{board}	5	10

The mass and power of each individual CPU is obtained from linear interpolation of the data in a look-up table (see Table 5):

$$\begin{aligned} M_{per\_cpu} &= \frac{M_{board}(2) - M_{board}(1)}{n_{instructions}(2) - n_{instructions}(1)}(n_{instructions} + n_{instructions}(1)) \\ P_{per\_cpu} &= \frac{P_{board}(2) - P_{board}(1)}{n_{instructions}(2) - n_{instructions}(1)}(n_{instructions} + n_{instructions}(1)) \end{aligned} \tag{61}$$

The total CPU mass and power is then the individual mass and power times the number of CPU's:

$$\begin{aligned} M_{cpu} &= M_{per\_cpu} n_{cpu} \\ P_{cpu} &= P_{per\_cpu} n_{cpu} \end{aligned} \tag{62}$$

The mass of the data storage unit is given by the data volume to be stored v_{data} times a data storage specific mass μ_{data}. Likewise the power associated to data storing is given by the data volume times the specific power π_{data}:

$$\begin{aligned} M_{data\_storage} &= \mu_{data} v_{data} \\ P_{data\_storage} &= \pi_{data} v_{data} \end{aligned} \tag{63}$$

In this implementation the data volume is a fraction ε_{data} of the input datarate, while the number of instructions are (1- ε_{data}) of the input datarate.
The mass of the box containing the data storage and CPU is a fraction ε_{box} of the mass of the CPU and data storage unit:

$$M_{box} = \varepsilon_{box}(M_{data\_storage} + M_{cpu}) \tag{64}$$

The mass of the power regulator is defined as:

$$M_{preg} = 0.02\left(P_{cpu} + P_{storage}\right) \tag{65}$$

The total mass of the C&DH system is given by:

$$M_{cdh} = M_{cpu} + M_{data\_wire} + M_{data\_storage} + M_{box} + M_{preg} \tag{66}$$

and the required power is:

$$P_{cdh} = P_{cpu} + P_{storage} \tag{67}$$

7.5. Using LPF as test case for SpaceART C&DH

As described above, LPF contains one computer on board, the OBC. It is the central control unit for all on board data handling activities including the AOCS, the DFACS, platform management and payload equipments.

7.5.1. LPF OBC

The OBC utilizes:

Two telecommand, telemetry and reconfiguration units, which will be represented by the CPU block of the SpaceART C&DH model, based on its technical specifications.

Four actuator and sensors interface modules, which will be represented by the CPU block of the SpaceART C&DH model, based on its technical specifications.

Two mass memory units, which will be represented by the CPU block of the SpaceART C&DH model, based on its technical specifications.

One interface bus MIL-1553B, which will be represented by the Payload interface fixed input parameter of the SpaceART C&DH model, based on its technical specifications.

All the interfaces with AOCS, Payload and TT&C will generate datarate that needs to be processed by the OBC. The datarate will enter as an input to the SpaceART C&DH model and will generate a power and mass output.

.

8. Structures and Mechanisms

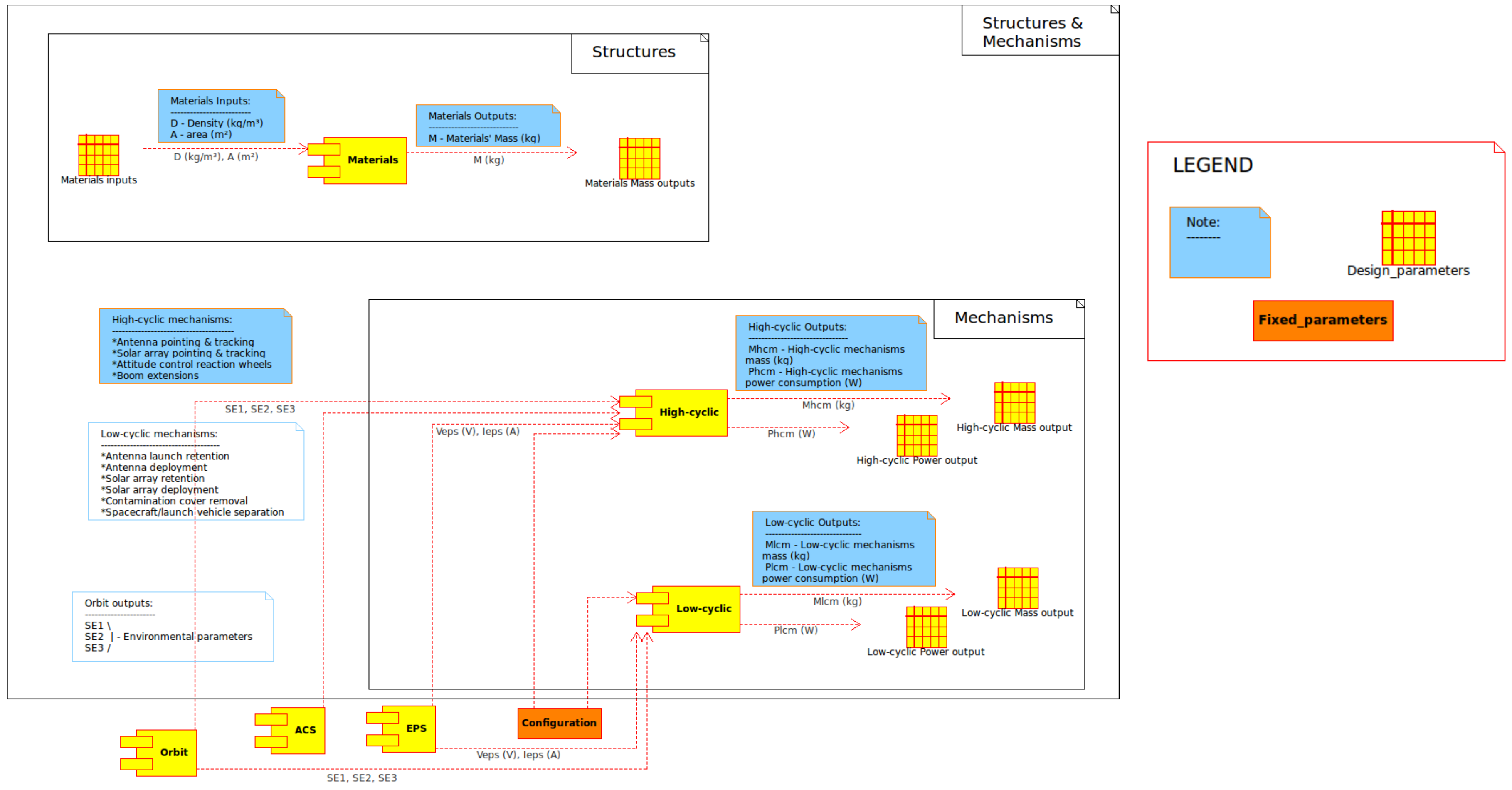


Figure 8. Structure and Mechanism UML Model

As seen above, the Structures & Mechanisms (S&M) model consists of two subsystems i.e Structures containing 1 block and Mechanisms containing 2 blocks, each one representing a unit of the S&M system. Overall there is a total of 4 input parameters for the S&M system.

Structures:

Materials:

The Materials block represents the hardware used to construct a spacecraft's structure.
The Materials inputs consist of

D - Density of the structural material measured in kilograms per meter cubed (kg/m^3) → Design parameter
A - the spacecraft's structure area measured in square meters (m^2) → Design parameter

Using the provided inputs, the corresponding function uses three possible methods for calculating the corresponding output(s) depending on the available data. If enough data is available, the model uses analytical formulas for computing each output. Alternatively, regression/interpolation curves are used. If none of the aforementioned methods is available, discrete data is used for computing each output parameter.

The output parameters are:

M - Materials' mass in kilograms (kg)

Mechanisms:

High-cyclic:

The High-cyclic block represents mechanisms such as the antenna(s) pointing and tracking, the solar array(s) pointing and tracking, the attitude control reaction wheels and boom extensions.

The High-cyclic inputs consist of

ACS provides the current spacecraft attitude to all high-cyclic mechanisms (ACS output)

SE1, SE2, SE3 - Environmental parameters specified by Orbit (Orbit output)

Configuration - Provides the current spacecraft configuration → Fixed parameter

Veps - EPS Voltage output in Volts (V) (EPS output)

Ieps - EPS Current output in Amperes (A) (EPS output)

Using the provided inputs, the corresponding function uses three possible methods for calculating the corresponding output(s) depending on the available data. If enough data is available, the model uses analytical formulas for computing each output. Alternatively, regression/interpolation curves are used. If none of the aforementioned methods is available, discrete data is used for computing each output parameter.

The output parameters are:

Mhcm – Hich-cyclic mechanisms' mass in kilograms (kg)

Phcm – High-cyclic mechanisms' power consumption in Watts (W)

Low-cyclic:

The Low-cyclic block represents mechanisms such as the antenna(s) launch retention, the antenna(s) deployment mechanism(s), the solar array(s) retention, the solar array(s) deployment mechanism(s), contamination covers removal mechanisms, spacecraft-launch vehicle separation mechanisms.

The Low-cyclic inputs consist of

SE1, SE2, SE3 - Environmental parameters specified by Orbit (Orbit output)

Configuration – Provides the current spacecraft configuration → Fixed parameter

Veps - EPS Voltage output in Volts (V) (EPS output)

Ieps - EPS Current output in Amperes (A) (EPS output)

Using the provided inputs, the corresponding function uses three possible methods for calculating the corresponding output(s) depending on the available data. If enough data is available, the model uses analytical formulas for computing each output. Alternatively, regression/interpolation curves are used. If none of the aforementioned methods is available, discrete data is used for computing each output parameter.

The output parameters are:

Mlcm - Low-cyclic mechanisms mass in kilograms (kg)

Plcm - Low-cyclic mechanisms power consumption in Watts (W)

The S&M system physical size computation is one more output the SpaceART considers adding to the software model. This will add up to the overall complexity of the model, which will also reflect on the development timeline. Nevertheless, it is considered an important addition which will lead to the S&M system model being a complete S&M system design tool.

Using LPF as test case for SpaceART S&M:

As described above, LPF's backbone is its octagonal shaped structure consisting of combined materials for maintaining high mechanical strength while keeping the mass to a minimum.

LPF Structure:

The Structure module utilizes:

Carbon fibre laminate skins bonded to aluminium honeycomb core consisting the satellite octagonal structure, which will be represented by the Materials block, based on the structure's materials technical specifications.

LPF Mechanisms:

The Mechanisms module utilizes:

A LPF/launch vehicle separation mechanism, which will be represented by the Low-cyclic block, based on the mechanism's technical specifications.

Since there are no corresponding modules on board LPF, the High-cyclic block can be omitted from the model, by setting its value to null.

Taking the aforementioned data into account, the SpaceART S&M software model will produce outputs that can size the LPF 's Structure and mechanisms mass and power consumption respectively, with high accuracy.

9. Power subsystem: UML diagram

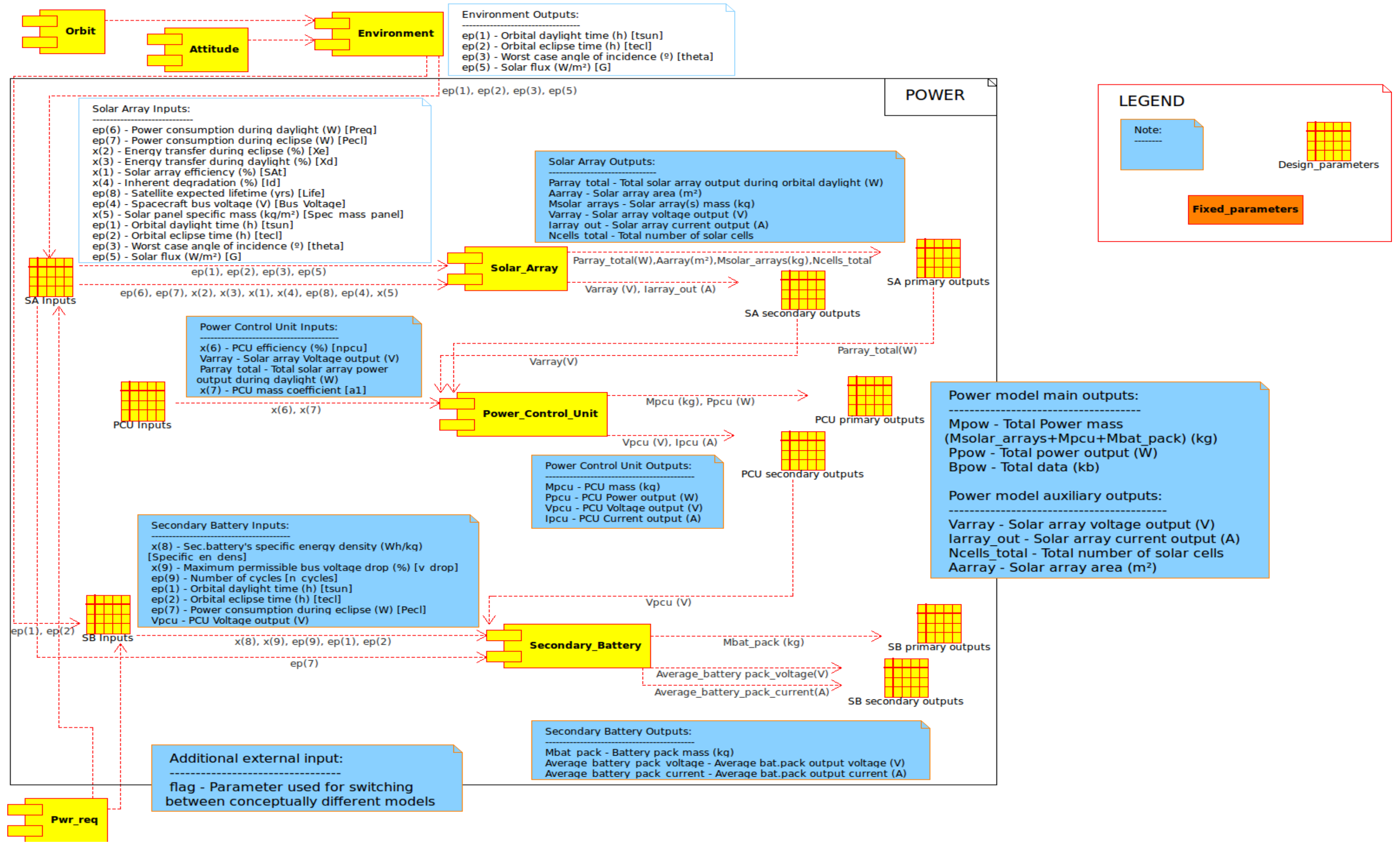

Orbit
Attitude
Environment
Environment Outputs:
ep(1) - Orbital daylight time (h) [tsun]
ep(2) - Orbital eclipse time (h) [tecl]
ep(3) - Worst case angle of incidence (º) [theta]
ep(5) - Solar flux (W/m²) [G]
ep(1), ep(2), ep(3), ep(5)
POWER
Solar Array Inputs:
ep(6) - Power consumption during daylight (W) [Preq]
ep(7) - Power consumption during eclipse (W) [Pecl]
x(2) - Energy transfer during eclipse (%) [Xe]
x(3) - Energy transfer during daylight (%) [Xd]
x(1) - Solar array efficiency (%) [SAt]
x(4) - Inherent degradation (%) [Id]
ep(8) - Satellite expected lifetime (yrs) [Life]
ep(4) - Spacecraft bus voltage (V) [Bus Voltage]
x(5) - Solar panel specific mass (kg/m²) [Spec mass panel]
ep(1) - Orbital daylight time (h) [tsun]
ep(2) - Orbital eclipse time (h) [tecl]
ep(3) - Worst case angle of incidence (º) [theta]
ep(5) - Solar flux (W/m²) [G]
SA Inputs
ep(1), ep(2), ep(3), ep(5)
ep(6), ep(7), x(2), x(3), x(1), x(4), ep(8), ep(4), x(5)
Solar_Array
Solar Array Outputs:
Parray total - Total solar array output during orbital daylight (W)
Aarray - Solar array area (m²)
Msolar arrays - Solar array(s) mass (kg)
Varray - Solar array voltage output (V)
Iarray out - Solar array current output (A)
Ncells total - Total number of solar cells
Parray_total(W),Aarray(m²),Msolar_arrays(kg),Ncells_total
Varray (V), Iarray_out (A)
SA primary outputs
SA secondary outputs
Parray_total(W)
Varray(V)
Power Control Unit Inputs:
x(6) - PCU efficiency (%) [npcu]
Varray - Solar array Voltage output (V)
Parray total - Total solar array power output during daylight (W)
x(7) - PCU mass coefficient [a1]
PCU Inputs
x(6), x(7)
Power_Control_Unit
Mpcu (kg), Ppcu (W)
Vpcu (V), Ipcu (A)
PCU primary outputs
PCU secondary outputs
Power Control Unit Outputs:
Mpcu - PCU mass (kg)
Ppcu - PCU Power output (W)
Vpcu - PCU Voltage output (V)
Ipcu - PCU Current output (A)
Power model main outputs:
Mpow - Total Power mass
(Msolar_arrays+Mpcu+Mbat_pack) (kg)
Ppow - Total power output (W)
Bpow - Total data (kb)
Power model auxiliary outputs:
Varray - Solar array voltage output (V)
Iarray_out - Solar array current output (A)
Ncells_total - Total number of solar cells
Aarray - Solar array area (m²)
Secondary Battery Inputs:
x(8) - Sec.battery's specific energy density (Wh/kg) [Specific en dens]
x(9) - Maximum permissible bus voltage drop (%) [v drop]
ep(9) - Number of cycles [n cycles]
ep(1) - Orbital daylight time (h) [tsun]
ep(2) - Orbital eclipse time (h) [tecl]
ep(7) - Power consumption during eclipse (W) [Pecl]
Vpcu - PCU Voltage output (V)
Vpcu (V)
SB Inputs
ep(1), ep(2)
x(8), x(9), ep(9), ep(1), ep(2)
ep(7)
Secondary_Battery
Mbat_pack (kg)
Average_battery pack_voltage(V)
Average_battery_pack_current(A)
SB primary outputs
SB secondary outputs
Secondary Battery Outputs:
Mbat pack - Battery pack mass (kg)
Average battery pack voltage - Average bat.pack output voltage (V)
Average battery pack current - Average bat.pack output current (A)
Additional external input:
flag - Parameter used for switching between conceptually different models
Pwr_req
LEGEND
Note:
Design_parameters
Fixed_parameters

Figure 9. Power Model UML Diagram

According to the Power system UML diagram the Power system model consists of 3 blocks, each one representing a separate physical unit. There is a total of 19 external input parameters for the Power system. The initial input is “flag” which specifies which model characteristics will be used.

The Power system receives inputs from external sources like the Environment block representing external environmental parameters that contribute to sizing parts of the EPS.

The model description in this section does not consider the integration with other subsystems but the Power subsystem alone. Therefore in this issue of the document it does not include the separation of the mass in absolute and normalized.

9.1. Solar Array (SA)

The Power system Solar Array unit generates electricity by making use of the photovoltaic phenomenon.

The Solar Array inputs consist of:

tsun – Orbital daylight time in hours (h) → Environmental/Orbital Parameter

tecl – Orbital eclipse time in hours (h) → Environmental/Orbital Parameter

theta – Worst case angle of incidence in degrees (º) → Environmental/Orbital Parameter

G – Solar flux in Watts per square meter (W/m^2) → Environmental/Orbital Parameter

Preq – Power consumption during daylight in Watts (W)

Pecl – Power consumption during eclipse in Watts (W)

Xe – Energy transfer during eclipse in percentage (%)

Xd – Energy transfer during daylight in percentage (%)

SAt – Solar array efficiency in percentage (%)

Id – Inherent degradation in percentage (%)

Life – Satellite expected lifetime in years (yrs)

Bus_Voltage – Spacecraft bus voltage in Volts (V)

Spec_mass_panel – Solar panel specific mass in kilograms per square meter (kg/m^2)

The output parameters are:

Parray_total – Total solar array output during orbital daylight in Watts (W)

Array – Solar array area in square meters (m^2)

Msolar_arrays – Solar array(s) mass in kilograms (kg)

Varray – Solar array voltage output in Volts (V)

Iarray_out – Solar array current output in Amperes (A)

Ncells_total - Total number of solar cells

9.2. *Power Control Unit (PCU)*

The Power Control Unit handles and distributes the power generated by the spacecraft's solar arrays.

The PCU inputs consist of:

npcu – PCU efficiency in percentage (%)

Varray – Solar array Voltage output (V) → SA output (internal parameter)

Parray_total – Total solar array power output during daylight (W) → SA output (internal parameter)

a1 – PCU mass coefficient

The output parameters are:

Mpcu – PCU mass in kilograms (kg)

Ppcu – PCU Power output in Watts (W)

Vpcu – PCU Voltage output in Volts (V)

Ipcu – PCU Current output in Amperes (A)

9.3. *Secondary Battery (SB)*

The Secondary Battery unit provides the spacecraft with power during eclipse time. The SB block includes the Power Regulator Unit (PRU). Depending on the Power system type (Sun regulated, Fully regulated, Peak Power Tracking), the PRU consists of a charge & a discharge regulator, or just a charge regulator depending on the type of the power system.

The SB inputs consist of:

Specific_en_dens – Secondary battery's specific energy density in Watthours per kilogram (Wh/kg)

v_drop – Maximum permissible bus voltage drop in percentage (%)

n_cycles – Number of cycles

tsun – Orbital daylight time in hours (h) → Environmental/Orbital Parameter

tecl – Orbital eclipse time in hours (h) → Environmental/Orbital Parameter

Pecl – Power consumption during eclipse in Watts (W)

Vpcu – PCU Voltage output in Volts (V)→ PCU output (internal parameter)

The <u>output</u> parameters are:

Mbat_pack - Battery pack mass in kilograms (kg)

Average_battery_pack_voltage - Average battery pack output voltage in Volts(V)

Average_battery_pack_current - Average battery pack output current in Amperes (A)

9.4. *Power system's external outputs*

The Power system's external outputs are:

Mpow – Total Power mass I.e the sum of Msolar_arrays+Mpcu+Mbat_pack (kg)

Ppow – Total power output I.e the sum of the power requirements of all the individual systems onboard the satellite (W)

Bpow – Total Power data output (kb)

Additionally the Power system provides an extra four auxiliary outputs:

Varray – Solar array Voltage output (V)

Iarray_out – Solar array current output in Amperes (A)

Ncells_total - Total number of solar cells

Array – Solar array area in square meters (m^2)

9.5. *SpaceART Power System model function*

```
%%          Power System model
%
%
%    [Mpow,Ppow,Bpow,Varray,Iarray_out,Ncells_total,Aarray] = power_sys(x,ep,flag)
%
%
%    DESCRIPTION
%    Electrical Power System (Power) model. It computes
%    the Power system mass, power output and data output based on the
%    specified inputs found below.
%
%
%    INPUT
%   *x - All the design and uncertain parameters
%      x(1) = SAt - solar cell efficiency (%)
%      x(2) = Xe - Energy transfer during eclipse (%)
%      x(3) = Xd - Energy transfer during daylight (%)
%      x(4) = Id - Inherent degradation (%)
%      x(5) = Spec_mass_panel array specific mass - (kg/m^2)
%      x(6) = npcu - PCU efficiency (%)
%      x(7) = a1 - PCU mass coefficient
%      x(8) = Specific_en_dens - Secondary batteries specific energy density (Wh/kg)
%      x(9) = v_drop - Maximum permissible bus voltage drop (%)
%   *ep - All the environmental parameters (i.e inputs which are fixed, that are neither design nor uncertain
%          parameters)
%      ep(1) = tsun - Orbital daylight time (h)
%      ep(2) = tecl - Orbital eclipse time (h)
%      ep(3) = theta - Worst case angle of incidence (º)
%      ep(4) = Bus_Voltage - Specified bus voltage (V)
%      ep(5) = G - Solar flux (W/m²)
%      ep(6) = Preq - Power consumption during daylight (W)
%      ep(7) = Pecl - Power consumption during eclipse (W)
%      ep(8) = Life - Expected satellite lifetime (yrs)
%      ep(9) = n_cycles - Number of charge/discharge battery cycles
%   *flag - Flag used to change between system types e.g from Power solar to Power nuclear
%
%
%    OUTPUT
%     Mpow - Power system's mass including solar array(s) mass (kg)
%     Ppow - Power system's power output (W)
%     Bpow - Power system's data output (kb)
%   Non sizing OUTPUTS
%     Aarray - Solar array area (m²)
%     Varray - Solar array voltage output (V)
%     Iarray_out - Solar array current output (A)
%     Ncells_total - Number of cells in total
%
%
%    FUNCTION CALLS
%        none
```

```
%
%
%     REFERENCES
%      Based on: The EPS UML and UML description
% [1] Spacecraft Power Systems, Mukund R. Patel, CRC Press
% [2] Spacecraft Systems Engineering 3rd ed. , P. Fortescue, J. Stark, G. Swinerd, Wiley
% [3] Spacecraft Power Technologies, Anthony K. Hyder, Ronald L. Wiley, G. Halpert,
%     Donna Jones Flood, S. Sabripour, World Scientific Publishing Company
% [4] Space Vehicle Design, Michael D. Griffin, James R. French, AIAA
% [5] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz,
%      Microcosm Press & Kluwer Academic Publishers
% [6] Maximum Fast Charging Rates in Ni-Cd Batteries, Viera Pérez, J. C., Gonzalez, M., Campo, J. C., &
%      Ferrero, F. J., Space Power, Proceedings of the Sixth European Conference held 6-10 May, 2002 in
%      Porto, Portugal. Edited by A. Wilson. European Space Agency, ESA SP-502, 2002, p.565
% [7] http://teva2.com/NiH2.html
% [8] Guidelines on Lithium-ion Battery Use in Space Applications, Barbara McKissock, Patricia Loyselle,
%      and Elisa Vogel, RP-08-75, NASA
% [9] Recent developments in evaporated CdTe solar cells, G. Khrypunov,A. Romeo, F. Kurdesau, D.L.
%     Batzner, H. Zogg, A.N. Tiwari, Solar Energy Materials & Solar Cells 90 (2006) 664–677


%% Internal functions called
[Parray_total,Aarray,Msolar_arrays,Varray,Iarray_out,Ncells_total] =
Solar_arrays(Preq,Pecl,tsun,tecl,Xe,Xd,G,SAt,Id,theta,Life,Bus_Voltage,Spec_mass_panel)

[Mpcu,Ppcu,Ipcu,Vpcu] = PCU (npcu,Varray,Parray_total,a1)

[Mbat_pack,Average_battery_pack_voltage,Average_battery_pack_current] = Sec_batteries
(Specific_en_dens,v_drop,Vpcu,Pecl,tecl,tsun,n_cycles)


%% Total Mass, Power, Data
Mpow = Msolar_arrays + Mpcu + Mbat_pack
Ppow = Ppcu
Bpow = 0
```

As seen above, the power_sys.m is divided into individual functions, each one representing a logical block of the UML model I.e a hardware unit of the actual physical system.

9.6. SpaceART Power System model description

9.6.1. Solar Arrays

```
[Parray_total,Aarray,Msolar_arrays,Varray,Iarray_out,Ncells_total] =
Solar_arrays(Preq,Pecl,tsun,tecl,Xe,Xd,G,SAt,Id,theta,Life,Bus_Voltage,Spec_mass_panel)
```

The “Solar_ arrays” function models the solar array physical and electrical characteristics based on a number of inputs like the total required power on-board, the satellite bus voltage and so on. Using analytical mathematical expressions, the function outputs physical and electrical characteristics of the solar array(s). Initially, the solar array power output per orbit requirement P_{sa} is calculated as follows

$$P_{sa} = \frac{\left(\frac{P_e T_e}{X_e}\right) + \left(\frac{P_d T_d}{X_d}\right)}{T_d} \quad [W] \tag{68}$$

where P_e – Power consumption during eclipse (W), T_e – Orbital eclipse time (h), X_e – *E*nergy transfer during eclipse, P_d – Power consumption during daylight (W), T_d – Orbital daylight time (h), X_d – *E*nergy transfer during daylight.

The BOL (Beginning of Life) power output per unit area P_{BOL} is then calculated as follows

$$P_o = SAt \cdot G \quad [W] \tag{69}$$

$$P_{BOL} = P_o I_d \cos\theta \quad [W] \tag{70}$$

where P_o – Ideal power output per unit area, SAt – *S*olar cell efficiency, G – *S*olar flux (W/m²),

I_d – *I*nherent degradation, θ – Worst case angle of incidence (rad).

In order for the model to calculate the EOL (End of Life) power output per unit area, a solar array degradation over satellite lifetime factor L_d is calculated as follows

$$L_d = (1 - D)^{Life} \tag{71}$$

where D – *A*rray degradation per year, $Life$ – *E*xpected satellite lifetime (yrs).

Once the satellite lifetime factor L_d is computed, the power output during EOL P_{EOL} can be calculated based on the power output per unit area P_{BOL} as follows

$$P_{EOL} = P_{BOL} \cdot L_d \quad [W] \tag{72}$$

The calculations described above are essential for calculating the solar array area A_{sa} as follows

$$A_{sa} = \frac{P_{sa}}{P_{EOL}} \quad \left[m^2\right] \tag{73}$$

where P_{sa} – Solar array power output per orbit requirement (W) and P_{EOL} – Power output during EOL (W) deriving from Eq. (70) and Eq. (72) respectively.

The Solar Array mass M_{sa} is then calculated based on the solar array area A_{sa} as follows

$$M_{sa} = A_{sa} SM \quad [kg] \tag{74}$$

where A_{sa} – Solar array area (m²) deriving from Eq. (73), SM – Specific mass of panel (kg/m²).

Finally the function calculates the number of cells connected in series forming a string, the number of strings connected in parallel forming an array and the total number of cells based on the bus voltage and power requirements inputs.

Analytically: The designer inputs the selected bus voltage level V_{bus} as well as a selected solar cell efficiency *Sat*. Taking into account that the solar array output voltage level should be higher than the selected bus voltage level, the solar array voltage V_{sa} is calculated as follows

$$V_{sa} = V_{bus} f_{Vbus} \tag{75}$$

where V_{bus} – Satellite bus voltage level (V), f_{Vbus} – 1.2 . As seen in Eq. (75) the bus voltage level is raised by 20% in order to achieve a satisfactory solar array voltage level, since the bus losses (voltage drop) as well as the secondary battery's nominal charge voltage need to be taken into account. The *SAt* input defines the type of solar cell that will be used including its intrinsic characteristics, utilizing the data included in Table 1.

Table 6: Solar cell intrinsic characteristics

	CdTe	p c-Si[2]	u c-Si[3]	3j[4] GaAs	Concentrator 3j GaAs	Multijunction cells
V_{mp} (V)	0.74	0.53	0.43	2.29	2.68	2.26
J_{mp} (A)	0.028	8.08	0.774	0.525	6.76	1.75
η_{cell} (%)	16.5	20.3	24.7	29.9	38.5	40.7
D_{cell} (%)	1	3.75	3.75	5	5	5

where V_{mp} – solar cell voltage output at maximum power point (V), J_{mp} – solar cell current output at maximum power point (A), η_{cell} – solar cell efficiency, D_{cell} – solar cell degradation per year. Using Piecewise cubic Hermite interpolation which seems to produce the most accurate results, the function derives the maximum power point Voltage V_{mp} and maximum power point Current J_{mp} based on the choice of solar array efficiency *SAt* assuming that the Power system is a PPT (Peak Power Tracking) one.

The V_{mp} and J_{mp} values are then used in order to derive the number of cells in series required per string $N_{cells/string}$ as follows

$$N_{cells/string} = \frac{V_{sa}}{V_{mp}} \tag{76}$$

where V_{mp} – Maximum power point Voltage (V), V_{sa} – Solar array voltage (V). Then the number of strings connected in parallel per circuit $N_{strings}$ -assuming that each array is considered an individual circuit- is calculated as follows

$$I_{sa} = \frac{P_{sa}}{V_{sa}} \tag{77}$$

$$N_{strings} = \frac{I_{sa}}{J_{mp}} \tag{78}$$

2 Polycrystalline Silicon
3 Monocrystalline Silicon
4 Triple junction

where I_{sa} – the solar array current output (A), P_{sa} – Solar array power output per orbit requirement (W) deriving from Eq. (68), V_{sa} – Solar array voltage (V) deriving from Eq. (75). Finally, the number of cells required in total per solar array is calculated as follows

$$N_{total} = N_{cells/strings} N_{strings} \qquad (79)$$

where $N_{cells/string}$ – Number of cells in series required per string deriving from Eq. (76),

$N_{strings}$ – Number of strings connected in parallel per circuit deriving from Eq. (78).

9.6.2. Power Control Unit

[Mpcu,Ppcu,Ipcu,Vpcu] = PCU (npcu,Varray,Parray_total,a1)

The "PCU" function models the power control unit physical and electrical characteristics based on a number of inputs like the PCU efficiency, the total solar array power output and so on. Using analytical mathematical expressions the function outputs the electrical and physical characteristics like the PCU power output P_{pcu} I.e the power that the PCU handles, the voltage V_{pcu} and current I_{pcu} output as well as the PCU mass M_{pcu}.

Initially the "PCU" function calculates the PCU Voltage output V_{pcu} based on a 5% maximum voltage drop that can occur between the Solar array circuit and the PCU due to losses from resistive loads like wiring as follows

$$V_{pcu} = V_{sa} f_{Vsa} \quad [V] \qquad (80)$$

where V_{sa} – Solar array voltage (V) deriving from Eq. (75), f_{Vsa}. Space grade PCUs have an efficiency of at least 95% depending on the technology and materials used according to bibliography. Therefore the PCU power output (P_{pcu}) is calculated as follows

$$P_{pcu} = P_{sa} \eta_{pcu} \quad [W] \qquad (81)$$

where P_{sa} – Solar array power output per orbit requirement (W) deriving from Eq. (1), η_{pcu} – PCU efficiency. The current output I_{pcu} is then calculated as follows

$$I_{pcu} = \frac{P_{pcu}}{V_{pcu}} \quad [A] \qquad (82)$$

where P_{pcu} – PCU power output (W) deriving from Eq. (81), V_{pcu} – PCU Voltage output (V) deriving from Eq. (80) Finally the PCU mass M_{pcu} is calculated as a fraction of the PCU power output:

$$M_{pcu} = aP_{pcu} \quad [kg] \qquad (83)$$

where a is a PCU mass coefficient.

9.6.2.1. Secondary battery

[Mbat_pack,Average_battery_pack_voltage,Average_battery_pack_current] = Sec_batteries (Specific_en_dens,v_drop,Vpcu,Pecl,tecl,tsun,n_cycles)

The “Sec_batteries” function models the battery pack, the unit supplying the satellite with power during any eclipse periods that will occur throughout an orbit.

Based on a satellite's eclipse and sunlight times, the power consumption during eclipse and sunlight and so on, the “Sec_batteries” function models the physical and electrical characteristics of the battery pack outputting the battery pack mass M_{bat_pack}, as well as the average battery pack voltage and current.

Initially the designer inputs the chosen specific energy density *Specific_en_dens* which defines the particular battery chemistry to be used including its intrinsic characteristics, utilizing the data included in Table 7.

Table 7: Secondary battery cells' intrinsic characteristics

	NiCd	NiH_2	NiMH	LiIon	LiPoly
E_d (Wh/kg)	60	75	80	150	200
V_{cell} (V)	1.25	1.25	1.25	3.6	3.5
η_{batt} (%)	85	86	87	95	99.8
q	145.8	176.3	TBD	TBD	TBD

where E_d is the available energy density (Wh/kg) , V_{cell} is the voltage per cell (V), η_{batt} is the battery energy efficiency, q is the y-intercept.

By interpolating the data in Table 2 the function derives the corresponding battery energy efficiency η_{batt} as well as the corresponding discharge voltage per battery cell $^{V}{}_{cell}$. Furthermore, using a simple linear relationship in logarithmic scale, the depth of discharge *DOD* is calculated based on the parameter q and the number of cycles N_{cycles}.

After deriving the value of V_{cell} based on the selection of a *Specific_en_dens* value, the model calculates the minimum permissible bus voltage V_{minp} as follows

$$V_{minp} = V_{pcu}\left(1 - v_{drop}\right) \quad [V] \tag{84}$$

where V_{pcu} – PCU Voltage output (V) deriving from Eq. (80), v_{drop} – Maximum allowable voltage drop.

Furthermore, the model sets the minimum battery cell voltage at the end of discharge at EOL $V_{cellmin}$ as follows

$$V_{minp} = V_{pcu}\left(1 - v_{drop}\right) \quad [V] \tag{85}$$

where V_{cell} - Voltage per cell (V), $f_{Vcell} = 0.8$.

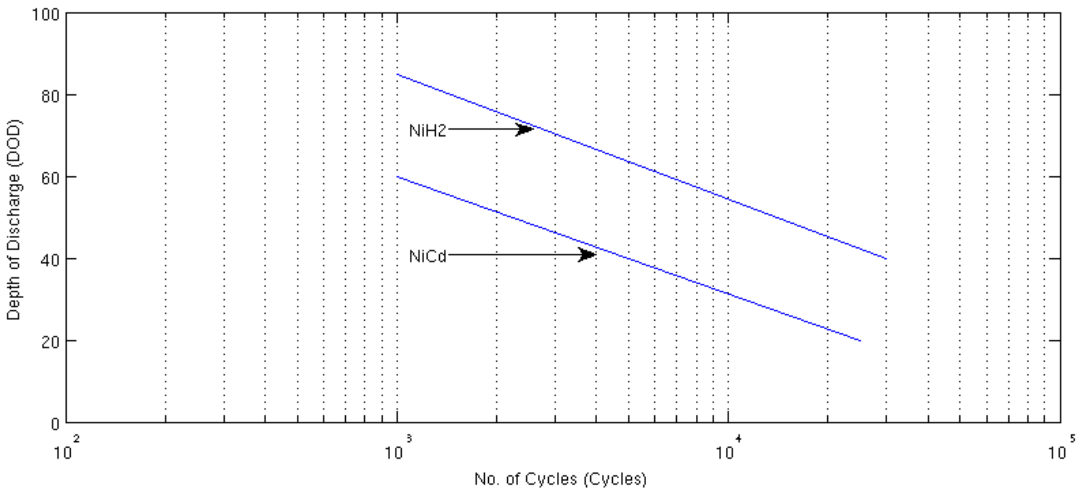


Figure 10: Number of Cycles versus Depth Of Discharge

Utilizing Eq. (86), the function calculates the number of cells connected in series $Cell_{series}$ that will set the battery cells' connection topology as follows

$$Cell_{series} = \frac{V_{minp}}{V_{cellmin}} \tag{86}$$

where V_{minp} – Minimum permissible bus voltage (V) deriving from Eq. (85), $V_{cellmin}$ – Minimum battery cell voltage at the end of discharge at EOL (V) deriving from Eq. (86). If the number of cells in series not an integer, the function rounds up the number to the nearest higher integer.

The function then calculates the depth of discharge *DOD* of the chosen type of battery as follows: Using the number of charge/discharge cycles input N_{cycles} and the y-intercept *q* values found in Table 2, a linear interpolation is performed between the lines of “lin-log”Figure 10.

The next step is to calculate the minimum battery pack energy $E_{minpack}$, providing the model with the minimum energy the batteries should have in order for the satellite to operate nominally even during the very end of the eclipse period. $E_{minpack}$ is calculated as follows

$$E_{minpack} = P_e T_e \quad [Wh] \tag{87}$$

where P_e – Power consumption during eclipse (W), T_e – Orbital eclipse time (h). Based on $E_{minpack}$ deriving from Eq. (87), the minimum battery capacity C_{min} can then be calculated as follows

$$C_{min} = \frac{E_{minpack}}{V_{cell} Cell_{series}} \quad [Ah] \tag{88}$$

where $E_{minpack}$ – Minimum battery pack energy (Wh) deriving from Eq. (87), V_{cell} - Voltage per cell (V), $Cell_{series}$ – Number of cells connected in series deriving from Eq.(86).

The required battery capacity is then calculated as follows

$$C = \frac{C_{min}}{DOD} \quad [Ah] \tag{89}$$

where C_{min} – minimum battery capacity (Ah) deriving from Eq. (88), *DOD* – Depth of discharge.

The required battery pack energy *E* is calculated utilizing $E_{minpack}$ as follows

$$E = \frac{E_{minpack}}{\eta_{batt}} \quad [Wh] \tag{90}$$

where $E_{minpack}$ – Minimum battery pack energy (Wh) deriving from Eq. (87), η_{batt} – battery energy efficiency taken from Table 7.

The average battery pack voltage V_{pack} is calculated as follows

$$\overline{Vpack} = V_{cell} Cells_{series} \quad [V] \tag{91}$$

where V_{cell} - Voltage per cell (V), $Cell_{series}$ - Number of cells connected in series deriving from Eq. (86). The average battery pack current is calculated after calculating the required power for charging the battery pack P_{charge} as follows

$$P_{charge} = \frac{E}{T_d} \quad [W] \tag{92}$$

where *E* – *R*equired battery pack energy (Wh) deriving from Eq. (90), T_d – Orbital daylight time (h)

Utilizing Eq. (92), the average battery pack current is calculated:

$$\overline{Ipack} = \frac{P_{charge}}{\overline{Vpack}} \quad [A] \tag{93}$$

where P_{charge} – Required power for charging the battery pack (W) deriving from Eq. (92), V_{pack} – average battery pack voltage (V) deriving from Eq. (91). Finally the mass of the battery cells M_{cells} is calculated as follows

$$M_{cells} = \frac{E}{Specific\_en\_dens} \quad [kg] \tag{94}$$

where *E* – *R*equired battery pack energy (Wh) deriving from Eq. (90), *Specific_en_dens* – Chosen specific energy density (Wh/kg). Finally the battery pack mass M_{pack} is considered equal to the mass of the battery cells M_{cells}:

$$M_{pack} = M_{cells} \quad [kg] \tag{95}$$

10. Using LISA PathFinder (LPF) as test case for the SpaceART Power system model

As described above, LPF has a fully regulated DC electrical power system. Power is generated from one body mounted solar array consisting of triple junction GaAs cells. A secondary battery for energy storage is used in various mission phases.

10.1. LPF Power System

The Power System utilizes:

One body mounted solar array, which will be represented by the Solar Array block of the SpaceART Power system model, based on its technical specifications.

One secondary battery, which will be represented by the Secondary Battery block of the SpaceART Power system model, based on its technical specifications.

The Power Control Unit block represents the power electronics on board the LPF Power System, handling and distributing power depending on each subsystem's power requirement. The Pwr_req block represents the sum of power requirements of all the individual subsystems on board.

Taking the aforementioned data into account, the SpaceART Power system software model will produce outputs that can size the LPF Power System's mass, power consumption and data output with high accuracy.

10.2. SpaceART Power system model LPF implementation

SpaceART is using the LPF as a test case in order to validate its Power system model. Taking into account the mission facts and figures, each input can become a design, uncertain or fixed/environmental parameter depending on their definition respectively:

Design parameters are defined as input parameters which can be decided by the designer(s), after taking into account several mission factors.

Fixed parameters (including environmental parameters) are defined as input parameters which are already known or given, therefore are considered constants.

Uncertain parameters are defined as input parameters which cannot be deterministically defined. Such parameters can either be design or fixed ones which cannot be defined with absolute certainty.

Taking the above definitions into account, the SpaceART Power system model inputs are defined as follows:

10.2.1. Solar Array (SA) LPF implementation

The Solar Array inputs consist of:

tsun – Orbital daylight time in hours (h) → Environmental/Orbital Parameter

tecl – Orbital eclipse time in hours (h) → Environmental/Orbital Parameter

theta – Worst case angle of incidence in degrees (rad) → Environmental/Orbital Parameter

G – Solar flux in Watts per square meter (W/m²) → Environmental/Orbital Parameter

Preq – Power consumption during daylight in Watts (W) → Environmental Parameter

Pecl – Power consumption during eclipse in Watts (W) → Environmental Parameter

Xe – Energy transfer during eclipse in percentage → Uncertain parameter

Xd – Energy transfer during daylight in percentage → Uncertain parameter

SAt – Solar array efficiency in percentage → Design parameter

Id – Inherent degradation in percentage → Uncertain parameter

Life – Satellite expected lifetime in years (yrs) → Environmental Parameter

Bus_Voltage – Spacecraft bus voltage in Volts (V) → Fixed parameter

Spec_mass_panel – Solar panel specific mass in kilograms per sq. m (kg/m²) → Design parameter

10.2.2. Power Control Unit (PCU) LPF implementation

The PCU inputs consist of:

npcu – PCU efficiency in percentage (%) → Uncertain parameter

Varray – Solar array Voltage output (V) → SA output (internal parameter)

Parray_total – Total solar array power output during daylight (W) → SA output (internal parameter)

a_1 – PCU mass coefficient → Design parameter

10.2.3. Secondary Battery (SB) LPF implementation

The SB inputs consist of:

Specific_en_dens – Secondary battery's specific energy density in Watthours per kilogram (Wh/kg) → $^{\text{Design}}$ parameter

v_drop – Maximum permissible bus voltage drop in percentage (%) → $^{\text{Design}}$ parameter

n_cycles – Number of cycles → Environmental/Orbital Parameter

tsun – Orbital daylight time in hours (h) → Environmental/Orbital Parameter

tecl – Orbital eclipse time in hours (h) → Environmental/Orbital Parameter

Pecl – Power consumption during eclipse in Watts (W) → Environmental Parameter

Vpcu – PCU Voltage output in Volts (V)→ PCU output (internal parameter)

11. Harness subsystem model: UML diagram

According to the Harness system UML diagram the Harness model (simple version) consists of 1 block, representing the actual physical unit. There is a total of 3 external input parameters for the Harness. The initial input is “flag” which specifies which model characteristics will be used.

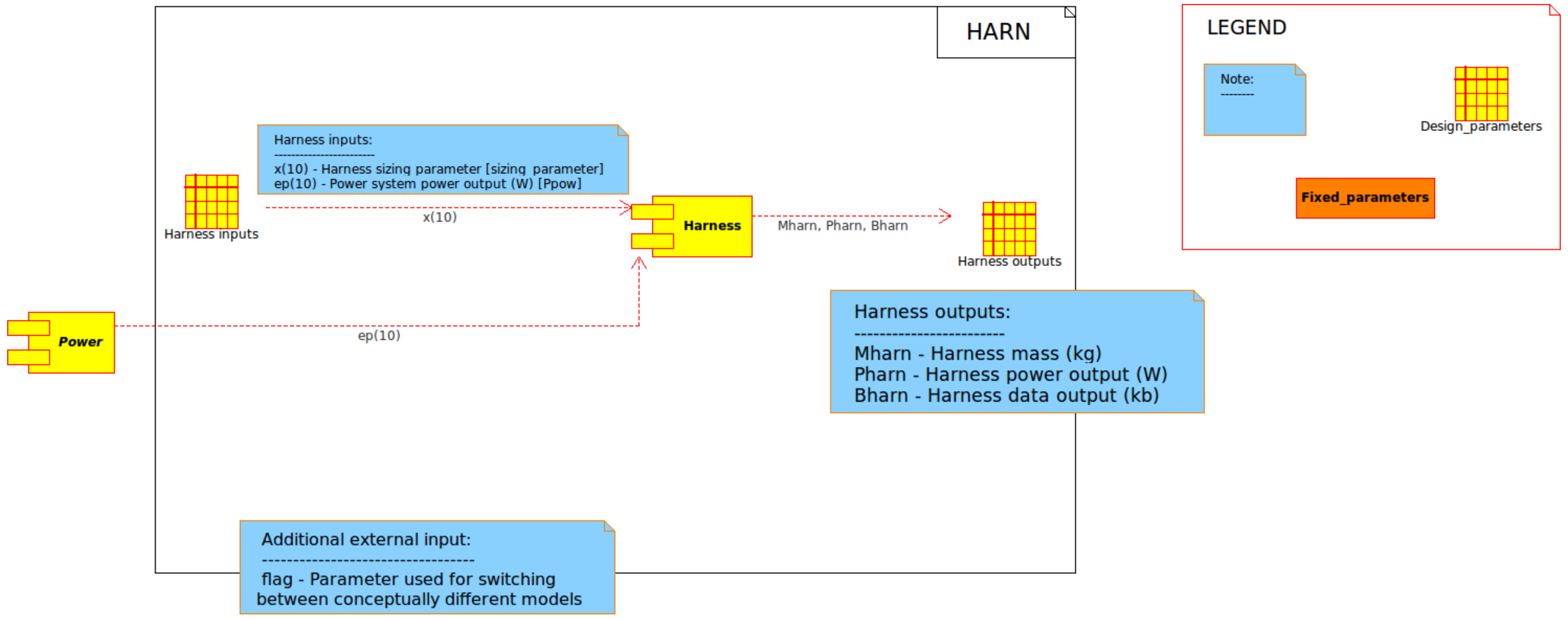

HARN
Harness inputs:
x(10) - Harness sizing parameter [sizing parameter]
ep(10) - Power system power output (W) [Ppow]
Harness inputs
x(10)
Harness
Mharn, Pharn, Bharn
Harness outputs
Power
ep(10)
Harness outputs:
Mharn - Harness mass (kg)
Pharn - Harness power output (W)
Bharn - Harness data output (kb)
Additional external input:
flag - Parameter used for switching between conceptually different models
LEGEND
Note:
Design_parameters
Fixed_parameters

Figure 11. Harness Model UML Diagram

The Harness receives inputs from external sources like the Power system (represented as one block for simplicity) that contribute to sizing the Harness system.

11.1. Harness:

The Harness unit represents all the galvanic connections between systems, used for transferring power and signals between the satellite's hardware units.

The Harness inputs consist of:

sizing_parameter – Parameter used for sizing the harness with respect to the overall power it handles

Ppow – Power system power output in Watts (W) → Power system external output

The output parameters are:

Mharn – Harness mass in kilograms (kg)

Pharn – Harness power output in Watts (W)

Bharn – Harness data output in kilobytes (kb)

11.2. SpaceART Harness System model function

```
%%     Harness system
%
%       [Mharn,Pharn,Bharn] = harn (x,ep,flag)
%
%    DESCRIPTION
%    Sizes the spacecraft harness mass utilizing the Power system output (simple model)
%
%    INPUT
%   *x - All the design and uncertain parameters
%      x(10) = sizing_parameter - Parameter sizing the HARN mass as a percentage of the total solar
%                array output power (%)
%   *ep - All the environmental parameters (i.e inputs which are fixed, that are neither design nor
%         uncertain parameters)
%      ep(10) = Ppow - Power system power output
%   *flag - Flag used to change between system types e.g from Harn simple to Harn advanced model
%
%    OUTPUT
%       Mharn - Harness mass (kg)
%       Pharn - Harness power output (W)
%       Bharn - Harness data output (kb)
%
%    FUNCTION CALLS
%        none
%
%     REFERENCES
% [1] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz,
%     Microcosm Press & Kluwer Academic Publishers
```

11.3. SpaceART Harness System model description

As seen above, the harn.m contains one function, represented by a logical block of the UML model I.e a hardware unit of the actual physical system.

The "Harn" function models the physical and electrical characteristics of the harness system based on a number of inputs like the Power system power output and so on. Using analytical mathematical expressions, the function outputs physical and electrical characteristics of the harness. Initially the model calculates the Harness overall mass M_{harn} as a fraction of the overall power it handles, as follows:

$$M_{harn} = SPP_{pow} \quad [kg] \qquad (96)$$

where SP – Sizing parameter (%), P_{pow} – Power system power output (W) deriving from Eq. (81) since $P_{pow} = P_{pcu}$.. The Harness power output P_{harn} is calculated after taking into account power losses due to Ohmic resistance, thermal dissipation and so on as follows:

$$P_{harn} = f_{Ppow}P_{pow} \quad [W] \qquad (97)$$

where f_{Ppow} – 0.98, P_{pow} – Power system power output (W) deriving from Eq. (81) since $P_{pow} = P_{pcu}$. The Harness data output B_{harn} is calculated after summing all the data outputs of each individual system respectively, as follows:

$$B_{harn} = \sum_{a=systems} B_a \qquad (98)$$

where a – Every electronics based system onboard the satellite (Power, TT&C, C&DH, ADCS etc), Ba – Data output of each individual system onboard the satellite (kb).

11.4. SpaceART Harness system model LPF implementation

SpaceART is using the LPF as a test case in order to validate its Harness model. Taking into account the mission facts and figures, each input can become a design, uncertain or fixed/environmental parameter depending on their definition respectively:

Design parameters are defined as input parameters which can be decided by the designer(s), after taking into account several mission factors.

Fixed parameters (including environmental parameters) are defined as input parameters which are already known or given, therefore are considered constants.

Uncertain parameters are defined as input parameters which cannot be deterministically defined. Such parameters can either be design or fixed ones which cannot be defined with absolute certainty.

Taking the above definitions into account, the SpaceART Harness system simple model inputs are defined as follows:

The Harness inputs consist of:

sizing_parameter – Parameter used for sizing the harness with respect to the overall power it handles → Uncertain parameter

Ppow – Power system power output in Watts (W) → Power system external output

11.5. References

Spacecraft Power Systems, Mukund R. Patel, CRC Press
Spacecraft Systems Engineering 3rd ed. , P. Fortescue, J. Stark, G. Swinerd, Wiley
Spacecraft Power Technologies, Anthony K. Hyder, Ronald L. Wiley, G. Halpert, Donna Jones Flood, S. Sabripour, World Scientific Publishing Company
Space Vehicle Design, Michael D. Griffin, James R. French, AIAA
Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz, Microcosm Press & Kluwer Academic Publishers
Maximum Fast Charging Rates in Ni-Cd Batteries, Viera Pérez, J. C. , Gonzalez, M. , Campo, J. C. , & Ferrero, F. J. , Space Power, Proceedings of the Sixth European Conference held 6-10 May, 2002 in Porto, Portugal. Edited by A. Wilson. European Space Agency, ESA SP-502, 2002, p.565
http://teva2.com/NiH2.html
Guidelines on Lithium-ion Battery Use in Space Applications, Barbara McKissock, Patricia Loyselle, and Elisa Vogel, RP-08-75, NASA

12. Propulsion subsystem model: UML diagram

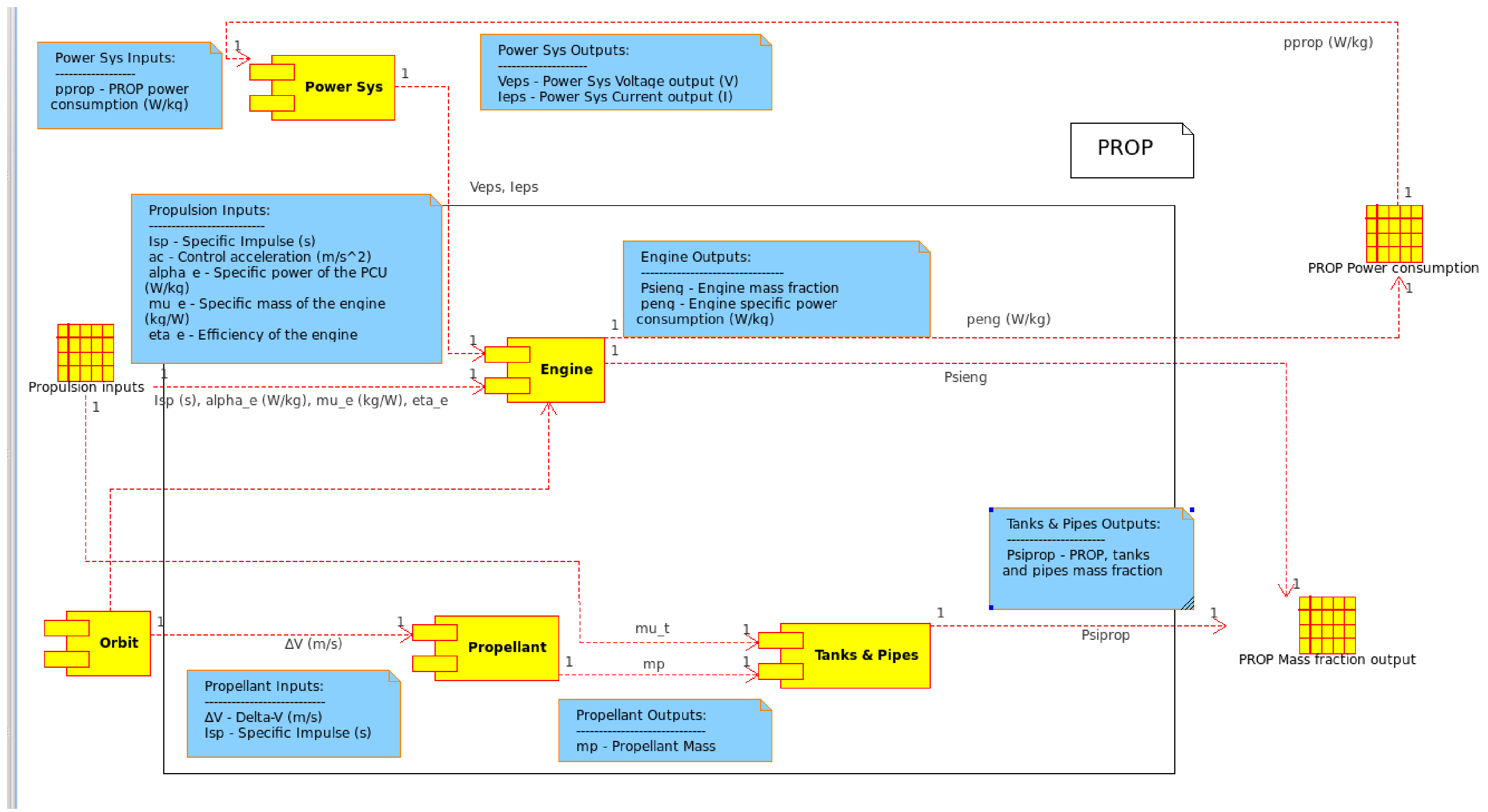

Power Sys Inputs:
pprop - PROP power consumption (W/kg)
Power Sys
Power Sys Outputs:
Veps - Power Sys Voltage output (V)
Ieps - Power Sys Current output (I)
pprop (W/kg)
PROP
Veps, Ieps
Propulsion Inputs:
Isp - Specific Impulse (s)
ac - Control acceleration (m/s^2)
alpha_e - Specific power of the PCU (W/kg)
mu_e - Specific mass of the engine (kg/W)
eta_e - Efficiency of the engine
Engine Outputs:
Psieng - Engine mass fraction
peng - Engine specific power consumption (W/kg)
PROP Power consumption
peng (W/kg)
Propulsion inputs
Isp (s), alpha_e (W/kg), mu_e (kg/W), eta_e
Engine
Psieng
Tanks & Pipes Outputs:
Psiprop - PROP, tanks and pipes mass fraction
Orbit
ΔV (m/s)
Propellant
mu_t
mp
Tanks & Pipes
Psiprop
PROP Mass fraction output
Propellant Inputs:
ΔV - Delta-V (m/s)
Isp - Specific Impulse (s)
Propellant Outputs:
mp - Propellant Mass
1

Figure 12. Propulsion Model UML Diagram

The PROP model consists of 3 blocks, each one representing a unit of the Propulsion system. There are a total of 7 input parameters for the PROP system plus a flag which is used to select the type of propulsion model (only one model is currently available).

12.1. *Engine*

The Engine block represents the engine(s) used for thrusting the spacecraft. The Engine inputs are:

alpha_e - Engine's specific power measured in Watts per kilogram (W/kg)

η_e - Engine efficiency measured in percentage

ac - Control acceleration measured in meters per second squared (m/s^2) (Orbit output)

μ_{eng}- Specific mass of the engine (kg/W)

The output parameters are:

m_{eng} – specific engine mass

m_{ppu}- specific mass of the PPU

p_{eng} – specific engine power consumption in W/kg

the_{eng} – specific engine thermal power output in W/kg

12.2. *Propellant*

The Propellant block represents the propellant properties used in a spacecraft, depending on the type of engines and the mission facts.

The Orbit block feeds the Propellant block with specific information used for sizing the Propellant mass.

The Propellant inputs consist of

Isp - Engine's specific impulse measured in seconds (s)

ΔV – Total change in velocity (m/s)

The output parameters are:
m_p – specific propellant mass

12.3. *Tanks and Pipes*

The tanks and pipes block represents all containers used for storing and distributing propellant on board the spacecraft. The tanks and pipes inputs are:

m_p – specific propellant mass

μ_t – mass fraction of the tanks and pipes

The output parameters are:
m_t – specific mass of tanks and pipes

The total specific mass of the propulsion system is $Psi_{prop}= m_t+m_{eng}+m_p+m_{ppu}$ with the associated specific power p_{eng} and thermal power the_{eng}.

12.4. SpaceART PROP model function

```
%
% [Phiprop,Psiprop, Pprop,Theprop] = prop_model_norm(x,ep,flag)
% Propulsion system model
%
%   INPUT:
%   Isp     = Specific Impulse (s)
%   alpha   = Specific Power (W/kg)
%   eta     = engine Efficiency
%   ep(1)   = Dv (m/s)
%   ep(2)   = ac control acceleration (m/s^2)
%   ep(3)   = percentage of the propellant that corresponds to
tanks and
%   pipes
%   ep(4)   = specific mass of the engine (kg/W)
%   flag    = switch between chemical and electric propulsion
%
%   OUTPUT:
%   Phiprop,Psiprop,    = Propulsion Mass functions (kg)
%   Pprop   = Power (W)
%   Theprop = thermal power rejected by the engine (W)
%
```

12.5. SpaceART Propulsion System Model Description

The mass and power of the propulsion system are not computed in absolute terms but with respect to the total mass of the spacecraft, therefore, in the following, we refer to specific mass and specific power. The total specific mass is composed of the specific mass of the propellant, the specific mass of the power unit, the specific mass of tanks and pipes, the specific mass of the engines. The power consumption depends on the engine efficiency η, the specific impulse I_{sp} and the control acceleration a_c. It is assumed that the control acceleration comes from the orbit design. The specific power required to operate the engine can be expressed as:

$$p_{eng} = \frac{a_c I_{sp}}{2\eta} \tag{99}$$

The specific mass of the engine is therefore given by:

$$m_{eng} = \mu_{eng} p_{eng} \tag{100}$$

and the specific mass of the power processing unit (PPU) is given by:

$$m_{ppu} = p_{eng} / \alpha_{eng} \tag{101}$$

The specific mass of propellant is a function of the total Δv, again derived from the orbit:

$$m_{prop} = 1 - e^{-\frac{\Delta v}{g_0 I_{sp}}} \tag{102}$$

The mass of tanks and pipes is computed as a fraction of the mass of propellant, although the mass of the pipes is actually dependent on the configuration:

$$m_{\mathrm{tank}} = \mu_{\mathrm{tank}} m_{prop} \tag{103}$$

The total specific mass of the propulsion system (or mass function) is therefore given by:

$$\Psi_{pro} = m_{eng} + m_{prop} + m_{tank} + m_{ppu} \tag{104}$$

and the associated specific power is:

$$p_{pro} = p_{eng} \tag{105}$$

12.6. Using LPF as test case for SpaceART PROP

The Propulsion module for LPF currently considers only the SEP micropropulsion unit on board LPF.

12.6.1. SpaceART PROP model applied to the LPF micropropulsion case.

The design, uncertain and environmental parameters for the LPF case are as follows:

12.6.1.1. Engine

The Engine block represents the engine(s) used for thrusting the spacecraft. The Engine inputs are:

alpha_e - Engine's specific power measured in Watts per kilogram (W/kg) - Design

η_e - Engine efficiency measured in percentage - Uncertain

ac - Control acceleration measured in meters per second squared (m/s^2) (Orbit output)- Environment

μ_{eng}- Specific mass of the engine (kg/W) – Design

The output parameters are:

m_{eng} – specific engine mass

p_{eng} – specific engine power consumption in W/kg

the_{eng} – specific engine thermal power output in W/kg

12.6.1.2. Propellant:

The Propellant block represents the propellant properties used in a spacecraft, depending on the type of engines and the mission facts.

The Orbit block feeds the Propellant block with specific information used for sizing the Propellant mass.

The Propellant inputs consist of

Isp - Engine's specific impulse measured in seconds (s) - Design

ΔV – Total change in velocity (m/s) – Environment

The output parameters are:
m_p – specific propellant mass

12.6.1.3. Tanks and Pipes

The tanks and pipes block represents all containers used for storing and distributing propellant on board the spacecraft. The tanks and pipes inputs are:

m_p – specific propellant mass

μ_t – mass fraction of the tanks and pipes - uncertain

The output parameters are:
m_t – specific mass of tanks and pipes

13. Thermal subsystem model: UML diagram

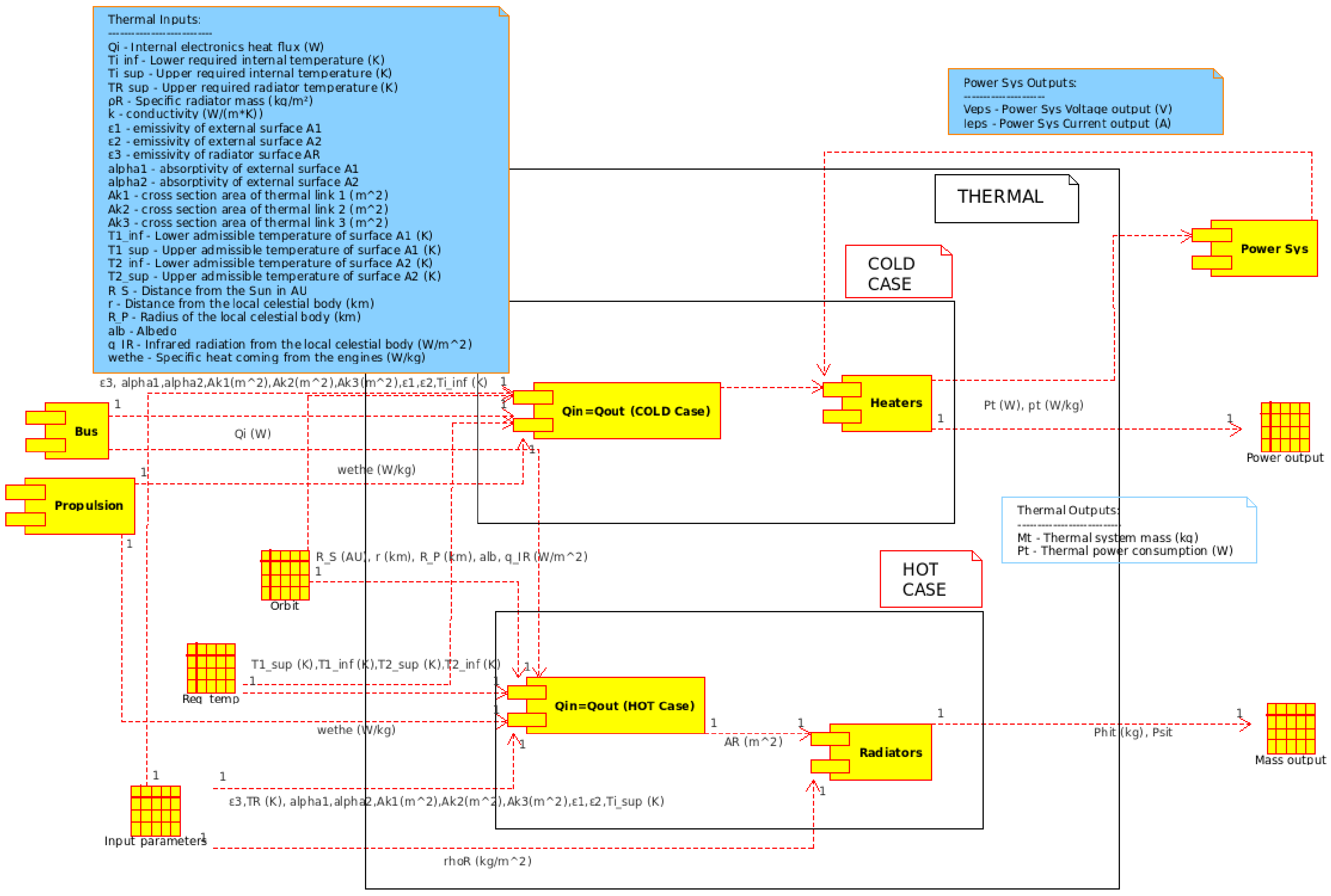

Thermal Inputs:
Qi - Internal electronics heat flux (W)
Ti inf - Lower required internal temperature (K)
Ti sup - Upper required internal temperature (K)
TR sup - Upper required radiator temperature (K)
ρR - Specific radiator mass (kg/m²)
k - conductivity (W/(m*K))
ε1 - emissivity of external surface A1
ε2 - emissivity of external surface A2
ε3 - emissivity of radiator surface AR
alpha1 - absorptivity of external surface A1
alpha2 - absorptivity of external surface A2
Ak1 - cross section area of thermal link 1 (m^2)
Ak2 - cross section area of thermal link 2 (m^2)
Ak3 - cross section area of thermal link 3 (m^2)
T1_inf - Lower admissible temperature of surface A1 (K)
T1_sup - Upper admissible temperature of surface A1 (K)
T2_inf - Lower admissible temperature of surface A2 (K)
T2_sup - Upper admissible temperature of surface A2 (K)
R_S - Distance from the Sun in AU
r - Distance from the local celestial body (km)
R_P - Radius of the local celestial body (km)
alb - Albedo
q_IR - Infrared radiation from the local celestial body (W/m^2)
wethe - Specific heat coming from the engines (W/kg)
Power Sys Outputs:
Veps - Power Sys Voltage output (V)
Ieps - Power Sys Current output (A)
THERMAL
COLD CASE
HOT CASE
Power Sys
Heaters
Qin=Qout (COLD Case)
Qin=Qout (HOT Case)
Radiators
Bus
Propulsion
Orbit
Req temp
Input parameters
Power output
Mass output
Thermal Outputs:
Mt - Thermal system mass (kg)
Pt - Thermal power consumption (W)
ε3, alpha1,alpha2,Ak1(m^2),Ak2(m^2),Ak3(m^2),ε1,ε2,Ti_inf (K)
Qi (W)
wethe (W/kg)
R_S (AU), r (km), R_P (km), alb, q_IR (W/m^2)
T1_sup (K),T1_inf (K),T2_sup (K),T2_inf (K)
wethe (W/kg)
ε3,TR (K), alpha1,alpha2,Ak1(m^2),Ak2(m^2),Ak3(m^2),ε1,ε2,Ti_sup (K)
rhoR (kg/m^2)
AR (m^2)
Pt (W), pt (W/kg)
Phit (kg), Psit
1

Figure 13 Thermal Model UML Diagram

As seen above, the Thermal system contains the Thermal network block and two modules -a hot and a cold one- representing both thermal cases used for designing a Thermal system. The Cold case contains 3 blocks whereas the Hot case 2 blocks. There is a total of 14 input parameters for the Thermal system.

13.1. Thermal network

The Thermal network block represents the thermal resistance network equivalent inside the spacecraft.

13.1.1. Cold Case

The cold-case block input parameters are:

Qi (COLD) – The heat generated by all the subsystems (W)

Ti_inf – lower required temperature inside the spacecraft (K)

T_{1_inf},T_{2_inf} – lower limit on the temperature of the external surfaces (K)

Its outputs are

Pt – Thermal system heaters' power consumption (W)

13.1.2. Hot Case

The hot-case input parameters are:

Q_i (HOT) – The heat generated by all the subsystems except for propulsion and attitude control (W)

ρR – Specific radiator mass in kilograms per square meter (kg/m^2)

k – the materials' conductivity measured in Watts per meter Kelvin (W/(m*K))

ε_1, ε_2 – the emissivity of the external surfaces

ε_R – the emissivity of the radiator

α_1, α_2 – the absorptivity of the external surfaces

T_{1_sup},T_{2_sup} – upper limit on the temperature of the external surfaces (K)

T_R – temperature of the radiator (K)

Ti_sup – upper required temperature inside the spacecraft (K)

A,A_2 – areas of the external surfaces (m^2)

Ak1,Ak2,Ak3 – cross section areas of the thermal links (m^2)

we – specific dissipated power from the propulsion system (W/kg)

Q_i – sum of all the internal sources of heat except for propulsion and attitude control (W)

r_S - distance from the Sun in AU

r - distance from the planet in km

R_P - radius of the planet in km

alb – albedo

q_IR - areal infrared radiation from the planet (W/m^2)

Its outputs size the radiators.

13.1.3. Radiators

The Radiators block represents the hardware used for radiating excessive heat into space. It is sized from the Qin=Qout output. Its sole output is

Mt – Thermal system mass for the Hot case, measured in kilograms (kg)

Ml – mass of the thermal links (kg)

13.2. SpaceART Thermal System model function

```
%
%  [Phithe,Psithe,P_the]=thermal(x,ep,flag)
%
%  Thermal model with 4 nodes.
%
%                Qi
%  Q_Sun    ->o---o---o-> AR*sigma*eps*TR^4
%                 |
%                 o
%                 ^
%                 | Q_IR+Q_ALB
%
%  It computes the required radiator size and heater power
%  to maintain the internal temperature within the prescribed
limits
%
%   INPUT:
%
%   Design/Uncertain
%
%   k=x(1)               : thermal conductivity of the thermal links
(W/mK)
%   Ak1=x(2)             : cross section area of the thermal link 1
(m^2)
%   Ak2=x(3)             : cross section area of the thermal link 2
(m^2)
%   Ak3=x(4)             : cross section area of the thermal link 3
(m^2)
%   eps(1)=x(5)          : emissivity of external surface A1
%   eps(2)=x(6)          : emissivity of external surface A2
%   eps(3)=x(7)          : emissivity of radiator surface AR
%   alpha(1)=x(8)        : absorptivity of external surface A1
%   alpha(2)=x(9)        : absorptivity of external surface A2
%   Ti_sup=x(10)        : hot case desired temperature of internal
components (K)
%   Ti_inf=x(11)         : cold case desired temperature of internal
components(K)
%
%   Environmental
%
%   Q_i=ep(1)            : internal heat
%   r_S=ep(2)            : distance from the Sun in AU
%   r=ep(3)              : distance from the planet in km
%   R_P=ep(4)            : radius of the planet in km
%   alb=ep(5)            : albedo
%   rho_R=ep(6)          : areal density of the radiator
%   q_IR=ep(7)           : areal infrared radiation from the planet
%   T1_limits=ep(8:9)    : upper and lower limits on the temperature
of A1
%   T2_limits=ep(10:11): upper and lower limits on the temperature
of A2
```

```
%   A1=ep(12)          : external area A1
%   A2=ep(13)          : external area A2
%   we=ep(14)          : specific dissipated power from the engine
%
%   OUTPUT
%
%    Phithe,Psithe: mass functions of the thermal system
%    P_the: power of the heaters
%    T_cold : temperatures in the cold case
%    T_hot : temperatures in the hot case
```

13.3. Thermal System Model Description

The thermal control system is modeled with an equivalent Oppenheim network with four nodes and considering steady state conditions.

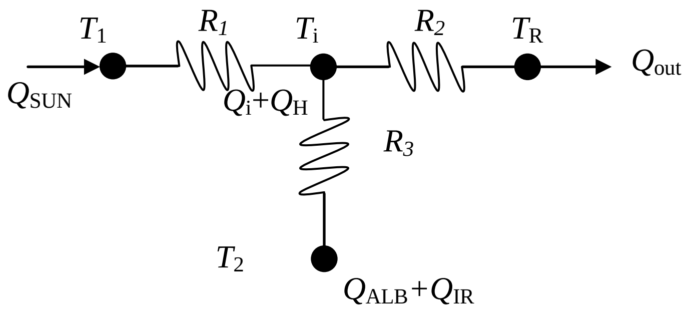


Figure 14. Equivalent thermal network

The external input heat comes from the faces exposed to the Sun and to the celestial body around which the spacecraft is orbiting. Therefore, one can have three contributions: direct Sun radiation Q_{SUN}, reflected Sun radiation Q_{ALB}, infrared radiation from the celestial body surface Q_{IR}.

$$Q_{in} = Q_{SUN} + Q_{ALB} + Q_{IR} \tag{106}$$

with:

$$Q_{SUN} = A_1 \alpha_1 \frac{S_0}{r_{SUN}^2} \cos\theta_1 \tag{107}$$

$$Q_{ALB} = \alpha_2 A_2 av \frac{R_B^2}{R_B^2 + r^2} \cos\theta_2 \tag{108}$$

$$Q_{IR} = q_{IR} \varepsilon_2 A_2 \frac{R_B^2}{R_B^2 + r^2} \cos\theta_2 \tag{109}$$

The two input nodes radiate towards outer space part of the incoming radiation:

$$Q_1 = A_1 \varepsilon_1 \sigma T_1^4 \tag{110}$$

$$Q_2 = A_2 \varepsilon_2 \sigma T_2^4 \tag{111}$$

The heat that is not rejected through radiation is transferred to the internal node, therefore the input nodes steady state equations are:

$$Q_{SUN} - A_1\varepsilon_1\sigma T_1^4 - R_1(T_1 - T_i) = 0 \tag{112}$$

$$Q_{IR} + Q_{ALB} - A_2\varepsilon_2\sigma T_2^4 - R_2(T_2 - T_i) = 0 \tag{113}$$

The internal node represents all internal components that require a careful thermal control (batteries, electronics, tanks, etc.). It receives part of the heat coming from the external input node and the heat internally generated Q_i. If the temperature is too low, an electric heating system generating Q_H is actively maintaining the temperature in the desired range. If the temperature is too high, the internal node dissipates through a thermal link R_2 with a radiator with area A_R. The steady state equation at the node is:

$$Q_i + Q_H + R_1(T_1 - T_i) + R_2(T_2 - T_i) - R_3(T_i - T_R) = 0 \tag{114}$$

The radiator is assumed to always face deep space, therefore its steady state equilibrium equation is:

$$R_2(T_i - T_R) - \sigma\varepsilon_R A_R T_R^4 = 0 \tag{115}$$

The thermal model is currently based on the assumption that the spacecraft always encounters a hot case during the mission and that it might encounter a cold in which the heaters are required. Therefore, the first step computes the area of the radiator that is required to maintain the three nodes at the required temperature.
The thermal links R_1, R_2 and R_3 are computed from the thermal conductivity, the cross section areas Ak_1,Ak_2 and Ak_3, and the length of the longest edge of the spacecraft l_{edge}. The thermal path length is assumed to be maximum half of the edge of the spacecraft, therefore:

$$R_i = \frac{2kA_{ki}}{l_{edge}} \quad i = 1,..,3 \tag{116}$$

The size of the external surfaces is assumed to come from the configuration block and the edge is computed as the square root of the area. The temperatures T_1 and T_2 are obtained by solving the quartic equations (112) and (113).
If the temperature T_1, respectively T_2, is above the upper limit, it is set to the upper limit and a $Q_{\sup}$ is added to Eq. (112) such that:

$$Q_{SUN} - A_1\varepsilon_1\sigma T_1^4 - R_1(T_1 - T_i) = Q_{\sup 1} \tag{117}$$

likewise for Eq. (113) one has:

$$Q_{IR} + Q_{ALB} - A_2\varepsilon_2\sigma T_2^4 - R_2(T_2 - T_i) = Q_{\sup 2} \tag{118}$$

If the temperature T_1, respectively T_2, is below the lower limit, it is set to the lower limit and a $Q_{\inf}$ is added to Eq. (112) such that:

$$Q_{SUN} - A_1\varepsilon_1\sigma T_1^4 - R_1(T_1 - T_i) = Q_{\inf 1} \tag{119}$$

likewise for Eq. (113) one has:

$$Q_{IR} + Q_{ALB} - A_2\varepsilon_2\sigma T_2^4 - R_2(T_2 - T_i) = Q_{\inf 2} \tag{120}$$

$Q_{\sup 1}$ and $Q_{\sup 2}$ imply a direct link to the radiator, therefore additional thermal links are introduced with masses given by:

$$\begin{aligned} m_{\sup 1} &= \frac{Q_{\sup 1} l_{edge}^2 \rho_{material}}{k(T_1 - T_R)} \\ m_{\sup 2} &= \frac{Q_{\sup 2} l_{edge}^2 \rho_{material}}{k(T_2 - T_R)} \end{aligned} \tag{121}$$

By combining Eq. (114) and (115) one gets the area of the radiator:

$$A_R = \frac{Q_i + R_1(T_1 - T_i) + R_2(T_2 - T_i) + Q_{\sup 1} + Q_{\sup 2}}{\sigma\varepsilon_R T_R^4} \tag{122}$$

If the area of the radiator is zero or negative the thermal link R_3 is set to 0 and a heating power is added to Eq. (114).
Once the radiator for the hot case is defined, the analysis of the cold case yields the sizing of the required power for the heaters. In fact, in the cold case one can assume that there is no direct Sun light or reflected radiation. Therefore, Eqs. (112) and (113) reduce to:

$$-A_1\varepsilon_1\sigma T_1^4 - R_1(T_1 - T_i) = 0 \qquad (123)$$
$$Q_{IR} - A_2\varepsilon_2\sigma T_2^4 - R_2(T_2 - T_i) = 0 \qquad (124)$$

Eqs. (123) and (124) are quartic equations in T_1 and T_2 and assuming the desired value for the internal temperature, one can find four complex conjugate solutions for each equation. Only the real positive solutions are retained and in particular the ones with a temperature higher than T_i.
Similarly, Eq. (115) can be solved for T_R. In this case the positive, real temperature lower than T_i is considered. Finally, the required heating power Q_H is computed from Eq. (114):

$$Q_H = -Q_i - R_1(T_1 - T_i) - R_2(T_2 - T_i) + R_3(T_i - T_R) \qquad (125)$$

The heating power is sized based on the larger one between Q_inf1+Q_inf2 and Q_H. The total mass of the thermal system is the sum of the radiator mass and the thermal links:

$$m_H = A_R\rho_R + \sum_i^3 \frac{R_i l_{edge}^2}{k}\rho_{material} + m_{\mathrm{sup1}} + m_{\mathrm{sup2}} \qquad (126)$$

13.4. *Using LPF as test case for SpaceART Thermal*

LPF thermal control system consists of MLI blankets, radiators and electric heaters, which guarantee a very stable environment inside the spacecraft.

13.4.1. LPF Thermal control system

The Thermal control module utilizes:

Radiators used for radiating any excess heat into space, which will be represented by the Radiators block in both Hot and Cold cases.

Electrical heaters used for keeping the ambient temperature within the lower electronics allowable range, which will be represented by the Heaters block.

13.4.2. Cold Case

The cold-case block input parameters are:

Q_i (COLD) – The heat generated by all the subsystems except for the propulsion and attitude control systems (W) - environmental

Ti_lower – lower required temperature inside the spacecraft (K) - design

T_{1_low},T_{2_low} – lower limit on the temperature of the external surfaces (K) - environmental

Its outputs are

Pt – Thermal system heaters' power consumption (W)

13.4.3. Hot Case

The hot-case input parameters are:

Q_i (HOT) – The heat generated by all the subsystems (W) - environmental

ρR – Specific radiator mass in kilograms per square meter (kg/m²) - design

k – the materials' conductivity measured in Watts per meter Kelvin (W/(m*K)) - design

ε_1, ε_2 – the emissivity of the external surfaces- uncertain

ε_R – the emissivity of the radiator- design

α_1, α_2 – the absorptivity of the external surfaces- uncertain

T_R – temperature of the radiator (K) - design

T_{1_up},T_{2_up} – upper limit on the temperature of the external surfaces (K) - environmental

Ti_upper – upper required temperature inside the spacecraft (K) - design

A,A_2 – areas of the external surfaces (m^2) - environmental

Ak1,Ak2,Ak3 – cross section areas of the thermal links (m^2) - design

we – specific dissipated power from the propulsion system (W/kg) - environmental

r_S - distance from the Sun in AU- environmental

r - distance from the planet in km- environmental

R_P - radius of the planet in km- environmental

alb – albedo- environmental

q_IR - areal infrared radiation from the planet (W/m^2) - environmental

Its outputs size the radiators.

13.4.4. Radiators

The Radiators block represents the hardware used for radiating excessive heat into space. It is sized from the Qin=Qout output. Its sole output is

Mt – Thermal system mass for the Hot case, measured in kilograms (kg)

Ml – mass of the thermal links (kg)

.

14. TT&C subsystem model: UML diagram

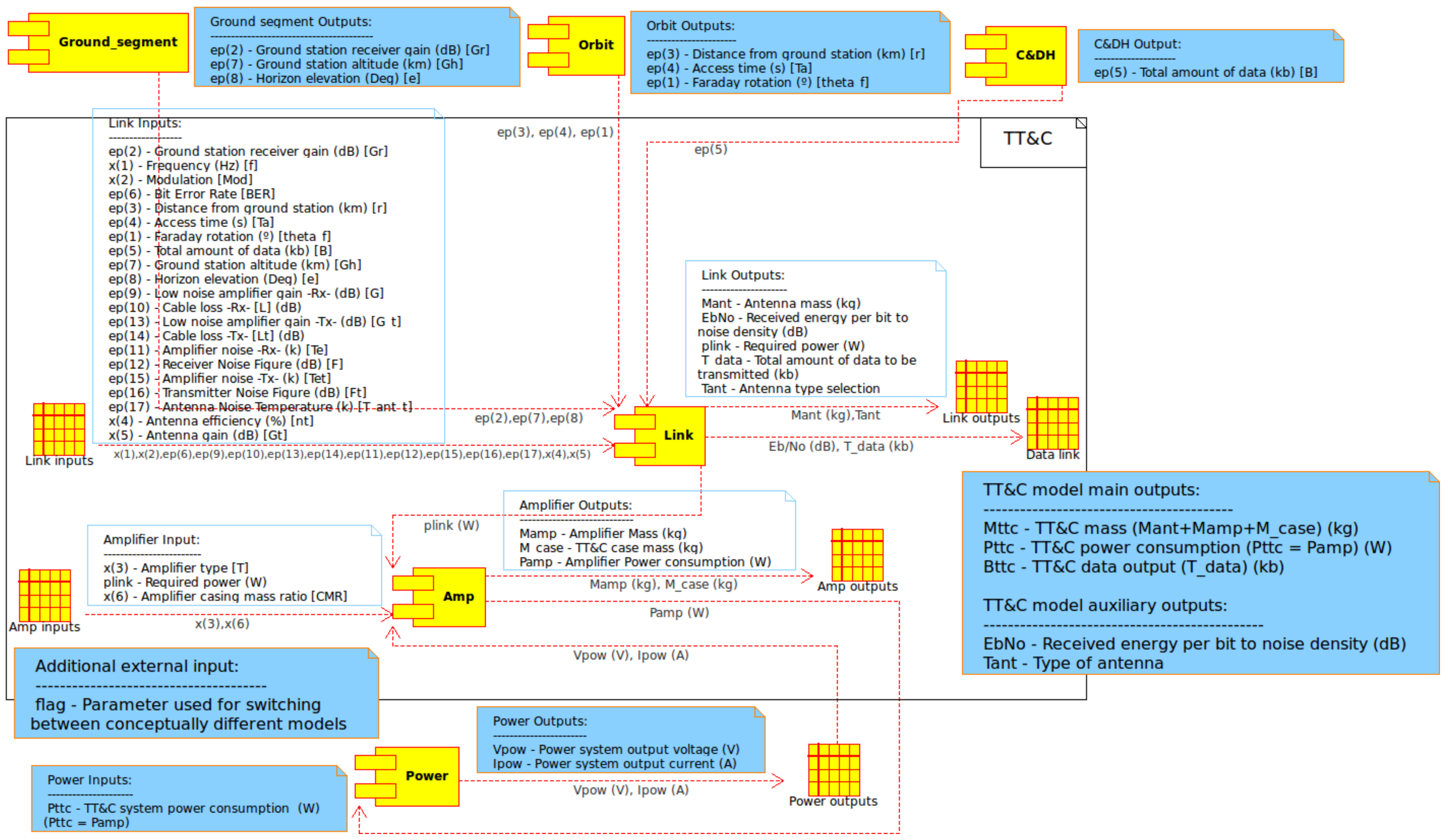

Ground_segment
Ground segment Outputs:
ep(2) - Ground station receiver gain (dB) [Gr]
ep(7) - Ground station altitude (km) [Gh]
ep(8) - Horizon elevation (Deg) [e]
Orbit
Orbit Outputs:
ep(3) - Distance from ground station (km) [r]
ep(4) - Access time (s) [Ta]
ep(1) - Faraday rotation (º) [theta f]
C&DH
C&DH Output:
ep(5) - Total amount of data (kb) [B]
TT&C
Link Inputs:
ep(2) - Ground station receiver gain (dB) [Gr]
x(1) - Frequency (Hz) [f]
x(2) - Modulation [Mod]
ep(6) - Bit Error Rate [BER]
ep(3) - Distance from ground station (km) [r]
ep(4) - Access time (s) [Ta]
ep(1) - Faraday rotation (º) [theta f]
ep(5) - Total amount of data (kb) [B]
ep(7) - Ground station altitude (km) [Gh]
ep(8) - Horizon elevation (Deg) [e]
ep(9) - Low noise amplifier gain -Rx- (dB) [G]
ep(10) - Cable loss -Rx- [L] (dB)
ep(13) - Low noise amplifier gain -Tx- (dB) [G t]
ep(14) - Cable loss -Tx- [Lt] (dB)
ep(11) - Amplifier noise -Rx- (k) [Te]
ep(12) - Receiver Noise Figure (dB) [F]
ep(15) - Amplifier noise -Tx- (k) [Tet]
ep(16) - Transmitter Noise Figure (dB) [Ft]
ep(17) - Antenna Noise Temperature (k) [T ant t]
x(4) - Antenna efficiency (%) [nt]
x(5) - Antenna gain (dB) [Gt]
ep(3), ep(4), ep(1)
ep(5)
Link Outputs:
Mant - Antenna mass (kg)
EbNo - Received energy per bit to noise density (dB)
plink - Required power (W)
T data - Total amount of data to be transmitted (kb)
Tant - Antenna type selection
Link inputs
x(1),x(2),ep(6),ep(9),ep(10),ep(13),ep(14),ep(11),ep(12),ep(15),ep(16),ep(17),x(4),x(5)
ep(2),ep(7),ep(8)
Link
Mant (kg),Tant
Eb/No (dB), T_data (kb)
Link outputs
Data link
Amplifier Outputs:
Mamp - Amplifier Mass (kg)
M case - TT&C case mass (kg)
Pamp - Amplifier Power consumption (W)
plink (W)
Amplifier Input:
x(3) - Amplifier type [T]
plink - Required power (W)
x(6) - Amplifier casing mass ratio [CMR]
Amp
Mamp (kg), M_case (kg)
Amp outputs
Pamp (W)
Amp inputs
x(3),x(6)
Vpow (V), Ipow (A)
Additional external input:
flag - Parameter used for switching between conceptually different models
TT&C model main outputs:
Mttc - TT&C mass (Mant+Mamp+M_case) (kg)
Pttc - TT&C power consumption (Pttc = Pamp) (W)
Bttc - TT&C data output (T_data) (kb)
TT&C model auxiliary outputs:
EbNo - Received energy per bit to noise density (dB)
Tant - Type of antenna
Power Outputs:
Vpow - Power system output voltage (V)
Ipow - Power system output current (A)
Power
Power outputs
Vpow (V), Ipow (A)
Power Inputs:
Pttc - TT&C system power consumption (W)
(Pttc = Pamp)

Figure 15TT&C system UML diagram

According to the TT&C system UML diagram the TT&C system model consists of 2 blocks, each one representing a separate unit. There is a total of 28 external input parameters for the TT&C system. The additional input “flag” specifies which model characteristics will be used. The TT&C system receives inputs from external sources like the Ground segment block , the Orbit block and the Command & Data Handling (C&DH) block representing external parameters that contribute to sizing parts of the TT&C.

14.1. Link

The TT&C system Link module defines the communication link architecture and design based on specific external inputs. The Link inputs consist of:

BER – Specified Bit Error Rate

Mod – Modulation selection

B – Total amount of data in kilobytes (kb) → C&DH output (internal parameter)

Ta – Ground station access time in seconds (s) → Environmental/Orbital Parameter

r – Distance from ground station in kilometres (km) → Environmental/Orbital Parameter

f – Frequency in Hertz (Hz)

Gr – Ground station receiver gain in decibels (dB)

theta_f – Faraday rotation in degrees (º) → Environmental/Orbital Parameter

Gh – Ground station altitude (km) → Environmental/Orbital Parameter

e – Horizon elevation (º) → Environmental/Orbital Parameter

G – Low noise amplifier gain (Receiver) → Environmental Parameter

L – Cable loss (Receiver) (dB) → Environmental Parameter

G_t – Low noise amplifier gain (Transmitter) (dB) → Environmental Parameter

Lt – Cable loss (Transmitter) (dB) → Environmental Parameter

Te – Amplifier noise (Receiver) (k)

F – Receiver noise figure (dB)

Tet – Amplifier noise (Transmitter) (k)

Ft – Transmitter noise figure (dB)

T_ant_t – Antenna noise temperature (k)

nt – Antenna efficiency (%)

Gt – Antenna gain (dB)

The output parameters are:

Mant – Antenna mass in kilograms (kg)

EbNo – Received energy per bit to noise density ratio (dB)

plink – Required link power in Watts (W)

T_data – Total amount of data to be transmitted in kilobytes (kb)

Tant – Antenna type selection

14.2. Amp

The TT&C Amp module sizes the amplifier unit based on both internal and external inputs.

The Amp inputs consist of:

T – Amplifier type selection

plink – Required link power in Watts (W) → Link internal output

CMR - Amplifier casing mass ratio

The output parameters are:

Mamp – Amplifier mass in kilograms (kg)

M_case – Electronics' casing mass in kilograms (kg)

Pamp – Amplifier power consumption in Watts (W)

14.3. TT&C system's external outputs

The TT&C system's external outputs are:

Mttc – TT&C total mass I.e the sum of Mant+Mamp+M_case (kg)

Pttc – TT&C total power consumption I.e the Pamp (W)

Bttc – TT&C total data output I.e T_data (kb)

Additionally the TT&C system provides an extra auxiliary output:

EbNo – Received energy per bit to noise density ratio (dB)

Tant - Type of antenna

14.4. SpaceART TT&C System model function

```
%%     Telemetry, Tracking & Commanding system
%
%    DESCRIPTION
%    Sizes the TT&C system based on Downlink Parameters
%    This model assumes worst case losses
%
%
%   [Mttc Pttc Bttc EbNo Tant] = ttc(x,ep,flag)
%
%
%   INPUT
%    *x - All the design and uncertain parameters
%      x(1) = f - Frequency (MHz)
%      x(2) = Mod - Modulation
%      x(3) = T - Amplifier type
%      x(4) = nt - Antenna efficiency
%      x(5) = Gt - Antenna gain (dB)
%      x(6) = CMR - Amplifier casing mass ratio
%
%    *ep - All the environmental parameters (i.e inputs which are fixed, that are neither design nor uncertain parameters)
%      ep(1) = theta_f - Farraday Rotation (Deg)
%      ep(2) = Gr - Ground Station Antenna Gain (dB)
%      ep(3) = r - Distance from Ground Station (km)
%      ep(4) = Ta - Access Time (s) *(Target acquisition time should be
%                  included)*
%      ep(5) = B - Total Amount of Data (kb) (C&DH output)
%      ep(6) = BER - Bit Error Rate
%      ep(7) = Gh - Ground station altitude (km)
%      ep(8) = e - Horizon elevation (Deg)
%      ep(9) = G - Low noise amplifier gain [dB] (Receiver)
%      ep(10) = L - Cable loss [dB] (Receiver)
%      ep(11) = Te - Amplifier noise (K) (Receiver)
%      ep(12) = F - Receiver Noise Figure (Receiver)
%      ep(13) = G_t - Low noise amplifier gain [dB]
%      ep(14) = Lt - Cable loss [dB] (Transmitter)
%      ep(15) = Tet - Amplifier noise (Transmitter) (K)
%      ep(16) = Ft - Receiver Noise Figure (Transmitter)
%      ep(17) = T_ant_t - Antenna Noise Temperature (K) (Transmitter)
%
%     *flag - Flag used to change between system types e.g from TTC DPSK modulation to TTC FSK modulation
%
%   OUTPUT
%      Mttc - TT&C system mass (kg)
%      Pttc - TT&C system power consumption (W)
%      Bttc - TT&C system data output (kb)
%      EbNo - Received energy per bit to noise density ratio (dB)
%      Tant - Type of antenna
```

```
%
%    FUNCTION CALLS
%        none
%
%    REFERENCES
% [1] Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz,
%    Microcosm Press & Kluwer Academic Publishers
% [2] Antenna Theory Analysis and Design 3rd ed. , Constantine A. Balanis, Wiley-Interscience publications
% [3] Satellite Communications 3rd ed. , Dennis Roddy, McGraw-Hill
% [4] Tutorial on Basic Link Budget Analysis, intersil(TM), Application Note 9804, Jum Zyren, Al Petrick, June 1998
% [5] www.atmmicrowave.com/wave-horn.html
% [6] http://www2.rfsworld.com/RFS_Edition3/pdfs/Microwave_Solid_Antennas_267-322.pdf
% [7] Satellite Technology, Principles and Applications, 2nd ed. , Anil K.
%    Maini, Varsha Agrawal, John Wiley & Sons Ltd.,

%% Internal functions called
[EbNo,Mant,plink,T_data,Tant] = link(BER,Mod,B,Ta,r,f,Gr,theta_f,Gh,e,G,L,G_t,Lt,Te,F,Tet,Ft,T_ant_t,nt,Gt)

[Mamp,M_case,Pamp] = amp(plink,T,CMR)


%% Total Mass, Power, Data
Mttc = M_case + Mamp + Mant

Pttc = Pamp

Bttc = T_data
```

As seen above, the ttc.m is divided into individual functions, each one representing a logical block of the UML model.

14.5. SpaceART TT&C System model description

14.5.1. Link

[EbNo,Mant,plink,T_data,Tant] = link(BER,Mod,B,Ta,r,f,Gr,theta_f,Gh,e,G,L,G_t,Lt,Te,F,Tet,Ft,T_ant_t,nt,Gt)

The "Link" function models the communication link characteristics based on specific inputs like the Bit Error Rate (*BER*), modulation (*Mod*), and ground station antenna gain (*Gr*). It outputs the mass of the antenna that can be used (Mant), the total amount of data to be transmitted (*T_data*) as well as the required link power (*plink*), the received energy per bit to noise density ratio (*EbNo*), and the type of antenna. Initially, the received energy per bit to noise density ratio (*EbNo*) is calculated, utilizing the information included in Figure 2. Using spline interpolation, the corresponding *EbNo* output is set based on the Modulation (*Mod*) and Bit Error Rate (*BER*) external inputs.

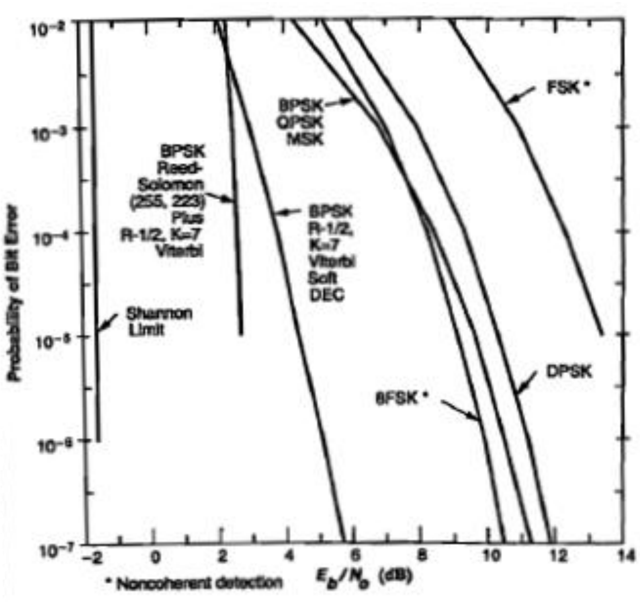


Figure 16 Bit Error Probability as a function of Eb/No

The total amount of data to be transmitted T_{data} is calculated as follows

$$T_{data} = B \cdot 10^3 \, [\mathrm{b}] \tag{127}$$

where B – Total amount of data (kb) passed from the C&DH (Command & Data Handling) system to the TT&C. After inputting the access time (T_a) in seconds, then the required data rate Rt is calculated based on Eq.(127) as follows

$$R_t = 10 \cdot \log_{10}\left(\frac{T_{data}}{\left(T_a - T_{aq}\right)} \right) [dB] \tag{128}$$

where T_{data} – The total amount of data to be transmitted (kb) deriving from Eq.(127), T_a – Access time (s), T_{aq} – Target acquisition time (s)

The Signal to Noise Ratio SN_{ratio} is then calculated in order to derive the Carrier to Noise Ratio CN_{ratio} as follows

$$\mathrm{SN}_{\mathrm{ratio}} = \mathrm{EbNo} \;\; [dB] \tag{129}$$

and

$$\mathrm{CN}_{\mathrm{ratio}} = \mathrm{SN}_{\mathrm{ratio}} + R_t \; [dB] \tag{130}$$

where *EbNo* – Received energy per bit to noise density ratio (dB) deriving from Figure 2, R_t – Required data rate deriving from Eq.(128).

The overall losses are then calculated using a series of mathematical equations. Firstly, the free space losses FS_L are calculated. The user inputs ground station distance r as well as the frequency f in order for the model to calculate FS_L as follows

$$\mathrm{FS}_L = 32.4 + 20\log_{10} r + 20\log_{10} f \;\; [\mathrm{dB}] \tag{131}$$

where r – Distance from ground station (km), f – Frequency (MHz). The Orbit block outputs the environmental parameter θ_f (Faraday rotation) which is passed as an external TT&C input leading to the calculation of the polarisation mismatch (Ionospheric) losses P_L as follows:

$$P_L = -20\log_{10}\left(\cos\theta_f\right) \;\; [\mathrm{dB}] \tag{132}$$

where θ_f – Faraday rotation (degrees). The atmospheric losses A_L are set based on the corresponding ground station altitude external input *Gh* using Table 8 below

Table 8: Atmospheric losses' change with ground station altitude

***Gh* (km)**	**A_L (dB)**
-2 to 2	0.04
2.1 to 6	0.025
6.1 to 10	0.008
10.1 to 14	0.004
14.1 to 18	0.001

Furthermore since the horizon elevation also affects the atmospheric losses, the atmospheric losses including the horizon elevation increase A_{LH} are calculated based on the horizon elevation external input e as follows

$$A_{\mathrm{LH}} = \frac{A_L}{\sin e} \, [\mathrm{dB}] \tag{133}$$

where e – horizon elevation (degrees), A_L – Atmospheric losses (dB) deriving from Table 8.

Based on the horizon elevation input e, the Rain absorption losses R_{aL} are then calculated by utilizing the data included in Figure 17.

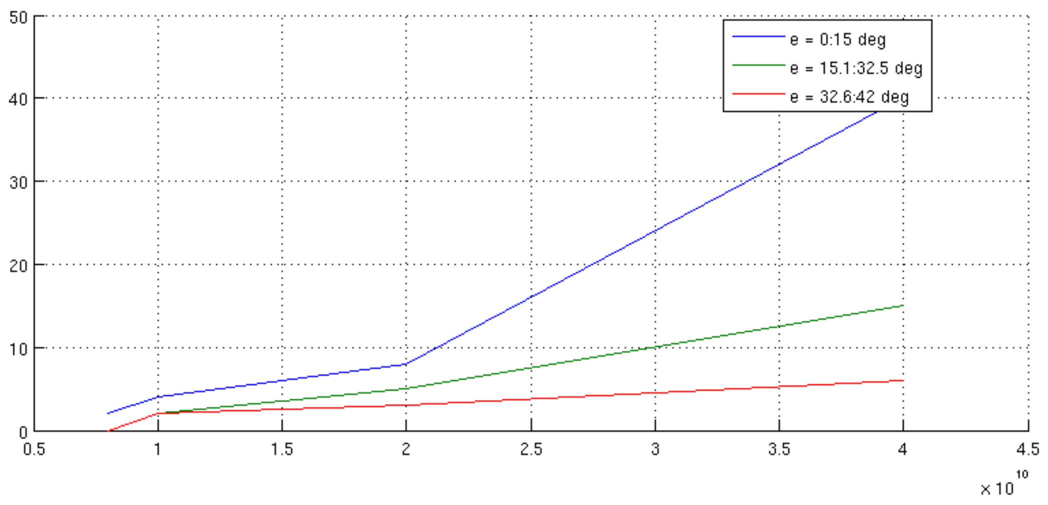


Figure 17 Rain absorption losses depending on the horizon elevation

Furthermore the worst case losses for the Feeder loss F_L, the Antenna misalignment loss AM_L and the Implementation loss I_L are taken into account as follows

Table 9: Worst case losses

F_L **[dB]**	AM_L **[dB]**	I_L **[dB]**
2	0.5	2

Summing up all the individual losses provides the total losses L_{TOTAL} for the given system as follows:

$$L_{\mathrm{TOTAL}} = \mathrm{FS}_L + F_L + \mathrm{AM}_L + A_{LH} + P_L + R_{aL} + I_L \ [\mathrm{dB}] \qquad (134)$$

where FS_L – Free space losses (dB) deriving from Eq. (131), F_L – Feeder loss (dB) deriving from Table 9, AM_L – Antenna misalignment loss (dB) deriving from Table 9, A_{LH} – Atmospheric losses including the horizon elevation increase (dB) deriving from Eq. (133), P_L – Polarisation mismatch (Ionospheric) losses (dB) deriving from Eq.(132), R_{aL} – Rain absorption losses (dB) deriving from Figure 17, I_L – Implementation loss (dB) deriving from Table 9.

Following the losses calculations, the system noise calculations are introduced. By interpolating a number of inputs, namely possible horizon elevations ranging from 1 to 90 degrees, possible frequencies ranging from 0.1 to 20 GHz and antenna temperatures ranging from 2 to 8000 Kelvin based on Figure 18, the Link function calculates the antenna noise temperature AN_{temp}.

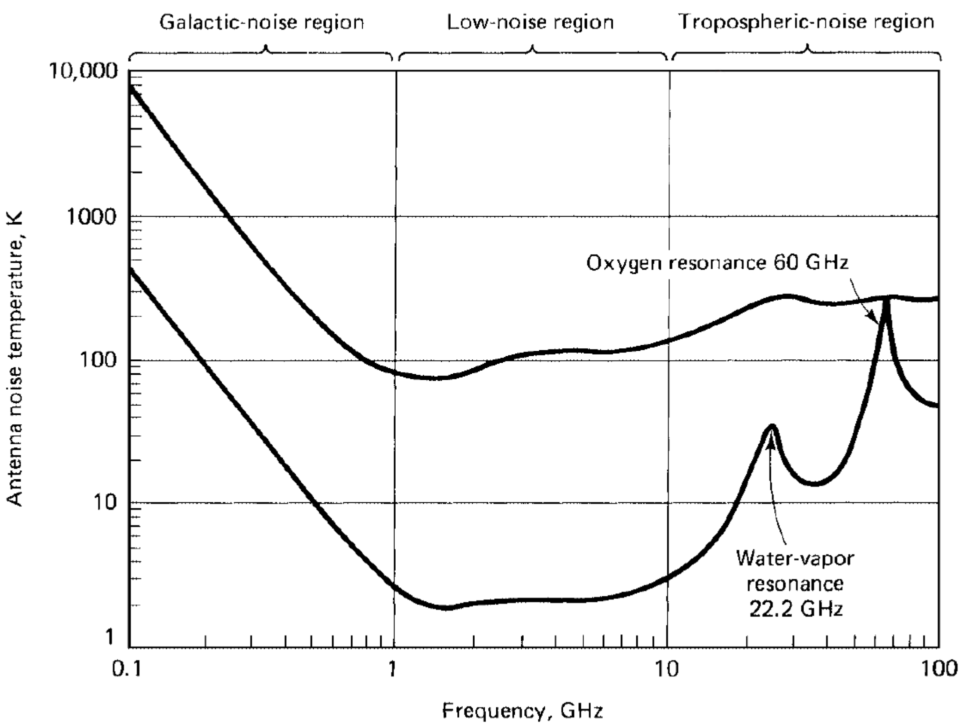


Figure 18: Irreducible noise temperature of an ideal ground-based

AN_{temp} is then utilized in order to derive the receiver noise figure $\mathrm{RN_{fig}}$ as follows:

$$\mathrm{RN_{fig}} = \mathrm{AN_{temp}} + T_e + \frac{\left(\left(10^{L/10}-1\right)k_o\right)}{10^{G/10}} + \frac{\left(10^{L/10}\right)\left(\left(10^{F/10}-1\right)k_o\right)}{10^{G/10}}\left[K\right] \qquad (135)$$

where AN_{temp} – Antenna noise temperature (K), T_e – Amplifier noise (K), L – Cable loss (dB), G – Low noise amplifier gain (dB), F – Receiver noise figure (dB) , k_o = 290 (K). Then the transmitter noise calculations are introduced. The first calculation refers to the system noise temperature S_{temp} based on the following equation:

$$S_{\mathrm{temp}} = \mathrm{AT_{tempT}} + T_{eT} + \frac{\left(\left(10^{L_T/10}-1\right)k_o\right)}{10^{G_T/10}} + \frac{\left(10^{L_T/10}\right)\left(\left(10^{F_T/10}-1\right)k_o\right)}{10^{G_T/10}}\left[K\right] \qquad (136)$$

where AT_{tempT} – Antenna noise temperature (K), T_{eT} – Amplifier noise (K), L_T – Cable loss (dB), G_T – Low noise amplifier gain (dB), F_T – Receiver noise figure (dB), k_o = 290 (K). The rain noise $\mathrm{N_{rain}}$ is then calculated as follows

$$N_{\mathrm{rain}} = \left(1-\frac{1}{10^{\mathrm{RA}/10}}\right)k_o\left[K\right] \qquad (137)$$

where RA – Rain absorption (dB), k_o = 290 (K). The total system noise (TS_{noise}) is then calculated as follows:

$$\mathrm{TS_{noise}} = 10\log_{10}\left(\mathrm{RN_{fig}} + S_{temp} + N_{rain}\right)\left[\mathrm{dB}\right] \qquad (138)$$

where RN_{fig} – Receiver noise figure (K) deriving from Eq. (9), S_{temp} – System temperature (K) deriving from Eq. (10), N_{rain} – Rain noise (K) deriving from Eq. (11). The receiving system performance (G/T) is then calculated as follows

$$G/T = \mathrm{Gr} - \mathrm{TS_{noise}}\left[\mathrm{dBK^{-1}}\right] \qquad (139)$$

where Gr - Ground station receiver gain (dB), TS_{noise} – Total system noise (dB). The Equivalent Isotropic Radiated Power (*EIRP*) is calculated as follows:

$$\mathrm{EIRP} = \mathrm{CN}_{\mathrm{ratio}} - G/T + L_{TOTAL} - k\,[\mathrm{dBW}] \tag{140}$$

where CN_{ratio} – Carrier to Noise Ratio (dBHz), *G/T* – Receiving system performance (dBK^{-1}), L_{TOTAL} – Total losses (dB) deriving from Eq.(134), k = 228.6 (dB).

Following the transmitter noise calculations, the antenna design is introduced. Currently the function models the aperture antennas, such as horns and parabolic antennas, which share a common behaviour and a similar expression. The antenna type is chosen on the basis of the required antenna gain, which is a design parameter. It is well know that the best aperture antenna for gains ≤ 20 dB belongs to the horn type set, therefore for gains ≤ 20 dB the horn type is selected and the mass of the antenna is computed as follows.

The antenna characteristic length (m) (it is the diameter of the normal conical section for conical horns, and an equivalent diameter for pyramidal horns) is

$$D_{ant} = \left(\frac{10^{\frac{G_t}{10}}}{n_{eff}}\right)^{0.5} \frac{c}{\pi\, n} \tag{141}$$

where n_{eff} – antenna efficiency (%), $c = 3\cdot 10^8$ (m/s), $n = f\cdot 10^6$ (Hz), then the lateral surface of the horn, S_{LAT} [m^2], is computed as a conical surface

$$S_{LAT} = \pi\,\frac{D_{ant}}{2}\sqrt{\frac{D_{ant}^2}{4} + L_{horn}^2} \tag{142}$$

and the mass, $M_{\mathrm{ant,horn}}$ [kg], is

$$M_{ant,horn} = S_{LAT}\,\rho_A \tag{143}$$

where *L* – the length of the horn antenna, it can be assumed 2 D_{ant} from available data, and ρ_A – the surface density [kg/m^2], which has a mean value of approximately 15 (from available data). In analogous manner, if the gain of the antenna is > 20 dB, the parabola antenna is selected, the diameter of the antenna is computed by the Eq. (141), and, in this case, the mass of the antenna, $\mathrm{M}_{\mathrm{ant,par}}$ [kg], is:

$$M_{ant,par} = \pi\,\frac{D_{ant}^2}{4}\,\rho_A \tag{144}$$

where ρ_A – the surface density [kg/m^2], has a typical value of 10, for parabolas as well. When more types of antenna will be modeled, the optimal ranges of working gains will probably be overlapping. In this case the code will chose the type of antenna on the basis of the mass: the antenna type requiring less mass will be chosen.

Following the antenna design calculations, the required RF link power is calculated. At first, the required power P_{ld} is calculated in dBW as follows

$$P_{\mathrm{Ld}} = \mathrm{EIRP} - \mathrm{G}_t\,[\mathrm{dBW}] \tag{145}$$

where EIRP – Equivalent Isotropic Radiated Power (dBW) deriving from Eq.(140), $\mathrm{G_t}$ is a design parameter that defines the selected antenna gain (dB). The conversion of P_{Ld} to Watts P_L is done as follows:

$$p_{link} = 10^{\left(P_{\mathrm{Ld}}/10\right)}\,[W] \tag{146}$$

where P_{Ld} – Required power (dBW) deriving from Eq.(145).

14.5.2. Amp

[Mamp,M_case,Pamp] = amp(plink,T,CMR)

The Amp function calculates the mass M_{amp}, the power output P_{amp} as well as the mass of the case M_{case} of the Amplifier of the TT&C system, utilizing inputs such as the required power p_{link} internal input deriving from Eq. (146), the type of amplifier T and so on.

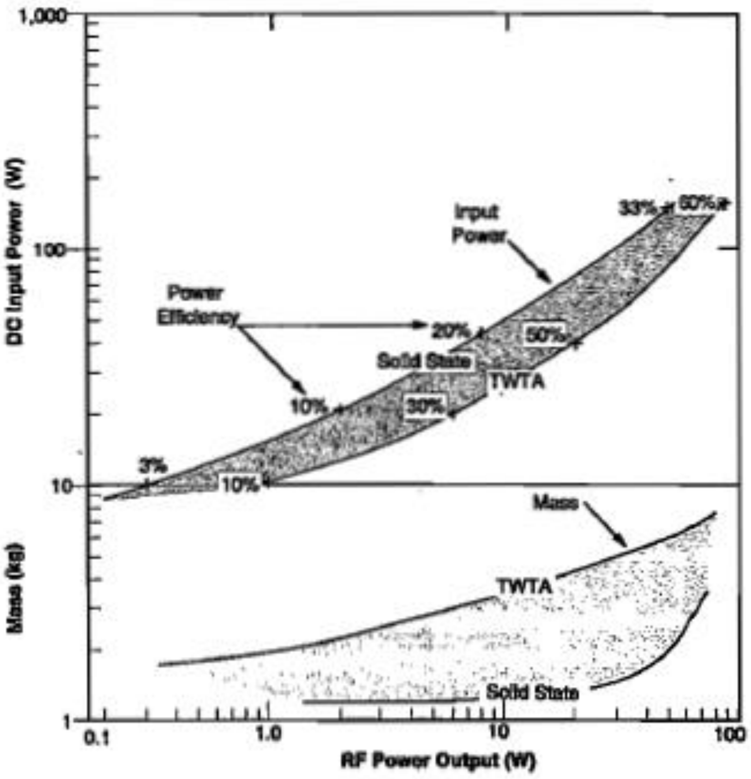


Figure 19: Satellite Transmitter Power and Mass vs RF Power Output

By interpolating the data included in Figure 19, the function derives the M_{amp} and P_{amp} values based on the required power p_{link} as well as the chosen amplifier type (T). With T ∈ [0, 1], it is possible to derive M_{amp} and P_{amp} for any kind of amplifier, whose characteristics are between TWTA type (T = 0) and solid state type (T = 1). Finally, the casing mass M_{case} is computed as a fraction of the amplifier mass:

$$M_{\text{case}} = M_{amp} \cdot CMR \left[\text{kg}\right] \tag{147}$$

where CMR is the ratio between the mass of the case and the amplifier mass.

14.6. Using LISA PathFinder (LPF) as test case for the SpaceART TT&C system model:

The LPF communications subsystem works at X-band frequency, utilizing two hemispherical antennas (for omni directional coverage during LEOP and transfer stages mainly) as well as a medium gain horn antenna for L1 communications.

14.6.1. SpaceART TT&C system model LPF implementation

SpaceART is using the LPF as a test case in order to validate its TT&C system model. Taking into account the mission facts and figures, each input can become a design, uncertain or fixed/environmental parameter depending on their definition respectively:

Design parameters are defined as input parameters which can be decided by the designer(s), after taking into account several mission factors.

Fixed parameters (including environmental parameters) are defined as input parameters which are already known or given, therefore are considered constants.

Uncertain parameters are defined as input parameters which cannot be deterministically defined. Such parameters can either be design or fixed ones which cannot be defined with absolute certainty.

Taking the above definitions into account, the SpaceART TT&C system model inputs are defined as follows:

14.6.2. Link LPF implementation

The Link inputs consist of:

BER – Specified Bit Error Rate → Fixed parameter

Mod – Modulation selection → Design parameter

B – Total amount of data in kilobytes (kb) → C&DH output (internal parameter)

Ta – Ground station access time in seconds (s) → Environmental/Orbital Parameter

r – Distance from ground station in kilometres (km) → Environmental/Orbital Parameter

f – Frequency in Hertz (Hz) → Design parameter

Gr – Ground station receiver gain in decibels (dB) → Environmental Parameter

theta_f – Faraday rotation in degrees (°) → Environmental/Orbital Parameter

Gh – Ground station altitude (km) → Environmental/Orbital Parameter

e – Horizon elevation (°) → Environmental/Orbital Parameter

G – Low noise amplifier gain (Receiver) → Environmental Parameter

L – Cable loss (Receiver) (dB) → Fixed Parameter

G_t – Low noise amplifier gain (Transmitter) (dB) → Fixed Parameter

Lt – Cable loss (Transmitter) (dB) → Fixed Parameter

Te – Amplifier noise (Receiver) (k) → Fixed parameter

F – Receiver noise figure (dB) → Fixed parameter

Tet – Amplifier noise (Transmitter) (k) → Fixed parameter

Ft – Transmitter noise figure (dB) → Fixed parameter

T_ant_t – Antenna noise temperature (k) → Fixed parameter

nt – Antenna efficiency (%)→Uncertain parameter

Gt – Antenna gain (dB) → Design parameter

14.6.3. Amp LPF implementation

The Amp inputs consist of:

T – Amplifier type selection → Design parameter

plink – Required link power in Watts (W) → Link internal output

CMR - Amplifier casing mass ratio → Uncertain parameter

14.6.4. References

Space Mission Analysis and Design 3rd ed. , Wiley J. Larson, James R. Wertz, Microcosm Press & Kluwer Academic Publishers

Antenna Theory Analysis and Design 3rd ed. , Constantine A. Balanis, Wiley-Interscience publications

Satellite Communications 3rd ed. , Dennis Roddy, McGraw-Hill

Tutorial on Basic Link Budget Analysis, intersil ™, Application Note 9804, Jum Zyren, Al Petrick, June 1998

www.atmmicrowave.com/wave-horn.html

http://www2.rfsworld.com/RFS_Edition3/pdfs/Microwave_Solid_Antennas_267-322.pdf

D
Mars science mission data

Figure D.1 is indicative of the dynamic nature of a schedule. In this graphic we see basic science operations of 4 instruments for 4 days during a particular time period (20-24 November 2016). Science and housekeeping operations differ depending on the time period, science requests and orbital attitude required.
In Figures D.2, D.3 we notice the variation in Earth / Sun distance from the spacecraft per time unit, which affects its power consumption (longer distance equals to higher TTC consumption) and generation (solar array output varies with distance and solar radiation angle of incidence) respectively. Similarly, eclipse duration (stored energy must suffice for the eclipse duration plus a safety margin) and Earth occultation (telecommunications not possible, mass storage must suffice at least until next telecommunications window plus safety margin) from the Martian disk.

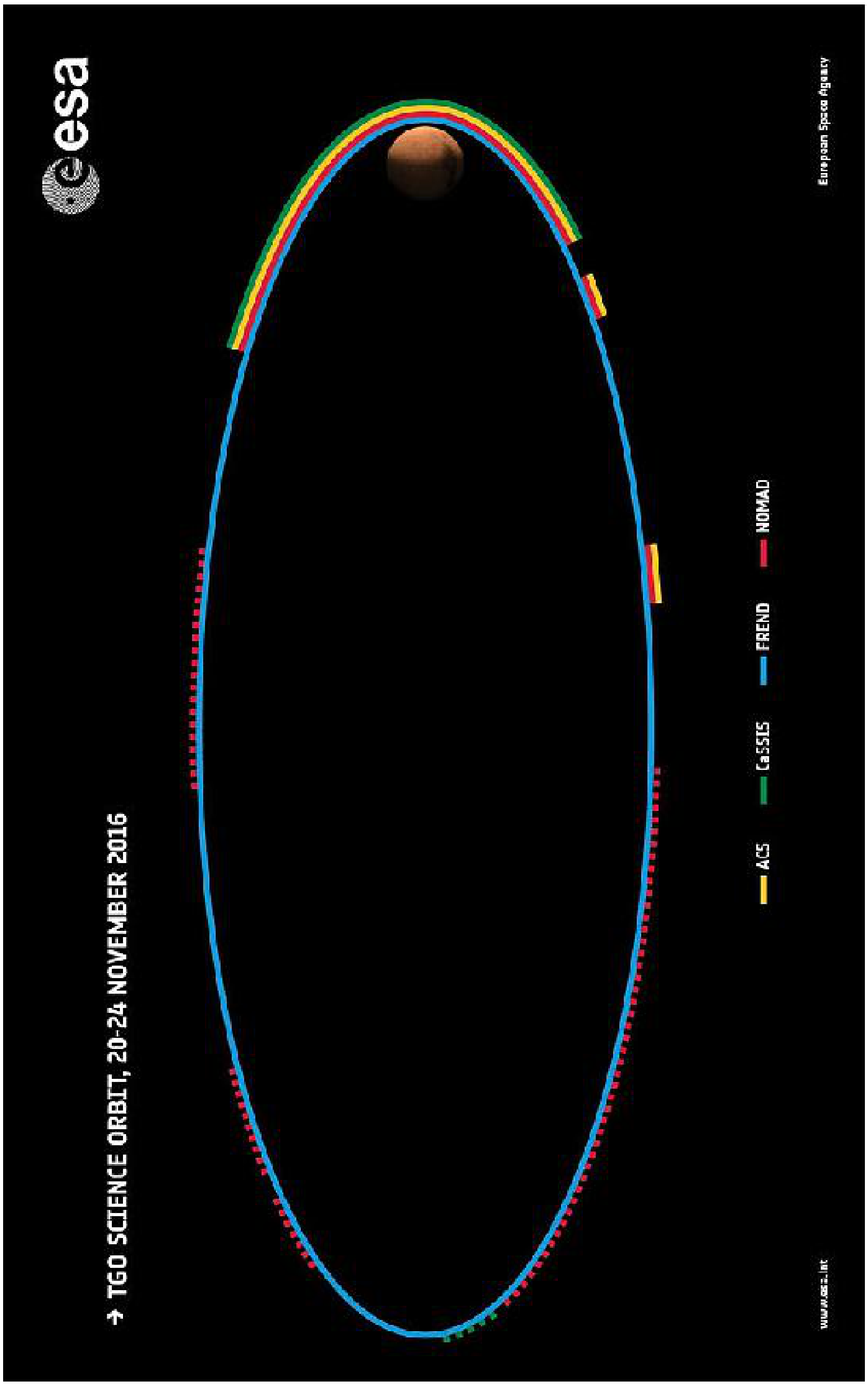


Figure D.1: ExoMars science orbit 1; A graphic summarising when the TGO instruments were operating during the 20 - 24 November orbit (image credit: ESA/Roscosmos/ExoMars/BIRA)

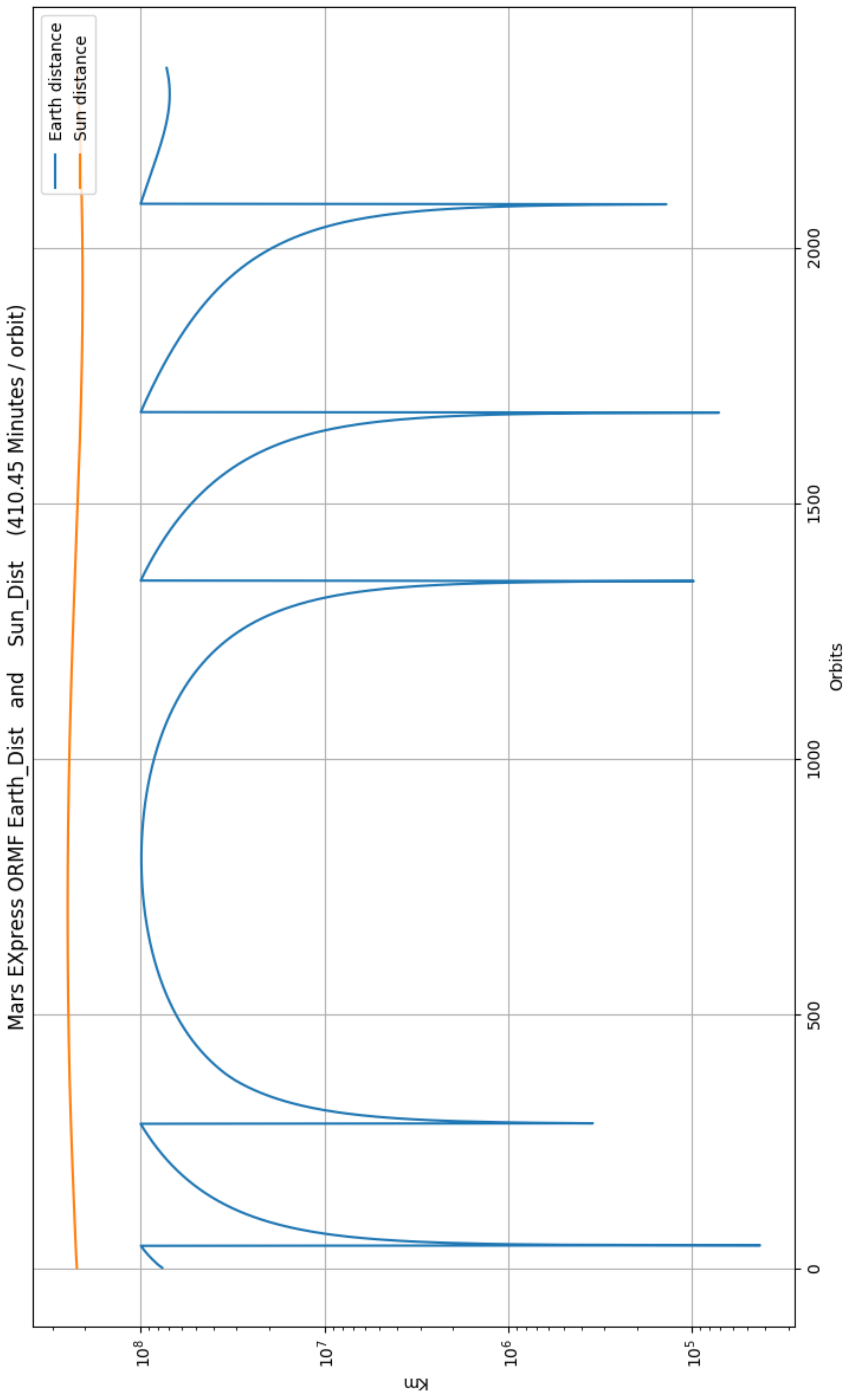


Figure D.2: MEX distance from Sun and Earth per orbit for the span of a month.

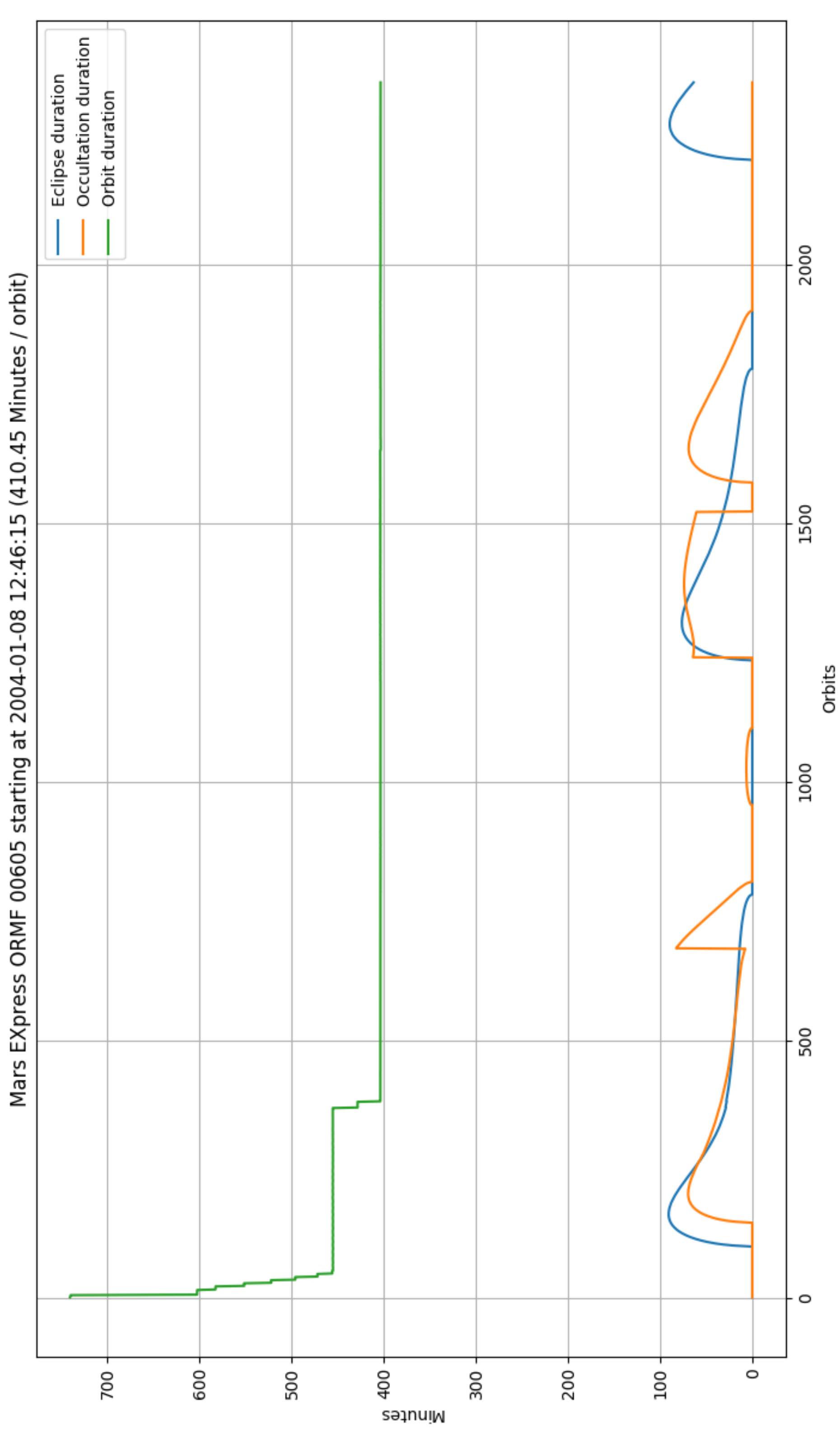


Figure D.3: MEX Solar eclipses and Earth occultations per orbit for a month.

E
PROBA-2 ground station visibility

Notice the variation occurring in two weekly ground station visibilities for different calendar periods. A similar kind of variation occurs on all types of scientific operations for instance topographic multi-spectral imaging, laser altimetry and so on.

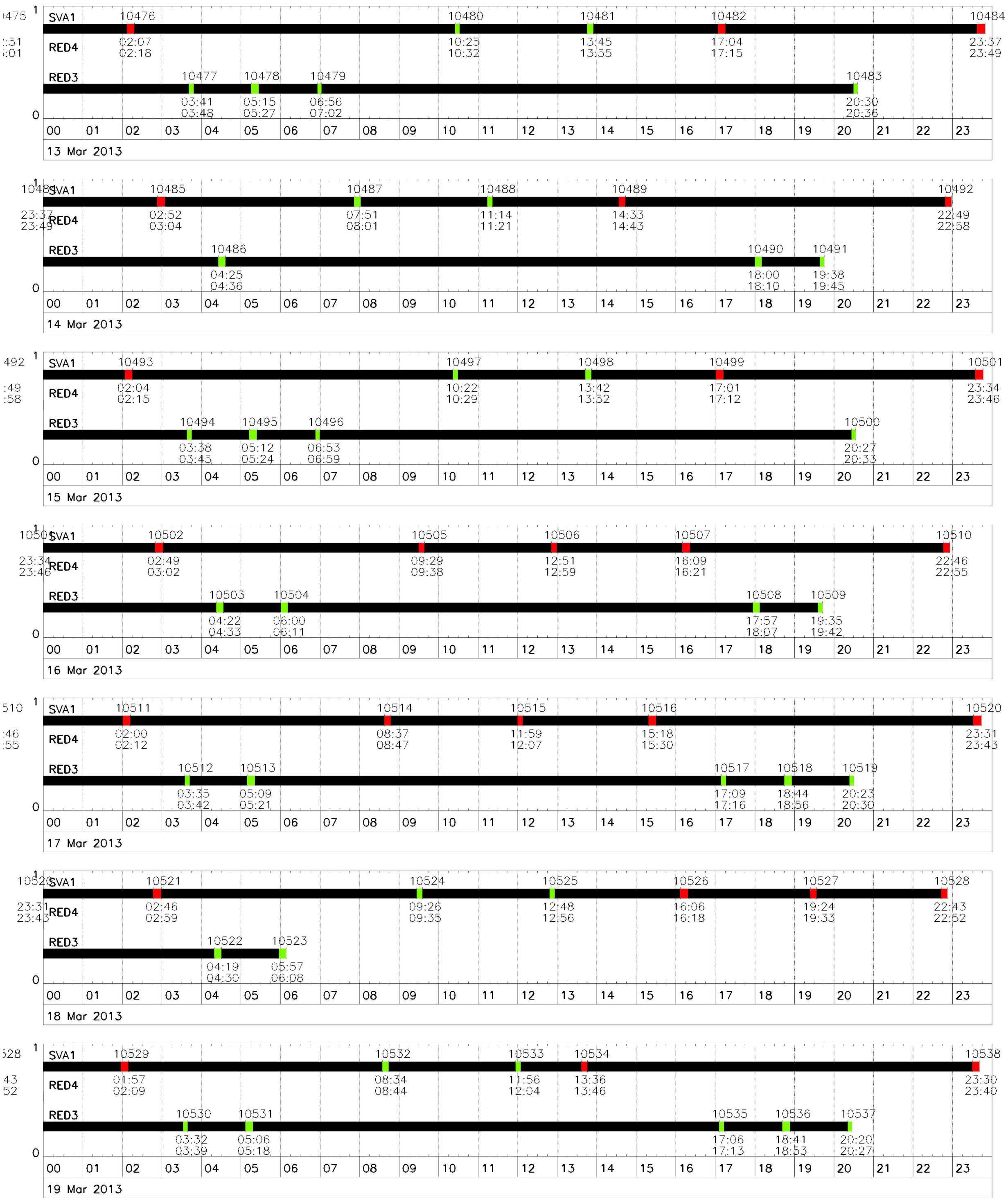


Figure E.1: PROBA-2 ground stations' visibility for the third week of March 2013. [Science Center (P2SC), 2009]

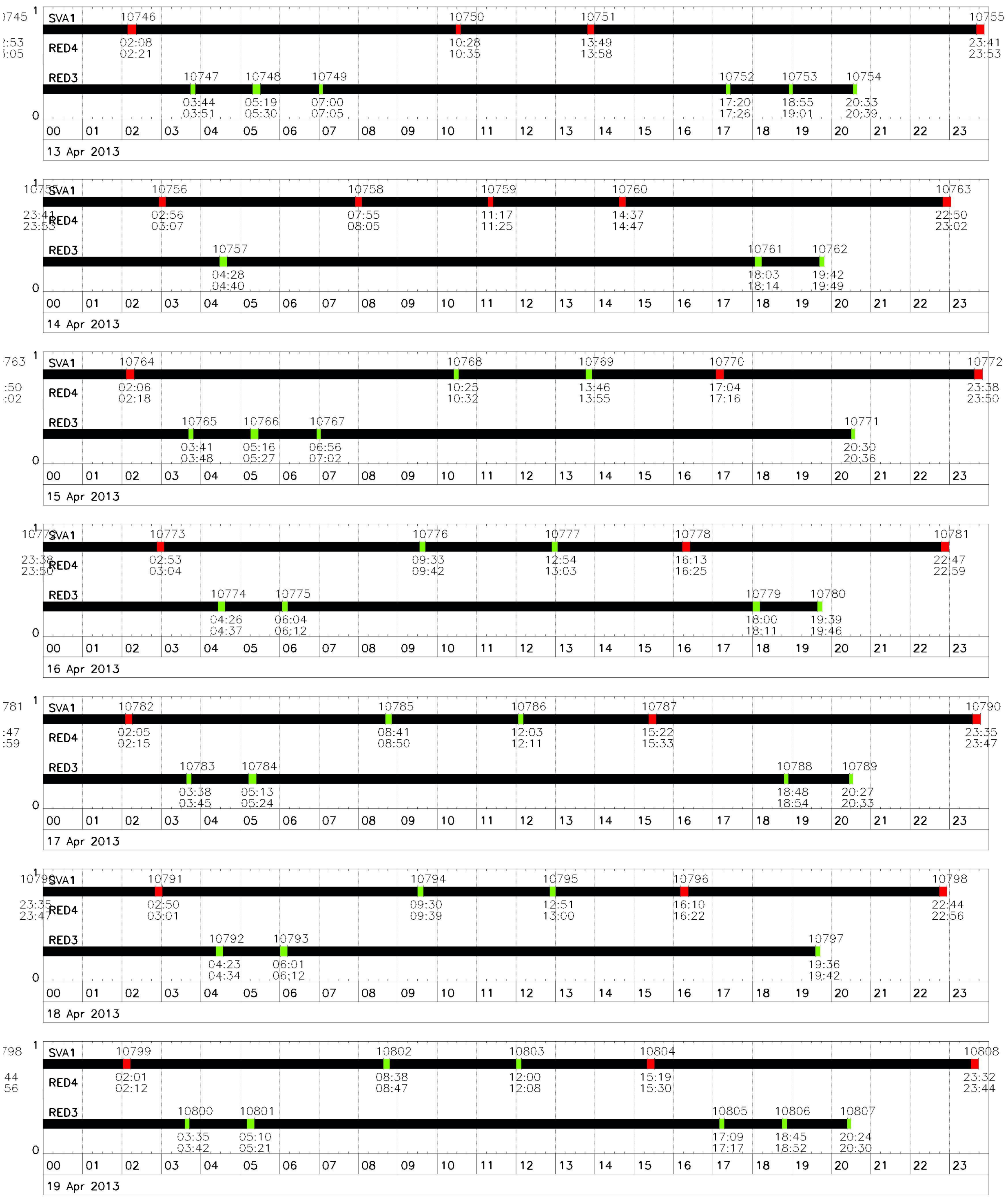


Figure E.2: PROBA-2 ground stations' visibility for the third week of April 2013. [Science Center (P2SC), 2009]

F
Subystem models' setup parameters

Sens.	Star sens. accur. (”)	Sun sens. accur. (°)	Horiz. sens. accur. (°)	Req. attitude accur. (°)
	20	0.03	0.2	0.7
IMU	Gyro drift rate (°/hr)	Accel. linearity (g/g^2)		
	0.009	1.5e-6		
PU	Surface area (m^2)	CoG (m)	Reflectance factor	
	3	0.3	0.3	
Grav.	Mom. of inert. z-y axis (kgm^2)	Resid. vehicle dipole (Am^2)		
	30	1		
Grav.	Solar flux at 1AU (W/m^2)	Angle of incidence (°)		
	1367	0		
Aero.	Drag coef.	CofAero.Press. - CoG (m)		
	2	0.2		
Aero.	Atm. density (kg/m^3)	Spacecraft velocity (m/s)		
	1e-13	V		
Sol.	Z-axis dev. from loc. | (rad)	Orbital radius (m)	Planet magn. field (T)	
	1	R	$2 \cdot 7.96e15/7078e3^3$	

Figure F.1: ADS setup [Wertz and Larson, 1999]

PU	Max. MIPS	Mass storage (Gbits)	
	240	12	
Data	Bus datarate (kbits/sec)	Data buses	Conductor spec. mass (g/m)
	500	2	2.01

Figure F.2: CDH setup [Griffin and French, 2004; Brown, 2002; States Department of Defense, 2018; Coordinating Committees, 1992; Soltero et al., 2010; Henehan and Johas-Teener, 2006]

Env.	Orb. daylight time (h)	Orb. eclipse time (h)	Min. angle of inc. (°)	
	t_d	t_e	23.5	
Array	Bus voltage (V)	Solar flux at 1AU (W/m^2)	Day power req. (W)	Ecl. power req. (W)
	28	1367	P_d	P_e
Array	Life (yr)	Solar cell effic. (%)	Inherent degradation (%)	Array spec. mass (kg/m^2)
	5	28.2	77	1.5
PCU	Energy transm. eclipse (%)	Energy transm. day (%)	Conversion effic. (%)	
	0.65	0.85	95	
Batt.	Specific energy dens. (Wh/kg)	Max. bus voltage drop (%)	No. of battery cycles	
	70	5	1500	

Figure F.3: EPS setup [Patel, 2004; Fortescue et al., 2011; Hyder, 2000; Griffin and French, 2004; Wertz and Larson, 1999; McKissock et al., 2009; Khrypunova et al., 2006]

Env.	Faraday rotation (°)	Receiver Gain (dB)	Transm. path (km)	Access time (s)
	9	30	R	t_{viz}
Env.	Total data (kb)	Bit Error Rate	GS altitude (km	Horiz. elev. (°)
	500e3	10e-6	GS_{alt}	5
Rx	Freq. (GHz)	Low noise amp. gain (dB)	Cable loss (dB)	
	2.45	50	5	
Rx	Amp. noise (k)	Noise figure (dB)		
	70	12		
Tx	Low noise amplifier gain (dB)	Cable loss (dB)	Antenna Type	
	30	3	Parabolic	
Tx	Modulation	Amplifier	Noise temp. (k)	
	0.5	0.5	30	
Ant.	Noise temp. (k)	Efficiency (%)	Diameter (m)	
	60	60	1.5	

Figure F.4: TTC setup [Wertz and Larson, 1999; Balanis, 2005; Roddy, 2006]

G Statistical analysis

G.1 Choosing the Correct Statistical Test

The UCLA Institute for Digital Research and Education (IDRE) has published general guidelines for choosing a statistical analysis [Leeper, 2011]. It is a helpful aid in choosing which analysis can be used, depending on the experiment at hand.

Table G.1: Choosing the Correct Statistical Test

No. of Indep. Var.	Nature of IVs	Nature of Dep. Var.	Test(s)
1	0 IVs (1 population)	interval & normal	one-sample t-test
		ordinal or interval	one-sample median
		categorical (2 categories)	binomial test
		categorical	Chi-square goodness-of-fit
	1 IV with 2 levels (indep. groups)	interval & normal	2 indep. sample t-test
		ordinal or interval	Wilcoxon-Mann Whitney test
		categorical	Chi-square test
			Fishers exact test
	1 IV with ≥ 2 levels (indep. groups)	interval & normal	one-way ANOVA
		ordinal or interval	Kruskal Wallis
		categorical	Chi-square test
	1 IV with 2 levels (dep./matched groups)	interval & normal	paired t-test
		ordinal or interval	Wilcoxon signed ranks test
		categorical	McNemar
	1 IV with ≥ 2 levels (dep./matched groups)	interval & normal	one-way repeated measures ANOVA
		ordinal or interval	Friedman test
		categorical (2 categories)	repeated measures logistic regression
	≥ 2 IVs (indep. groups)	interval & normal	factorial ANOVA
		ordinal or interval	ordered logistic regression
		categorical (2 categories)	factorial logistic regression
	1 interval IV	interval & normal	correlation
		interval & normal	simple linear regression
		ordinal or interval	non-parametric correlation
		categorical	simple logistic regression
	≥ 1 interval IVs and/or ≥ 1 categorical IVs	interval & normal	multiple regression
			analysis of covariance
		categorical	multiple logistic regression
			discriminant analysis
2+	1 IV with ≥ 2 levels (indep. groups)	interval & normal	one-way MANOVA
	2+	interval & normal	multivariate multiple linear regression
	0	interval & normal	factor analysis
2 sets of 2+	0	interval & normal	canonical correlation

G.2 Post-hoc analysis

G.2.1 Chapter 4

Using the Friedman test, we inferred that in the iteration-constrained runs and the cases run for 8 hours, at least one algorithm has a different median from the rest. We therefore rejected the null hypothesis H_0 that all groups come from the same distribution, sharing the same median. To find which algorithm(s) are significantly different in fitness for the iteration-constrained case, we present the corresponding post-hoc analysis below. The time-constrained case comprises a small sample size, thus we infer results by observation of boxplots in Figure 4.24.

Table G.2: Nemenyi post-hoc test (pt. 1) of iteration-constrained runs. Significant results in **bold**

Algorithm	ACS	ACS +LS	GA	GA +LS	sGA	sGA +LS	RAND	RAND +LS
ACS +LS	1.00000	-	-	-	-	-	-	-
GA	**0.00380**	**0.00812**	-	-	-	-	-	-
GA +LS	**0.00073**	**0.00170**	1.00000	-	-	-	-	-
sGA	**1.4e-10**	**5.4e-10**	0.17608	0.39512	-	-	-	-
sGA +LS	**3.2e-10**	**1.2e-09**	0.23165	0.47975	1.00000	-	-	-
RAND	**6.3e-05**	**2.3e-05**	**2.0e-13**	**1.3e-13**	**< 2e-16**	**< 2e-16**	-	-
RAND +LS	**0.02002**	**0.00995**	**7.0e-13**	**2.1e-13**	**1.4e-13**	**1.5e-13**	0.99553	-
SA	0.91838	0.96753	0.59994	0.32404	**1.3e-05**	**2.3e-05**	**1.2e-09**	**4.7e-06**
SA +LS	0.77199	0.87608	0.80428	0.53566	**6.0e-05**	**0.00011**	**1.5e-10**	**8.5e-07**
sSA	**9.9e-12**	**4.1e-11**	0.06547	0.18540	1.00000	1.00000	**< 2e-16**	**4.6e-14**
sSA +LS	**9.9e-12**	**4.1e-11**	0.06547	0.18540	1.00000	1.00000	**< 2e-16**	**4.6e-14**
DE	0.61589	0.47186	**1.5e-08**	**1.1e-09**	**1.4e-13**	**1.6e-13**	0.33084	0.99210
DE +LS	0.88546	0.78519	**2.3e-07**	**2.0e-08**	**1.1e-13**	**1.2e-13**	0.12015	0.91083
GREEDY	**9.0e-10**	**2.4e-10**	**1.4e-13**	**< 2e-16**	**< 2e-16**	**< 2e-16**	0.90283	0.11343
GREEDY +LS	**3.7e-06**	**1.2e-06**	**1.1e-13**	**1.9e-13**	**< 2e-16**	**< 2e-16**	1.00000	0.90283

Table G.3: Nemenyi post-hoc test (pt. 2) of iteration-constrained runs. Significant results in **bold**

Algorithm	SA	SA +LS	sSA	sSA +LS	DE	DE +LS	GREEDY
ACS +LS	-	-	-	-	-	-	-
GA	-	-	-	-	-	-	-
GA +LS	-	-	-	-	-	-	-
sGA	-	-	-	-	-	-	-
sGA +LS	-	-	-	-	-	-	-
RAND	-	-	-	-	-	-	-
RAND +LS	-	-	-	-	-	-	-
SA	-	-	-	-	-	-	-
SA +LS	1.00000	-	-	-	-	-	-
sSA	**1.7e-06**	**8.9e-06**	-	-	-	-	-
sSA +LS	**1.7e-06**	**8.9e-06**	1.00000	-	-	-	-
DE	**0.00415**	**0.00118**	**1.7e-13**	**1.7e-13**	-	-	-
DE +LS	**0.02157**	**0.00718**	**1.6e-13**	**1.6e-13**	1.00000	-	-
GREEDY	**1.4e-13**	**1.1e-13**	**< 2e-16**	**< 2e-16**	**0.00063**	**7.8e-05**	-
GREEDY +LS	**2.8e-11**	**3.0e-12**	**< 2e-16**	**< 2e-16**	**0.09222**	**0.02323**	0.99553