

CRANFIELD UNIVERSITY

MARIA CARRILLO BARRENECHEA

STOCHASTIC OPTIMISATION FOR COMPLEX MIXED-INTEGER
PROGRAMMING PROBLEMS IN ASTEROID TOUR MISSIONS

SCHOOL OF AEROSPACE, TRANSPORT AND MANAGEMENT
Astronautics and Space Engineering

MSc

Academic Year: 2020 - 2021

Supervisor: Dr Joan-Pau Sánchez Cuartielles
Associate Supervisor: Dr Javier Del Ser Lorente
September 2021

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANAGEMENT
Astronautics and Space Engineering

MSc

Academic Year 2020 - 2021

MARIA CARRILLO BARRENECHEA

Stochastic Optimisation for Complex Mixed-Integer Programming
Problems in Asteroid Tour Missions

Supervisor: Joan-Pau Sánchez Cuartielles
Associate Supervisor: Javier Del Ser Lorente
September 2021

This thesis is submitted in partial fulfilment of the requirements for
the degree of Astronautics and Space Engineering MSc

© Cranfield University 2021. All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

ABSTRACT

Deep space exploration is key to understand the origin of our Solar System and address the Earth impact risk. Space Trajectory Design (STD) has evolved and incremented in complexity due to the interest within the space community to explore multiple celestial bodies in a single mission. This thesis focuses on an Asteroid Tour Trajectory in the context of the CASTAway mission. CASTAway is a mission proposal for European Space Agency's 5th call of medium-size missions to explore the Asteroid Main Belt.

The objective is not to find the global optima but find feasible sequences of asteroid fly-bys, as per feasible tours of 12 asteroids of a total Δv of less than 9 km/s is meant. The complexity of the problem is given by the large number of possible permutations of 12-asteroid tour solutions – even with a reduced catalogue of 158 asteroids – and because of being a Mixed-Integer Non-Linear Programming (MINLP) problem. Because of this, metaheuristics are used to tackle the problem. A novel problem modelling that achieves uniqueness on the cost paths of the Search Space and a novel ACO solver is presented, with the general objective for the whole CASTPath project of finding a robust low computational heuristic. Due to the scientific interest on having diversity in the sequences, a similarity measurement tool is also developed.

Several test cases with different ACO tuning parameters are run on a High Performance Computer. Results show that this algorithm outperforms the previous heuristics on CASTPath obtaining the lowest Δv (7.27 km/s) achieved by an heuristic and finding multiple feasible sequences (97 in 1 h). Moreover, the new problem modelling has allowed within the research group, to find the global optima (6.98 km/s) for this asteroid catalogue by Dynamic Programming.

Keywords:

Space Trajectory Design, Ant Colony Optimisation, ACO, Multiple Fly-By Trajectory, Lambert Arc, Metaheuristics, Heuristics, CASTAway, CASTPath, ESA M-class Mission, Main Asteroid Belt

ACKNOWLEDGEMENTS

I would like to thank my supervisor Joan Pau for his guidance and feedback during the development of this project. It is a pleasure to work in such an organised and nice work environment. His and Andrea Bellome's support and assistance has been key for the completion of the project. Thank you Andrea for your support when the results were not as good!

As an old mentor from the previous studies, it has been a pleasure to have Javier Del Ser as associate supervisor. His knowledge on AI was the final piece of the puzzle to have the best team besides me I could have ever think about. It is always a pleasure to work with you Javi, eskerrik asko!

This marks the end of my studies (official studies, you never stop learning) and I could not fell prouder and luckier for the support I have always had from my family, friends and partner; for the formed friendships and lived experiences; and for the projects I have participated in which helped to discover my two passions: AI and Space.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS.....	iii
LIST OF FIGURES.....	vi
LIST OF TABLES	vii
LIST OF EQUATIONS.....	viii
LIST OF ABBREVIATIONS	ix
1 Introduction.....	1
1.1 Space Trajectory Design.....	1
1.2 CASTAway asteroid tour mission	1
1.3 Optimisation and Metaheuristic techniques	2
1.4 Objectives and Scope of the Thesis.....	2
1.5 Structure of the Thesis.....	3
2 Multi-body Space Trajectory Design.....	5
2.1 Asteroid Tour Trajectory Design	5
2.1.1 Travelling Salesman Problem (TSP)	5
2.1.2 Mixed-Integer Non-Linear Programming (MINLP) Problems.....	7
2.2 Optimisation techniques on Asteroid Tour Trajectory Design	9
2.2.1 Deterministic and stochastic solvers	9
2.2.2 Metaheuristic algorithms	10
2.3 Previous work on CASTPath	12
3 Problem Statement.....	14
3.1 CASTAway asteroid tour optimisation problem.....	14
3.2 Problem Baseline.....	14
3.2.1 Filtering by MOID and baseline Earth-Mars leg.....	16
3.2.2 Objective function: Δv	17
3.3 Combinatorial Problem	18
3.4 Drivers and constraints	19
4 Proposed Approach.....	20
4.1 Search Space	20
4.2 Score Matrix.....	23
4.3 Cleaning of Search Space	24
4.4 Similarity Measure	28
4.5 Ant Colony Optimisation metaheuristic	29
4.5.1 ACO algorithm definition	29
4.5.2 Implemented ACO algorithm	32
4.5.3 Multiobjective formulation of ACO	39
5 Simulations.....	42
5.1 Approaches set-up.....	42
5.1.1 Execution time and number of runs or rounds.....	42

5.1.2 Score Matrix cleaning threshold	42
5.1.3 ACO parameters	43
5.2 Test cases	44
5.3 Analysis tools.....	45
5.3.1 Box and Whiskers Plot	45
5.3.2 Stacked Bar Graph.....	46
5.3.3 Similarity analysis.....	47
5.4 Computational tools	47
6 Results	48
6.1 Score Matrix validation test.....	48
6.2 Whole Search	49
6.3 Score Matrix filtered at 9 km/s	55
6.4 Score Matrix filtered at 1 km/s	60
6.4.1 Comparison of $\beta = 4$ and $\beta = 5$	60
6.4.2 Removing the pheromone contribution, $\alpha = 0$	65
6.4.3 Comparison of $l = 54$ and $l = 611$	66
7 Discussion.....	67
7.1 Comparison of the different set-ups.....	67
7.1.1 Index reward	67
7.1.2 Smoothness of the Search Space.....	67
7.1.3 Trail dependency of the nodes in the Search Grid	68
7.2 Comparison with previous work	69
7.2.1 Use of Score Matrix.....	69
7.2.2 ACO Tabu Search.....	70
7.2.3 Dynamic Programming.....	71
8 Conclusions and Future Work	73
8.1 Conclusions	73
8.2 Future Work.....	74
REFERENCES.....	76
APPENDICES	82

LIST OF FIGURES

Figure 2-1. TSP: Cities or <i>nodes</i> to visit (in blue) in the Search Space limited by the US region (in red) [10]	6
Figure 2-2. TSP: Lowest cost tour [10]	7
Figure 2-3. Classification of Optimisation algorithms, Metaheuristics.....	11
Figure 3-1. 2D sketch of the s/c trajectory and Δv between asteroid j and k	17
Figure 4-1. Comparison of a leg Δv changing prior asteroid in a 2D sketch (example with not real asteroid indices).....	21
Figure 4-2. Schematic diagram of the Search Grid modelled by subsets and pairs of nodes, an example of a chosen path	22
Figure 4-3. Similarity Measure example	28
Figure 6-1. Best Δv achieved so far in each iteration in the Whole Search for 10 mins, 30 mins and 60 mins, for $\beta = 1$ and $\beta = 5$	50
Figure 6-2. Boxplot of results of ACO for Whole Search and $\beta = 1$	51
Figure 6-3. Boxplot of results of ACO for Whole Search and $\beta = 5$	52
Figure 6-4. Stacked Bar Graph of results of ACO for Whole Search, $\beta = 1$ (left) and $\beta = 5$ (right)	53
Figure 6-5. Best Δv achieved so far in each iteration of ACO with $SM \leq 9$ km/s for 10 mins, 30 mins and 60 mins, for $\iota = 54$ and $\iota = 611$	56
Figure 6-6. Boxplot of results of ACO with $SM \leq 9$ km/s and $\iota = 54$	57
Figure 6-7. Boxplot of results of ACO with $SM \leq 9$ km/s and $\iota = 611$	57
Figure 6-8. Stacked Bar Graph of results of ACO with $SM \leq 9$ km/s and $\iota = 54$ (left) and $\iota = 611$ (right). Note that the y axis is in logarithmic scale.....	58
Figure 6-9. Boxplot of results of ACO with $SM \leq 1$ km/s and $\beta = 4$	62
Figure 6-10. Boxplot of results of ACO with $SM \leq 1$ km/s and $\beta = 5$	62
Figure 6-11. Stacked Bar Graph of results of ACO with $SM \leq 1$ km/s and $\beta = 4$ (left) and $\beta = 5$ (right).....	63
Figure 6-12. Boxplot of results of ACO with $SM \leq 1$ km/s and $\alpha = 0$. Note that only 2 solutions exist for 30 min and 4 solutions for 60 min.....	65
Figure 6-13. Boxplot of results of ACO with $SM \leq 1$ km/s, $\beta = 5$ and $\iota = 611$	66
Figure 7-1. ACO Tabu Search results boxplot. Plot made from the data results of Tena [33]	71

LIST OF TABLES

Table 3-1. Baseline Earth-Mars leg from [33].....	16
Table 4-1. Score Matrix (SM)	24
Table 4-2. Score Matrix (SM) Cleaning	27
Table 4-3. Next possible nodes selection from SM. Example for pair (A_{23}, A_{56})	38
Table 6-1. Test to prove the cost uniqueness with the implementation of the SM and comparison on computational time for using the SM and the wrapper function tool	48
Table 6-2. Parameters and results of ACO solver for Whole Search and $\beta = 1$	54
Table 6-3. Parameters and results of ACO solver for Whole Search and $\beta = 5$	54
Table 6-4. Parameters and results of ACO solver with $SM \leq 9$ km/s and $\iota = 54$	59
Table 6-5. Parameters and results of ACO solver with $SM \leq 9$ km/s and $\iota = 61$	59
Table 6-6. Parameters and results of ACO solver with $SM \leq 1$ km/s and $\beta = 4$	64
Table 6-7. Parameters and results of ACO solver with $SM \leq 1$ km/s and $\beta = 5$. Best solution encountered by heuristics in CASTPath.....	64

LIST OF EQUATIONS

(2-1).....	8
(3-1).....	15
(3-2).....	17
(3-3).....	17
(3-4).....	18
(4-1).....	26
(4-2).....	26
(4-3).....	26
(4-4).....	28
(4-5).....	29
(4-6).....	31
(4-7).....	31
(4-8).....	31
(4-9).....	35
(4-10).....	35
(4-11).....	35
(4-12).....	35
(4-13).....	37
(4-14).....	37
(4-15).....	38
(4-16).....	38
(4-17).....	39
(4-18).....	40
(4-19).....	40
(4-20).....	40

LIST OF ABBREVIATIONS

ACO	Ant Colony Optimisation
AMB	Asteroid Main Belt
AU	Astronomical Unit
BB	Branch and Bound
CASTAway	Comet and Asteroid Space Telescope Away
CASTPath	Comet and Asteroid Space Telescope Path
CRO	Coral Reefs Optimisation
CS	Cuckoo Search
ESA	European Space Agency
FA	Firefly Algorithm
GA	Genetic Algorithm
HOPC	Hybrid Optimisation Control Problems
IQR	Interquartile Range
MINLP	Mixed-Integer Non-Linear Programming
MOID	Minimum Orbit Intersection Distance
NSGA	Non-dominated Sorting Genetic Algorithm
PSO	Particle Swarm Optimisation
REQ	Requirement
S/C	Spacecraft
SQP	Sequential Quadratic Programming
STD	Space Trajectory Design
ToF	Time of Flight
TSP	Travel Salesman Problem

1 INTRODUCTION

1.1 Space Trajectory Design

In the recent years, there has been a considerable growth on the research and development of missions for the deep space exploration. These missions are of great interest for the scientific community to understand the origin of our Solar System, essential for the human survival in the sense of analysing Earth asteroid impact risk, and necessary for future interplanetary travels.

Space Trajectory Design (STD) has evolved and incremented in complexity due to the interest within the space community to explore multiple celestial bodies in a single mission. A mission with a single target is normally optimised by minimising the mass of propellant and/or the Time of Flight (ToF). A multiple-body tour is more challenging considering the dynamics of the celestial bodies, and variability depending on the launch dates and transfer times.

Some examples of the different types of problems that exist within the STD field are Active Debris Removal (ADR), targeting various dead satellites to capture in the trajectory; Multiple-gravity Assist (MGA) that includes several swing-bys of planets to gain energy towards the goal destination; and asteroid exploration. This thesis focuses on this last STD type problem in the context of the CASTAway mission.

1.2 CASTAway asteroid tour mission

CASTAway is a mission proposal for European Space Agency's (ESA) 5th call of M-class, medium-size, missions (M5) to explore Solar System's Asteroid Main Belt (AMB) [1]. CASTAway stands for Comet and Asteroid Space Telescope - Away [in the AMB], which is a small telescope to be launched by 2029+. The telescope maps variations in composition and size distribution of several thousands of point source objects (at long-range scale), while performing a minimum of ten asteroid flybys (doubling the number of asteroids that a single mission has been capable to fly by) to detect small celestial objects

(~10 m). This mission aims to provide comprehensive data on the AMB for the study of Solar System evolution theories [1], [2].

The CASTPath project was born under the CASTAway feasibility study for the mission trajectory design. Sánchez et al. [2] successfully demonstrated the feasibility of the mission, but noted that further improvements could be done in the trajectory design optimisation.

1.3 Optimisation and Metaheuristic techniques

The CASTAway trajectory design is a combinatorial problem, where the best sequence of asteroids from a given number of combinations is sought. Considering the large amount of asteroids within the AMB, the number of possible combinations is extremely large to be tackled in an analytical conservative way. This is why the use of heuristic and even more, metaheuristic algorithms to solve STD projects is spread even though much research needs to be done on this field.

The choice of the optimisation problem solver depends on the problem nature itself. Metaheuristic techniques and analytical solvers should be compared for a problem in particular. Apart from that, one can have an idea of which solvers will perform better than others, but it is always necessary to perform some analysis and comparison among the solvers and within one solver itself among different tuning parameters. This is the reason why the CASTPath project aims to compare different heuristics and analytical solvers for the STD [3].

1.4 Objectives and Scope of the Thesis

This thesis focuses on the development of a novel solver with the general objective for the whole CASTPath project of finding a robust low computational cost CASTAway trajectory design solver. Following the reasoning presented in 1.3. Optimisation and Metaheuristic techniques, the thesis will also analyse and compare different metaheuristic algorithms and approaches of previous work in CASTPath to solve the problem. The objective is not to find the global optima

but find feasible sequences of asteroids. As per feasible sequences of 12 asteroids of a total Δv of less than 9 km/s is meant.

A filtered database of 158 asteroids of Minimum Orbital Intersection Distance $MOID \leq 0.05$ AU will be used for all the simulations, which was created by previous researchers in CASTPath project. The Ant Colony Optimisation metaheuristic will be used following the best results on CASTPath so far. Thus, the scope of the thesis comprises the following specific objectives:

- A new more exact and unique problem formulation that avoids the splitting of the problem into an approximated search and an optimal refinement, reducing the computational time of the solver
- The development of a robust solver for CASTPath
- A novel implementation of the Ant Colony Optimisation metaheuristic algorithm on CASTPath
- The development of a comparison tool to measure the diversity of the solutions encountered by each solver in CASTPath
- The comparison of heuristic and deterministic algorithms for CASTPath

1.5 Structure of the Thesis

This thesis is structured as follows: Sections 2 and 3 present a state-of-the-art review on first, multi-body trajectory design (2.1) and optimisation and metaheuristic techniques (2.2), and second, on previous work on the trajectory design CASTPath (2.3) and CASTAway problem (3.1). Section 3 also includes the problem baseline (3.2), drivers and constraints (0) derived from the state-of-the-art of CASTPath and a justification on the use of metaheuristics to solve the problem (3.3). The ACO standard definition is given later in the document (4.5.1) along with the proposed ACO approach (4.5.2, 4.5.3) for the ease of read.

Section 4 presents the proposed approach for the modelling of the Search Space (4.1), the Score Matrix definition (4.2) and pruning (4.3), and a definition

of a Similarity Measure score (4.4). Section 4 ends with the Ant Colony Optimisation algorithm used to solve the CASTPath problem (4.5).

Section 5 presents the simulations approach. The different cases parameters set-up is discussed and justified in 5.1. The different test cases whose results will be shown in section 5 are summarised in section 5.2. Before the results, the different analysis tools are defined in 5.3, and the computational tools used are specified in 5.4.

Section 6 presents the results for all the test cases and an individual analysis is done per test case. Section 7 gives a further discussion on first, a global analysis of the results (7.1) and second, a comparison with previous work in CASTPath (7.2). The thesis ends with conclusions and future work in section 8.

Finally, appendices include a table with the definition and value of the variables used within all the report (Appendix A8.2Appendix A) and the ACO backtrack algorithm diagram (Appendix B).

2 MULTI-BODY SPACE TRAJECTORY DESIGN

This section aims on analysing the type of problem in hand based on the state-of-the-art on multi-body Space Trajectory Design, to continue with a review on the possible optimisation techniques found on the literature to tackle this type of problems. The section finishes with a summary on the different solvers used on previous work on the CASTPath project.

2.1 Asteroid Tour Trajectory Design

Up to date, a total of 14 asteroids have been successfully visited in a total of 10 missions [4]. Several propositions on asteroid tours have been submitted in the Global Trajectory Optimization Competition (GTOC) [5] as [6], [7]; other examples of asteroid tour mission design can be found in [1] (for the CASTAway trajectory design) or [8], [9]. An important consideration for this problem is that an *asteroid tour*, referring to a space trajectory that flies by not one but multiple celestial bodies, implies a high level of optimisation complexity. This section analyses the two types of problem an asteroid tour trajectory design is: a Combinatorial Problem (CP) and a Mixed-Integer Non-Linear Programming (MINLP) problem.

2.1.1 Travelling Salesman Problem (TSP)

Combinatorial Problems are often modelled with a *Search Space* which is a grid of connected *nodes*. A very common example of CP is the Travelling Salesman Problem (TSP) that is about a salesperson that has to visit a given number of cities, which will be the nodes, that are connected by roads or paths of a given length. For a better illustration of the problem, the MATLAB TSP example code available in [10]. Figure 2-1 shows the cities to be visited as blue dots within the Search Space (US in this example, limited in red) and the starting point or hometown of the salesperson in yellow.

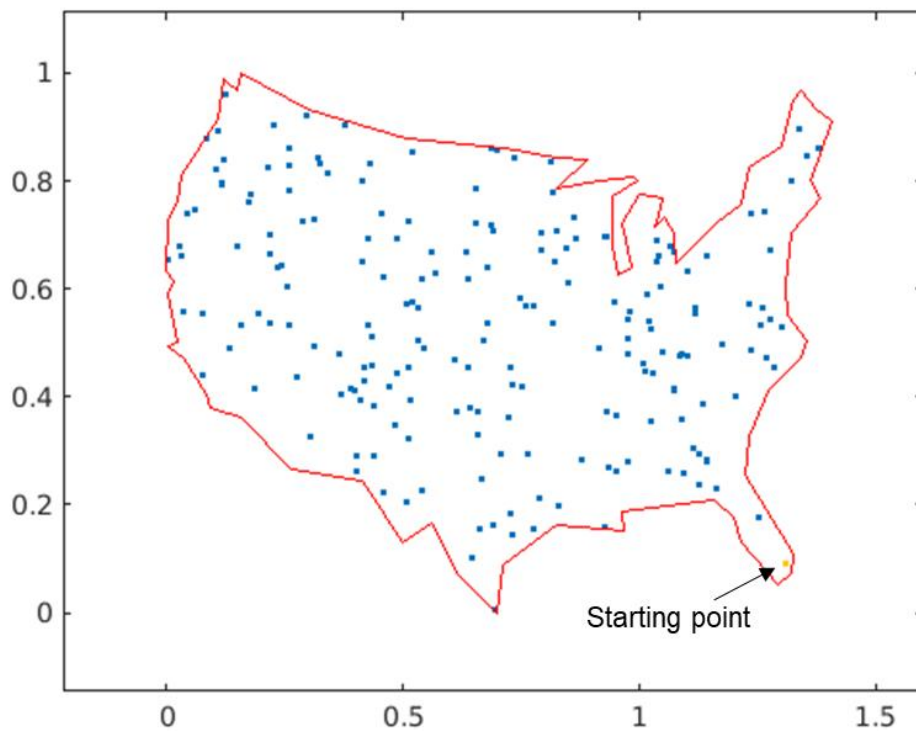


Figure 2-1. TSP: Cities or *nodes* to visit (in blue) in the Search Space limited by the US region (in red) [10]

The length of each path can be related to the fuel consumption of the salesperson's car, or the time to travel, which will be the cost of the path. The salesperson has to start from his hometown, visit each city once and return back home. As an optimisation problem, the cheapest (in fuel consumption) or shortest (in time of travel) path or *tour* wants to be found.

The high complexity of the problem can be appreciated when one starts listing all the possible combinations of paths to find the cheapest tour. Figure 2-2 shows the solution to the problem. There exist many techniques to solve the TSP, a description of the optimisation techniques is given in 2.2. Optimisation techniques on Asteroid Tour Trajectory Design.

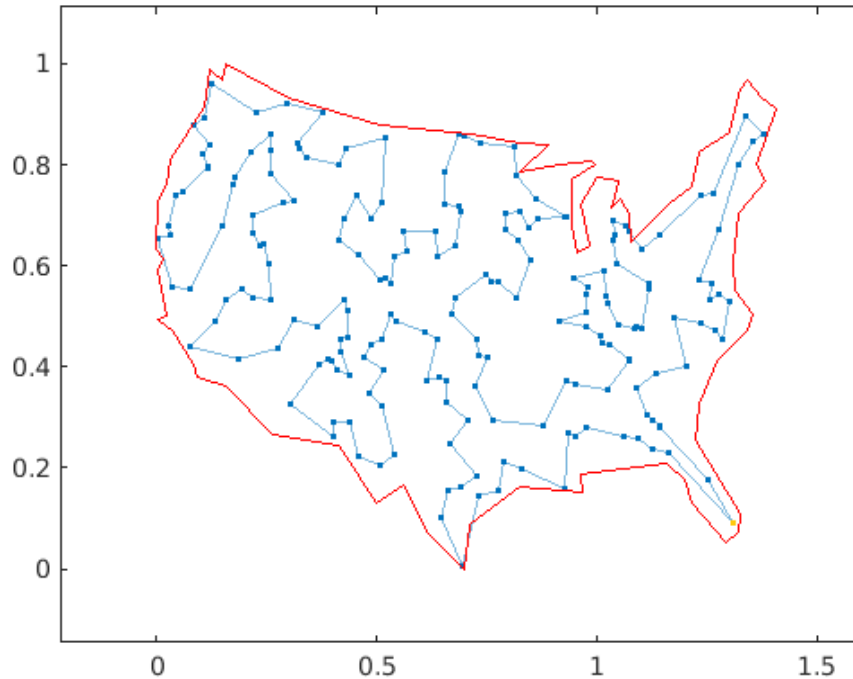


Figure 2-2. TSP: Lowest cost tour [10]

Moving the TSP example to the problem in hand, the cities are the asteroids that will be the nodes of the Search Space or Search Grid. Each combination of asteroids will have a *cost* which can be the Δv or the Time of Flight (ToF) which normally wants to be minimised. The cost can be calculated by a given function which will be the *objective function* or *fitness function* to be minimised. The objective function of CASTPath will be further explained in 3.2.2. Objective function: Δv .

2.1.2 Mixed-Integer Non-Linear Programming (MINLP) Problems

As introduced in this section, multiple fly-by trajectory design problems are high complexity optimisation problems. In concrete, an asteroid tour trajectory design is a Mixed-Integer Non-Linear Programming (MINLP) problem, also known as Hybrid Optimisation Control Problems (HOPC). MINLP problems are one of the most general modelling paradigms in the mathematics field. It combines the difficulties of Mixed-Integer Linear Programming (MILP), containing both discrete and continuous variables, and of Non-Linear Programming (NLP),

which are non-linear in the objective function and/or in its constraints [11]. MINLP problems are mathematically defined as follows [11], [12]:

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{y}) \quad \forall \begin{cases} \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \\ \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^n \end{cases}, \\ \text{subject to} \quad & c_i(\mathbf{x}, \mathbf{y}) = 0 \quad \forall i \in [1, m] \\ & c_i(\mathbf{x}, \mathbf{y}) \leq 0 \quad \forall i \in (m, p] \end{aligned} \tag{2-1}$$

Where f and c_i are continuously differentiable functions, being f the objective function to be minimised and c_i the constraint functions of the problem, $c_i : \mathbb{R}^n \rightarrow \mathbb{R}$. \mathbf{x} and \mathbf{y} are the continuous and discrete variables contained in the sets \mathcal{X} and \mathcal{Y} , respectively. The equality constraint subset has a cardinality of m and the inequality constraint subset has a cardinality of $p - m$, $(m, p) \in \mathbb{N}$. $\mathcal{X} \subseteq \mathbb{R}^n$ is a bounded polyhedral set¹ and $\mathcal{Y} \subseteq \mathbb{Z}^n$ is the set of integer variables. Lower and upper bounds $l \leq c_i(\mathbf{x}, \mathbf{y}) \leq u$ or maximisation functions can also be included [11].

MINLP has a wide range of applications, within which Space Trajectory Design (MINLP-STD) stands out for the CASTPath project. The complexity of the problem has led to deep analysis on MINLP solvers on the research community. Schlueter [13] proposes *MIDACO*, a MINLP with ACO and Oracle Penalty Method for challenging space problems. Shlueter et al. [14] applied the solver, among other space applications, to ESA Advanced Concept Team global trajectory optimisation problems and to an interplanetary space mission design to Jupiter. Other space trajectory design modelling by MINLTP can be seen in Bellome et al. [15], where the application is Multiple-gravity assist (MGA) trajectories of planet fly-bys, or Chai et al. [16], for the optimisation of a constrained trajectory of space manouvre vehicles.

¹ A set in \mathbb{R}^n is said to be polyhedral if it is the intersection of a finite number of closed half spaces [44].

2.2 Optimisation techniques on Asteroid Tour Trajectory Design

After analysing the types of problem of an asteroid tour trajectory design, this section presents the different optimisation techniques in the literature and studies the possible implementation of different types of solvers.

2.2.1 Deterministic and stochastic solvers

MINLP problems are solved by optimisation algorithms. Optimisation algorithms can be classified into three main branches (see Figure 2-3): exact methods (or deterministic), heuristic methods and metaheuristic methods (these last two are stochastic).

Deterministic or exact methods are “determined” to find a unique solution or output to the problem given a certain number of known inputs, if and only if the problem itself is deterministic [17]. As a very simple example, a deterministic solver can be the sum of two input variables say, x and y , which give the output z . Given $x = 5$ and $y = 20$, the output z is determined to be 25, $z = x + y = 25$.

Stochastic or probabilistic methods, on the contrary, do not give a determined output as one or more of their elements are random. Thus, for the given inputs, the solver can output several different results [17]. Following the previous example, take that the solver now, instead of a simple sum, is the sum of a random number between 0 and 1 and the two inputs: $z = rand + x + y$. For the previous fixed given inputs, this solver now can give many numbers between 25 and 26, being non-deterministic.

It is important to note that there exist several randomness distributions, the most common one being the normal distribution. The randomness applied to the stochastic solver should follow the statistical distribution of the nature of the problem. Types of different statistical distributions can be seen in [18].

For this thesis, due to the high complexity of the problem, metaheuristic techniques will be used. However, both deterministic and stochastic techniques have been used to solve MINLP [19]. Some examples of deterministic

techniques are Outer Approximation (OA) [20], [21], Branch and Bound (BB) [22], [23], and Generalised Benders Decomposition (GBD) [24].

Stochastic solvers are used to solve MINLP because of the non-linearity on the fitness function (the function that gives a score to the possible solution) decision variables [11]. The most common ones are Genetic Algorithms (GA) [25] or Ant Colony Optimisation (ACO) [26] and Particle Swarm Optimisation (PSO) [27], which both come inside the category of Swarm Intelligence (SI).

2.2.2 Metaheuristic algorithms

A definition of heuristic by Kenny et al. [28] is given as “*the algorithm designed to solve a problem in a faster and more efficient fashion than traditional methods by sacrificing optimality, accuracy, precision, or completeness for speed*”. Metaheuristics can be seen as solvers that go beyond the heuristic, to solve problems for which there does not exist the best or most efficient solver. While heuristics are designed to solve a specific problem type, metaheuristics can be adapted and solve any problem. Heuristics (where metaheuristics can be included) in general do not guarantee global optima, but they find sufficiently good solutions for cases when the complexity of the problem is very high, there are not enough inputs, or the computational time wants to be limited.

Most of metaheuristic algorithms are inspired by nature: swarms or social animals like ant colonies in ACO, fireflies' communication in Firefly Algorithm (FA) or general swarm behaviour in PSO; evolution strategies as genetic evolution in GA, the obligate parasitism of cuckoo birds in Cuckoo Search (CS) or Coral Reefs Optimisation (CRO), etc. The algorithms imitate the intelligence of nature to apply it as an optimisation solver of many types of problems. Figure 2-3 shows a full classification of several metaheuristic algorithms, inside the 3-branch classification of optimisation algorithms. This thesis will use the Ant Colony Optimisation, which is one of the principal solvers used to solve TSP. The algorithm is introduced later in the document, in 4.5. Ant Colony Optimisation metaheuristic, followed by the proposed novel ACO solver for asteroid tour trajectory design.

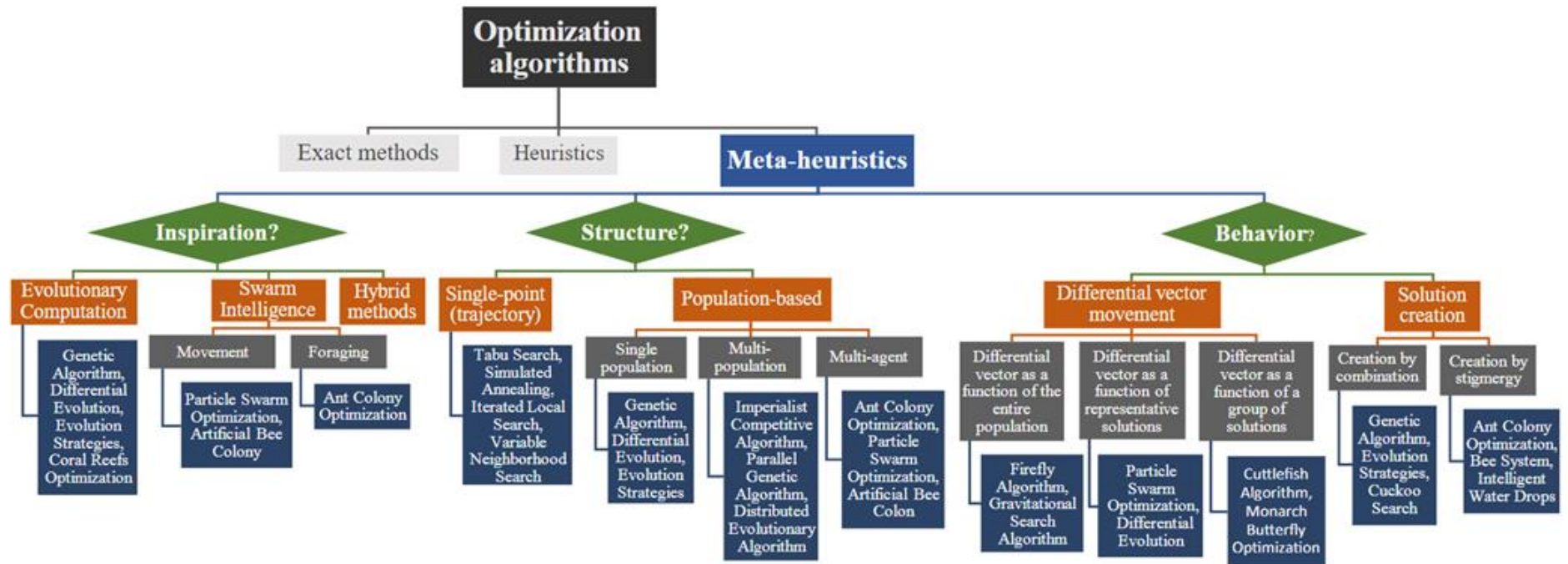


Figure 2-3. Classification of Optimisation algorithms, Metaheuristics

Other types of heuristics considered to tackle the CASTAway tour problem are:

- **Multitasking:** Multiform optimisation uses multitasking to solve one problem by defining different formulations [29]. For CASTPath, each formulation could be the different asteroid sequences, being used to solve one problem as different problems reducing computational time.
- **Multimodal optimisation:** This optimisation technique is used to seek multiple solutions, or local minima [30], [31], which is interesting for the problem in hand to give several asteroid sequences solutions to the scientific researchers.
- **Novelty search:** As the name describes, this algorithm aims for novelty or diversity in its search while evaluating the solutions based on their k-nearest neighbours [32]. This is again, interesting for providing not only feasible but diverse solutions.
- **Multi/Many objective optimisation:** These algorithms evaluate the solutions considering more than one objective. This approach was considered to solve the problem that Tena predicted in his thesis [33] on the prompt selection of very distanced asteroids. This problem will be further discussed with the chosen metaheuristic in 4.5.3. Multiobjective formulation of ACO.

2.3 Previous work on CASTPath

Within the CASTPath project for the mission analysis of the CASTAway mission, a toolbox on MATLAB was developed which has been longer extended in this thesis. The mission trajectory proposal by OHB System AG and Dr Joan-Pau Sánchez Cuartiellas established the baseline for the toolbox, that was further developed by Curzi [34] and later, by Tena [33] and Bellome et al. [3].

Curzi [34] provided a filter of the asteroid database based on Minimum Orbital Distance (MOID) time of the asteroids that is longer presented in 3.2.1. Filtering by MOID and baseline Earth-Mars leg.

Sánchez et al. [2] implemented a standard GA using the MATLAB built-in function to choose among different transfer strategies to reach to the AMB. The best transfer was chosen to be an Earth-Mars sequence (one Mars gravity assist) that was further investigated later by Tena [33] to choose a concrete Earth-Mars leg, the details on this will be presented in 3.2. Problem Baseline.

In the same work, Sánchez et al. [2] analysed multiple fly-by asteroid sequences options by splitting the problem into two subproblems:

- P1: A discrete combinatorial problem, solved with a Branch and Bound deterministic technique, to find feasible sequences of 10 asteroids fly-bys
- P2: A continuous optimisation problem that aims on minimising the Δv of P1 sequence outputs by choosing the best dates for each fly-by.

Tena [33] took the same problem structure approach and solved the problem with a MOID filter of 0.05 AU asteroid database and looking for sequences of 12 asteroids. His thesis was focused on analysing different heuristics for the P1 subproblem, given an approximation of the asteroid legs Δv that was then refined by the P2. The heuristics compared were: GA, ACO, ACO with Tabu of legs of more than 1 km/s, ACO with the Tabu and GA. He obtained the best results on CASTPath till the moment for the last solver case² for an execution time of 1 h in Cranfield University's HPC of 7.6962 km/s.

Tena [33] also explained the strong dependency of the Δv between asteroids and the possibility of constructing a *score matrix*. This was disregarded because of the dependency of the previous asteroids and therefore, the non-uniqueness of the cost of the asteroid legs. However, this possibility was further analysed in this thesis where a construction of a Score Matrix is achieved, which will be presented in section 4.2.

Finally, Bellome et al. [3] found the global optimum to the problems for different MOID thresholds, based on Dynamic programming. This data will be used to analyse the solutions found in this thesis with the proposed solver.

² Note that there is a typo in the ACO Tabu score in [33]. The best Δv of this solver is 7.84 km/s.

3 PROBLEM STATEMENT

This section provides the background and baseline of the problem being analysed. Section 3.1 defines the CASTAway asteroid tour optimisation problem and its specific objectives. 3.2. Problem Baseline, enunciates all the assumptions, details the baseline Earth-Mars transfer and defines the objective function. Section 3.3. Combinatorial Problem aims at demonstrating the reasoning behind the use of metaheuristics to solve the STD problem and the last section 0 explains the main drivers and constraints to tackle the problem.

3.1 CASTAway asteroid tour optimisation problem

CASTPath is a computational tool for Mission Analysis and Trajectory Design project that aims on studying the feasibility of CASTAway mission. The toolbox, coded in MATLAB, consists of several astrodynamics and mission analysis functions that analyse different possibilities of multiple asteroid fly-bys, aiming to obtain several attractive tour options for the CASTAway mission: **low-energy multiple fly-by asteroid tours of low Time of Flight (ToF)**.

The asteroid database used is called CATABC by Curzi [34] that contains information about **101,993 asteroids of the Asteroid Main Belt**. CATABC is a pruned asteroid-belt database from the initial 600,000 asteroids in the online available HORIZON catalogue in the JPL Small-Body Database [35]. The database is pruned by a threshold of 2 km/s for the fly-by relative velocity. For more details on the CATABC database refer to Curzi [34].

3.2 Problem Baseline

In order to analyse as many celestial bodies as possible, the CASTAway trajectory needs to be designed as to maximise the time the telescope spends within the AMB, so maximising the ToF within the AMB. However, at the same time, the total ToF and the Δv are aimed to be minimised. Thus, the Δv will be minimised and the ToF will be limited by an upper threshold. Previous work on CASTAway tour design [2] show that an Earth-Mars leg helps increasing the

time spent within the AMB, as the Mars swing-by is used to increase the spacecraft energy.

A multiple asteroid tour problem is a MINLP (introduced in 2.1.2. Mixed-Integer Non-Linear Programming (MINLP) Problems), that can be modelled therefore as to minimise the cost function of the fly-bys' total Δv Δv_{tot} with an upper bound constraint in the ToF of ToF_{max} . The discrete variables correspond to the asteroid targets A_k for the fly-bys of the spacecraft (where k is the MOID time index) and the continuous variables \mathbf{t} are the ToF of each tour leg. The non-linearity of the problem is due to the motion of the spacecraft and the targets with respect to the Sun, being a two-body gravitational problem.

Following the general MINLP definition (2-1), the MINLP-STD problem can be defined as:

$$\begin{aligned}
 \min \quad & \Delta v_{tot} = f(\mathbf{A}, \mathbf{t}) \quad \forall \left\{ \begin{array}{l} \mathbf{A} \in \mathbb{Z}^{L_{tour}^{ast}+1} \\ \mathbf{t} \in \mathbb{R}^{L_{tour}^{ast}+1} \end{array} \right\} \\
 \text{where} \quad & \mathbf{A} = \{A^0, A^1, \dots, A^m, \dots, A^{L_{tour}^{ast}}\}, \\
 & A^m = A_k \in \mathbb{Z}^{n_{ast,tot}+1} \mid k \in [0, n_{ast,tot}] \\
 & \mathbf{t} = \{t_0, ToF_1, \dots, ToF_{L_{tour}^{ast}}\} \\
 \text{subject to} \quad & \sum_{i=1}^{L_{tour}^{ast}} ToF_i \leq ToF_{max}
 \end{aligned} \tag{3-1}$$

Where \mathbf{A} is the set of asteroids of a tour of length L_{tour}^{ast} , where the first element is always Earth, $A^0 = A_0$ (and the rest of elements are ordered by the sequence index m).

However, it needs to be highlighted that the aim of CASTPath is **not to find the global minimum, but to find a set of feasible tour solutions** to the CASTAway problem. In this way, the scientific community can choose an interesting set of asteroids among different low-cost fast asteroid tour trajectories, making a trade-off between the scientific interest and the STD optimality. Therefore, this problem can be better seen as a **Constraint Satisfaction Problem** than as a Global Optimisation Problem, with the constraints that will be presented in 0.

Drivers and constraints.

As mentioned in 2.1.2. Mixed-Integer Non-Linear Programming (MINLP) Problems, MINLP-STD are very challenging problems to solve. In the next subsections some assumptions are made so that the problem is simplified.

3.2.1 Filtering by MOID and baseline Earth-Mars leg

The Δv is the difference in velocity the spacecraft needs to achieve in order to change its trajectory at a certain instance to intersect the target asteroid orbit at its Minimum Orbital Intersection Distance (MOID) time. The MOID is the minimum distance between the nearest two points of two different orbits [36]. For a given inclined asteroid orbit, the MOID would be more particularly defined as the distance between the point at which the asteroid orbit intersects the orbital plane of the initial trajectory of the spacecraft, and the closest point on the spacecraft's orbit.

Following previous work on CASTPath by Tena, and in order to compare the results, the same threshold of $\text{MOID} \leq 0.05 \text{ AU}$ is chosen (see Tena [33] for more details). However, other thresholds can be applied in the filtering prior to the solver as future work. This pruning lowers the number of asteroids from the initial 101,993 to $n_{ast} = 158$ asteroids. Thus, the baseline Earth-Mars leg is the following:

Table 3-1. Baseline Earth-Mars leg from [33]

Launch date	Time of Flight (ToF)	Wet Mass
24/12/2030 12:00 GMT (11,315 MJD2000)	800 days	1,827 kg

The *MOID time* t_{MOID} is defined in this project as the epoch at which the asteroid reaches its MOID. It is assumed that the MOID time is the best epoch to fly-by the asteroid. Following this reasoning, **all the asteroid encounters will be analysed at their corresponding MOID time within the $\text{MOID} \leq 0.05 \text{ AU}$ threshold.** All the asteroids in the catalogue will be ordered in MOID time and the *MOID time index* will be defined as the ascending chronological

order resulting indices. Asteroid 0 will be Earth, the departing point for all the possible tour sequences, asteroid 1 will be the asteroid with the lowest MOID time and asteroid 158 the asteroid with the highest MOID time.

3.2.2 Objective function: Δv

With this structure of the problem, where a threshold of the ToF is applied, the objective function can be reduced to minimise the total Δv : Δv_{tot} . Each leg is a Lambert Arc whose cost will be the required change in velocity Δv to follow it: the norm of the difference between the velocity vector at the arrival of the asteroid and the velocity vector of departure at this point, i.e., at its MOID time. Figure 3-1 illustrates the spacecraft trajectory in 2D between to asteroids of MOID time indices j and k with the arrival vector $\vec{v}_{j,arrival}$ and departure vector $\vec{v}_{j,departure}$ from asteroid j and the Δv of the leg given by:

$$\Delta v_{leg(j,k)} = |\vec{v}_{j,departure} - \vec{v}_{j,arrival}| \tag{3-2}$$

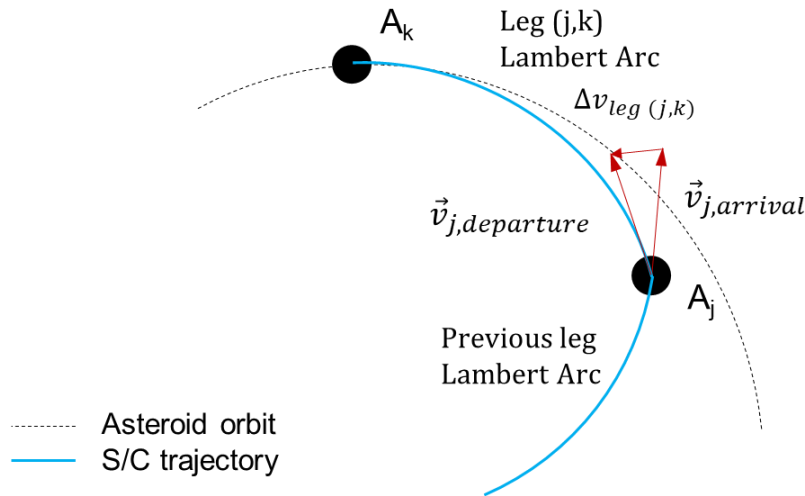


Figure 3-1. 2D sketch of the s/c trajectory and Δv between asteroid j and k

After the MOID filtering, and knowing that each asteroid A_k has a unique MOID time, the objective function of the MINPL-STD problem is simplified to the following expression:

$$\min \quad \Delta v_{tot} = f(A_k), \quad \forall A_k \in \mathcal{G} \subseteq \mathbb{Z}^{n_{ast}+1} \mid k \in [0, n_{ast}] \tag{3-3}$$

3.3 Combinatorial Problem

As introduced in 2.1. Asteroid Tour Trajectory Design, the problem in matter is a Combinatorial Problem, where the solutions are combinations of the different elements of the Search Space, i.e., combinations of asteroids. As the asteroids are ordered in MOID times, the order of the combinations matter, being the tour sequences non-repeated permutations of the list of asteroids. The number of non-repeated permutations of n elements (asteroids) for k long sequences (or number of visited asteroids) is given by:

$$P(n, k) = \frac{n!}{(n - k)!} \quad (3-4)$$

The provided catalogue contains a total of 101,993 asteroids in the AMB. Aiming for a tour of $k = L_{tour}^{ast} = 12$ asteroids, the total number of permutations with $n = n_{ast,tot} = 101,993$ is $P(n_{ast,tot}, L_{tour}^{ast}) \approx 1.27 \times 10^{60}$. If solving 12 Lambert-Arcs and one Mars swing-by, i.e, one tour sequence, takes 1 ms [33], it would take 4.02×10^{15} years for a full exploration of all the possible sequences in the catalogue. As an example for comparison, the universe is nearly 1.4×10^{10} years old. Therefore, **a full exploration by means of an exact method is not feasible.**

As introduced in the previous section 3.2, filtering the asteroids with a MOID bigger than 0.05 AU reduces the database to 158 asteroids. For a sequence of $n = n_{ast} = 158$ and $k = L_{tour}^{ast} = 12$ there exist a total of $P(n_{ast}, L_{tour}^{ast}) \approx 1.58 \times 10^{26}$ possible permutations, which is still, a too large number of evaluations. Thus, even if the problem definition has been simplified as (3-3), still it is a very complex MINTP problem to solve.

3.4 Drivers and constraints

The numbers shown in the previous section 3.3 illustrate the complexity of the problem and justify the use of metaheuristics, introduced in 1.3. Optimisation and Metaheuristic techniques, to solve the CASTAway trajectory design. Therefore, the focus of CASTPath has been extended to the **comparison of different solvers** to achieve the most robust, low computational cost **algorithm to find several local optima** to the problem. Note that heuristics do not guarantee finding the global optimum [37].

As already stated in 1.4. Objectives and Scope of the Thesis, this thesis has the main objective of developing a solver to compare and, ideally, improve previous work in CASTPath. Bearing this in mind, the two requirements or constraints for the project are:

REQ-001 Sequences shall contain 12 asteroids of increasing MOID

REQ-002 The total Δv cost of the tour shall be of 9 km/s as maximum

Even if the requirement for CASTAway mission is to find sequences of at least 10 asteroids, REQ-001 sets it to 12 asteroids following Tena's work on CASTPath [33]. REQ-002 is set at a threshold of 9 km/s based on previous work on CASTPath by Mohd [38] and Tena [33] who also apply this threshold. Both requirements are set for the sake of comparison with previous analysis on CASTPath. In this thesis, when referring to *feasible* solutions or tours, it means those tours comply with both of the requirements presented.

4 PROPOSED APPROACH

In this section the proposed approach is presented. Section 4.1. Search Space, introduces a new formulation of the Search Space that makes the nodes paths unique, making possible the construction of a score matrix presented in 4.2. Score Matrix. A pruning based on the project requirements is presented in 4.3. Cleaning of Search Space. The following section 4.4. Similarity Measure, presents a score tool for quantifying the diversity of the solutions. Finally, section 4.5. Ant Colony Optimisation metaheuristic explains the standard ACO algorithm and defines the proposed or implemented ACO.

4.1 Search Space

Consider the leg composed by asteroids A_{18}^2 and A_{31}^3 that correspond to the 2nd and 3rd asteroids in a candidate solution sequence. The Δv required for the spacecraft to follow the Lambert Arc that connects asteroid 18 with asteroid 31 is dependent on the previous asteroid, the 1st asteroid in the sequence. The Δv of a leg (A_j^{m-1}, A_k^m) subject to previous visited asteroid A_i^{m-2} is notated as $\Delta v_{j,k}^i$. Figure 4-1 shows two different cases of possible tours, changing the first asteroid in the sequence. Note that the drawing is a 2D sketch with made-up indices for the sake of illustration, which does not correspond to the real trajectories of the numbered asteroids.

The angle between the tangent at A_{18}^2 of (A_{10}^1, A_{18}^2) leg Lambert Arc and the one of (A_{18}^2, A_{31}^3) leg Lambert Arc is smaller than the angle between the tangent at A_{18}^2 of (A_6^1, A_{18}^2) leg Lambert Arc and the one of (A_{18}^2, A_{31}^3) leg Lambert Arc. In other words, the spacecraft needs to deviate less its trajectory to flyby asteroid 18 when coming from asteroid 10 than 6. Thus, $\Delta v_{18,31}^{10} < \Delta v_{18,31}^6$ and it is concluded that the Δv of a leg is dependent on the asteroid prior to that leg.

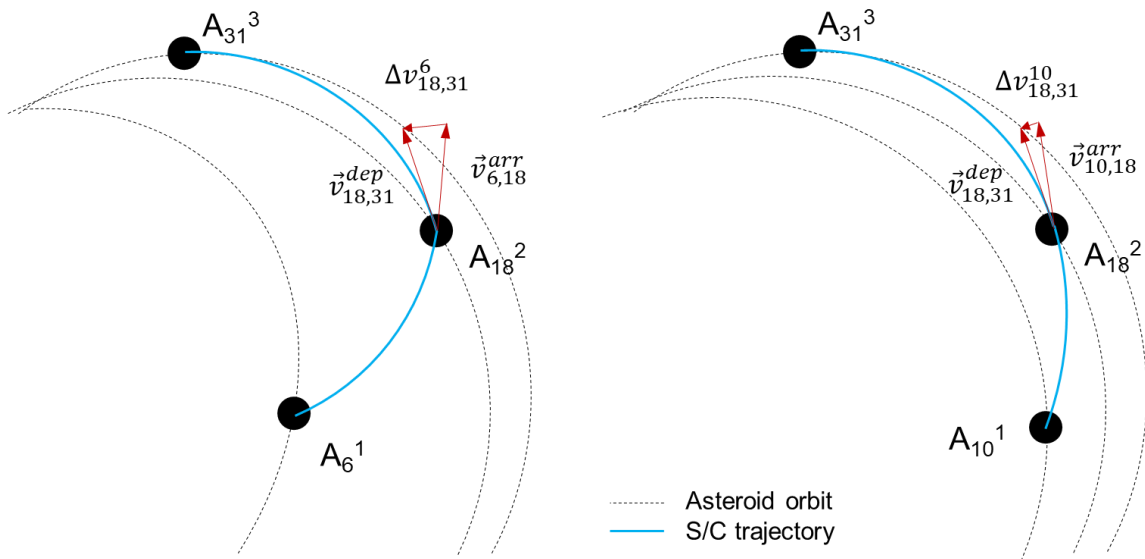


Figure 4-1. Comparison of a leg Δv changing prior asteroid in a 2D sketch (example with not real asteroid indices)

Because of this tri-dependency, the Search Space is a tree search that can be modelled as a multi-dimensional space of connected subspaces, being the Search Space a Search Grid \mathcal{G} . These subspaces $S_i \subseteq \mathcal{G}$ contain nodes which are pairs of asteroids. The initial subspace S_0 contains all the pairs of asteroids $(A_0, A_i) \forall i \in \mathbb{N} \mid 1 < i \leq n_{ast}$. S_0 is connected to a set of subspaces $S_i \forall i \in \mathbb{N} \mid 0 < i \leq n_{ast}$ each of which are again connected to a set of subspaces $S_j \forall j \in \mathbb{N} \mid i < j \leq n_{ast}$ and so on. The total number of subspaces in \mathcal{G} is n_{ast} as the last subspace is $S_{n_{ast}-1}$ containing node $(A_{n_{ast}-1}, A_{n_{ast}})$.

Figure 4-2 shows an example of a chosen path in the Search Space \mathcal{G} . When choosing a node, the first asteroid in the node will be equal to the second asteroid in the previous node (linked by the same colour in the figure). This is how the subsets are connected among themselves. In this space, the cost of the paths between the nodes are unique, which is the main advantage of modelling the Search Space in this way.

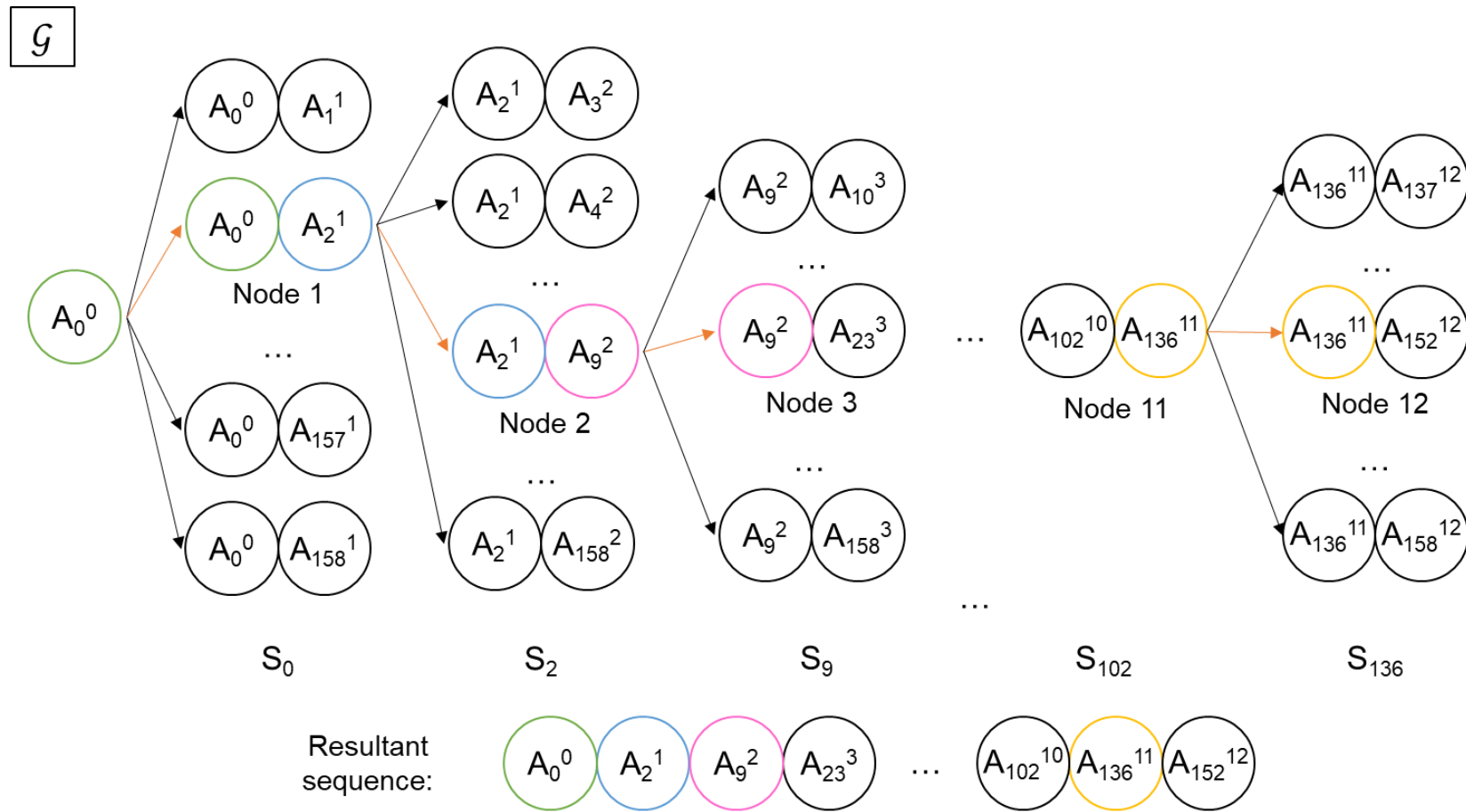


Figure 4-2. Schematic diagram of the Search Grid modelled by subsets and pairs of nodes, an example of a chosen path

4.2 Score Matrix

As explained in the previous section 4.1, the cost of a tour leg is given by the Δv of going from asteroid A^{m-1} to asteroid A^m subject to having passed by asteroid A^{m-2} , so for the triplet $(A_i^{m-2}, A_j^{m-1}, A_k^m)$: $\Delta v_{j,k}^i$. Being the Δv a tri-asteroid dependent cost, unique for each of the legs of the Search Space, a Score Matrix (SM) with the following characteristics was created for this research:

- **Tri-structured:** the matrix relates the pair of nodes of departure in rows with the asteroids of arrival in columns.
- **Unique:** All the costs contained within the matrix are constant due to its structure.
- **Strictly upper triangular form pattern repetition:** If the Score Matrix was asteroid-to-asteroid the matrix would be a strictly upper triangular one (all the main diagonal and lower diagonal are non-feasible paths). As the rows of the matrix are pairs of nodes, this pattern is repeated downwards through the matrix.

Table 4-1 represents the Score Matrix. All the entries marked with an 'x' are non-feasible paths, as the asteroid indices can only increase in the sequence of the tour, neither decrease nor be equal.

Table 4-1. Score Matrix (SM)

From \ To	A_1	A_2	A_3	A_4	A_5	\dots	A_{156}	A_{157}	A_{158}
(A_0, A_1)	x	$\Delta v_{1,2}^0$	$\Delta v_{1,3}^0$	$\Delta v_{1,4}^0$	$\Delta v_{1,5}^0$	\dots	$\Delta v_{1,156}^0$	$\Delta v_{1,157}^0$	$\Delta v_{1,158}^0$
(A_0, A_2)	x	x	$\Delta v_{2,3}^0$	$\Delta v_{2,4}^0$	$\Delta v_{2,5}^0$	\dots	$\Delta v_{2,156}^0$	$\Delta v_{2,157}^0$	$\Delta v_{2,158}^0$
(A_0, A_3)	x	x	x	$\Delta v_{3,4}^0$	$\Delta v_{3,5}^0$	\dots	$\Delta v_{3,156}^0$	$\Delta v_{3,157}^0$	$\Delta v_{3,158}^0$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_0, A_{156})	x	x	x	x	x	\dots	x	$\Delta v_{156,157}^0$	$\Delta v_{156,158}^0$
(A_0, A_{157})	x	x	x	x	x	\dots	x	x	$\Delta v_{157,158}^0$
(A_0, A_{158})	x	x	x	x	x	\dots	x	x	x
(A_1, A_2)	x	x	$\Delta v_{2,3}^1$	$\Delta v_{2,4}^1$	$\Delta v_{2,5}^1$	\dots	$\Delta v_{2,156}^1$	$\Delta v_{2,157}^1$	$\Delta v_{2,158}^1$
(A_1, A_3)	x	x	x	$\Delta v_{3,4}^1$	$\Delta v_{3,5}^1$	\dots	$\Delta v_{3,156}^1$	$\Delta v_{3,157}^1$	$\Delta v_{3,158}^1$
(A_1, A_4)	x	x	x	x	$\Delta v_{4,5}^1$	\dots	$\Delta v_{4,156}^1$	$\Delta v_{4,157}^1$	$\Delta v_{4,158}^1$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_2, A_3)	x	x	x	$\Delta v_{3,4}^2$	$\Delta v_{3,5}^2$	\dots	$\Delta v_{3,156}^2$	$\Delta v_{3,157}^2$	$\Delta v_{3,158}^2$
(A_2, A_4)	x	x	x	x	$\Delta v_{4,5}^2$	\dots	$\Delta v_{4,156}^2$	$\Delta v_{4,157}^2$	$\Delta v_{4,158}^2$
(A_2, A_5)	x	x	x	x	x	\dots	$\Delta v_{5,156}^2$	$\Delta v_{5,157}^2$	$\Delta v_{5,158}^2$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_{156}, A_{157})	x	x	x	x	x	\dots	x	x	$\Delta v_{157,158}^{156}$
(A_{157}, A_{158})	x	x	x	x	x	\dots	x	x	x

4.3 Cleaning of Search Space

Let N be the set of feasible solutions within the whole Search Space grid \mathcal{G} . With the aim of reducing the computational cost of the search, a reduction in the Search Space dimensions is done by removing all the elements that are not included in N . The new cleaned Search Space is notated as \mathcal{G}_c .

The cleaning of non-feasible solutions of the Search Space is done considering the next two constraints of the problem (see 0.

Drivers and constraints):

- Sequences shall contain 12 asteroids of increasing MOID (REQ-001)
- The total Δv cost of the tour shall be of 9 km/s as maximum (REQ-002)

The first constraint REQ-001 implies some asteroids cannot be solutions of the sequence depending on their position m in the sequence. Thus, the feasible solutions set considering both constraints is given by (4-1):

$$N = \left\{ \left\{ A^0 A^1 \dots A^m \dots A^{L_{tour}^{ast}} \right\} \in \mathbb{N} \mid \begin{array}{l} A^m \leq n_{ast} - (m + L_{tour}^{ast}) \\ \Delta v_{tot} \leq \Delta v_{max} \end{array} \right\} \quad (4-1)$$

Where Δv_{tot} is the sum of the Δv of all the legs that compose the 12-asteroid sequence, adding Earth-first asteroid Lambert Arc cost $\Delta v_{A^0, A^1}$.

$$\Delta v_{tot} = \Delta v_{A^0, A^1} + \sum_{m=2}^{L_{tour}^{ast}} \Delta v_{A^{m-1}, A^m} \quad (4-2)$$

However, as the Search Space nodes are modelled as pairs of asteroids, the only pre-cleaning that can be done for the solver is given by (4-3), that does not consider the position of the asteroid in the sequence for the first constraint. For the second constraint, REQ-002, it is ensured that feasible solutions are not removed from the cleaned Search Space \mathcal{G}_c by marking as non-feasible the asteroid legs with a $\Delta v \geq 9$ km/s. If none of the possible triplets from a pair (A_i, A_j) comply this criterion, the pair is removed from \mathcal{G}_c .

$$\mathcal{G}_c = \left\{ (A_i, A_j) \in \mathbb{N} \mid \begin{array}{l} j \leq n_{ast} - (i + L_{tour}^{ast}) \\ \Delta v_{j,k}^i \not\geq \Delta v_{max} \forall k \in \mathbb{N} \setminus \{j+1, n_{ast}\} \end{array} \right\} \quad (4-3)$$

Imposing these two constraints of the problem into the Score Matrix some Δv are removed, leaving some pairs and triplets as non-feasible solution of the Search Space. The cleaning of the Score Matrix is shown in Table 4-2, where the removed rows (i.e., pairs) are due to the first constraint, and the red crosses are due to the second constraint.

Table 4-2. Score Matrix (SM) Cleaning

From \ To	A_1	A_2	A_3	A_4	A_5	\dots	A_{156}	A_{157}	A_{158}
(A_0, A_1)	x	x	$\Delta v_{1,3}^0$	$\Delta v_{1,4}^0$	$\Delta v_{1,5}^0$	\dots	$\Delta v_{1,156}^0$	$\Delta v_{1,157}^0$	$\Delta v_{1,158}^0$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_0, A_{146})	x	x	x	x	x	\dots	x	$\Delta v_{146,157}^0$	$\Delta v_{146,158}^0$
(A_0, A_{147})	x	x	x	x	x	\dots	x	$\Delta v_{147,157}^0$	$\Delta v_{147,158}^0$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_1, A_2)	x	x	x	$\Delta v_{2,4}^1$	$\Delta v_{2,5}^1$	\dots	$\Delta v_{2,156}^1$	$\Delta v_{2,157}^1$	$\Delta v_{2,158}^1$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_1, A_{147})	x	x	x	x	x	\dots	$\Delta v_{147,156}^1$	$\Delta v_{147,157}^1$	$\Delta v_{147,158}^1$
(A_1, A_{148})	x	x	x	x	x	\dots	$\Delta v_{148,156}^1$	$\Delta v_{148,157}^1$	$\Delta v_{148,158}^1$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_{10}, A_{11})	x	x	x	x	x	\dots	$\Delta v_{11,156}^{10}$	$\Delta v_{11,157}^{10}$	$\Delta v_{11,158}^{10}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_{10}, A_{156})	x	x	x	x	x	\dots	x	x	$\Delta v_{156,158}^{10}$
(A_{10}, A_{157})	x	x	x	x	x	\dots	x	x	$\Delta v_{157,158}^{10}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_{11}, A_{12})	x	x	x	x	x	\dots	$\Delta v_{12,156}^{11}$	$\Delta v_{12,157}^{11}$	$\Delta v_{12,158}^{11}$
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots
(A_{11}, A_{157})	x	x	x	x	x	\dots	x	x	$\Delta v_{157,158}^{11}$
(A_{11}, A_{158})	x	x	x	x	x	\dots	x	x	x
(A_{156}, A_{157})	x	x	x	x	x	\dots	x	x	$\Delta v_{157,158}^{156}$
(A_{156}, A_{158})	x	x	x	x	x	\dots	x	x	x
(A_{157}, A_{158})	x	x	x	x	x	\dots	x	x	x

With this pruning, the **total number of pairs of nodes is reduced a 5%** from 12,561 to 11,890 (11,889 pairs of nodes “from”), and the **total number of feasible triplets is reduced an 8%**, from 657,359 to 604,292. Thus, the Score Matrix SM is a matrix containing Δv of dimensions $11,889 \times 158$:

$$SM = [\Delta v]_{n_{pairs,from} \times n_{ast}} = [\Delta v]_{11,889 \times 158} \quad (4-4)$$

4.4 Similarity Measure

A Similarity Measure tool has been developed for the comparison of solutions encountered by the different algorithms used to solve the CASTAway problem. However, as future work, this tool can also be used within the solver to increase the diversity of the solutions encountered.

Consider the two sequences (Seq) depicted in Figure 4-3. The Similarity Measure considers two possible similarities between the asteroid sequence solutions:

- Asteroid-by-asteroid similarity score (in grey and green): ψ_{ast}
- Subsequences similarity score (in blue and orange): ψ_{sub}

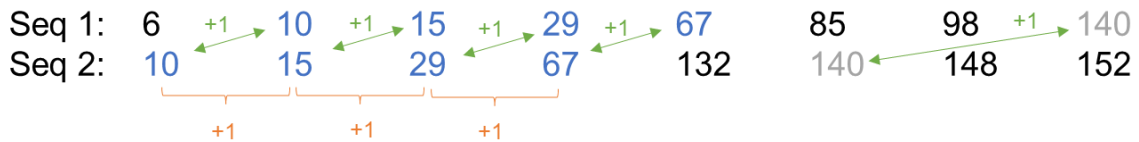


Figure 4-3. Similarity Measure example

As it can be seen, Earth (asteroid 0) and Mars are not considered in the similarity measure as all the sequences pass through these planets. The asteroid-by-asteroid score is the result of adding a unit each time an asteroid is repeated (no matter the position in the sequence); while the subsequences score is the result of adding a unit every 2 equal subsequent asteroids. In the example, $\psi_{ast} = 5$ and $\psi_{sub} = 3$. This scoring is done one independent of each other. In this way, the presence of subsequences adds more similarity: In the example, because of having an equal subsequence of 4 elements, the total score with unitary weights would be 7 for that part.

Each of the scores are pondered by a different weight and a total similarity score is given by:

$$\Psi = w_{ast} \cdot \psi_{ast} + w_{sub} \cdot \psi_{sub} \quad (4-5)$$

Thus, considering unitary weights, the maximum similarity score for 2 equal sequences of 12 asteroids would be of $\Psi = 1 \cdot 12 + 1 \cdot 11 = 23$.

4.5 Ant Colony Optimisation metaheuristic

ACO stands for Ant Colony Optimisation. It is a metaheuristic algorithm that based on ant colonies behaviour, solves complex combinatorial problems, finding local optima. Although nowadays there exist many variants, this algorithm was first proposed in the literature by [39]. In section 4.5.1, this first ACO algorithm called Ant System will be explained. Other variants found in the literature are the Ant Colony System by [40] or the MAX-MIN ant system introduced by [41].

After its definition, a comparison of the implemented ACO with the standard one is made and the proposed ACO solver model is formulated in sections 4.5.2 and 4.5.3. Note that the equations are directly formulated for a Search Space where the nodes are pair of elements, as for the problem analysed in this thesis.

4.5.1 ACO algorithm definition

The principle behind the solver is that ants, when looking for food, pass information to the rest of the colony in an indirect way: by laying a trail of a chemical substance called *pheromone*. The rest of the ants detect this pheromone making a path more attractive to them, and the colony finds an optimum way from their nest to the food in a collaborative and efficient way.

In ACO, some agents called *artificial* ants construct candidate solutions in a discrete graph, whose nodes are equivalent to the cities in the TSP (see section 2.1.1). The construction of a candidate solution for each ant is stochastic and guided by the pheromone laid by other previous ants in the graph. The pheromone τ is analogous to a reward associated to each path in the graph.

The discrete combinatorial problem is solved iteratively by populations of ants. A pseudocode of the algorithm is shown in Algorithm 4-1.

Algorithm 4-1. Ant Colony Optimisation pseudocode

procedure Ant Colony Optimisation

Initialisation

- 1: Set parameters
- 2: Initialise pheromone

Graph exploration

- 3: WHILE (termination criteria [graph exploration] not met) DO

Let population explore

- 4: FOR each ant in the population

Construct a candidate solution

- 5: WHILE (termination criteria [candidate solution] not met) DO

- 6: Calculate probability of choosing next node Equation (4-6)

- 7: Select next node based on probability

- 8: ENDWHILE

- 9: ENDFOR

- 10: Daemon actions (optional)

Update pheromone

- 11: Update pheromone trails with population explorations Equation (4-7)

- 12: ENDWHILE
-

The pheromone is initialised to be equally distributed over the whole search space. The first population of ants starts then constructing its candidate solutions through the graph.

The candidate solution is a sequence of selected nodes of the Search Graph \mathcal{G}_c . Each ant in the population starts with an empty candidate solution tour $\xi = \emptyset$. At each step (i, j) , the ant selects the next node (j, k) from the feasible solutions of the Search Graph, $N(\xi) \subseteq \mathcal{G}_c$, on the construction of its candidate selection based on the probability $P_{(i,j),k}$.

The probability of an ant selecting a node (j, k) depends on two parameters:

- the **pheromone** $\tau_{(i,j),k}$ value weighted by an exponent parameter α and
- an **heuristic parameter** $\eta_{(i,j),k}$ weighted by an exponent parameter β ,

both associated to the path between the two nodes $(i, j), (j, k)$, so to the triplet (i, j, k) .

$$P_{(i,j),k} = \frac{\tau_{(i,j),k}^\alpha \cdot \eta_{(i,j),k}^\beta}{\sum_{(A_i^{m-2}, A_j^{m-1}, A_l^m)} \tau_{(i,j),l}^\alpha \cdot \eta_{(i,j),l}^\beta} \forall (A_i^{m-2}, A_j^{m-1}, A_k^m) \in N(\xi) \quad (4-6)$$

The ant stops searching more nodes when the termination criteria for the candidate solution has been met. This process is iterated for all the n_{ant} ants in the population.

Before updating the pheromone, some daemon actions can be undertaken as part of the specific problem resolution. Daemon actions are for instance, local search routines that ants cannot undergo to move the solutions to local optima.

The pheromone τ is updated after the exploration of the whole population, considering an evaporation rate $\rho \in (0,1]$ applied to the whole graph pheromone and how many ants pass through those paths. This can be seen in equation (4-7) where the pheromone τ of the path from node (i, j) to node (j, k) (so for the triplet (i, j, k)) is evaporated and the new pheromone trail $\Delta\tau_{(i,j),k}^a$ is added.

$$\tau_{(i,j),k} \leftarrow (1 - \rho) \cdot \tau_{(i,j),k} + \sum_{a=1}^{n_{ant}} \Delta\tau_{(i,j),k}^a \quad (4-7)$$

The new pheromone trail of each node is given by the inverse of the total cost \mathcal{L} of ant a tour ξ (being the same for all the nodes of that tour):

$$\Delta\tau_{(i,j),k}^a = \begin{cases} \frac{1}{\mathcal{L}_\xi^a} & \text{if ant } a \text{ has gone through } ((i, j), k) \text{ in its tour } \xi \\ 0 & \text{otherwise} \end{cases} \quad (4-8)$$

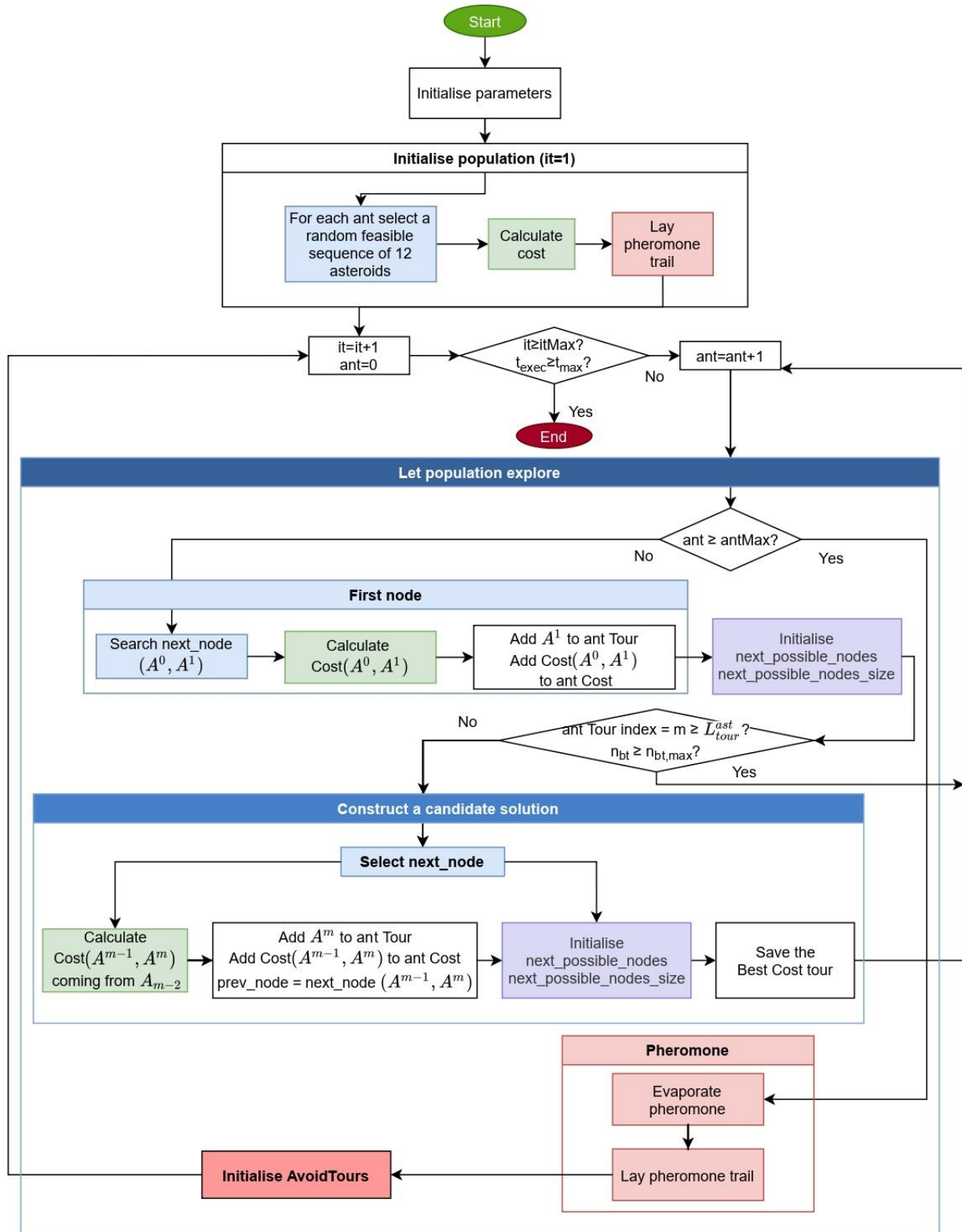
The next population of ants repeats this process with more information than the previous one. The termination criteria for the graph exploration can be fixed by a maximum number of populations, by a maximum computational time, or a criterion related to the problem.

4.5.2 Implemented ACO algorithm

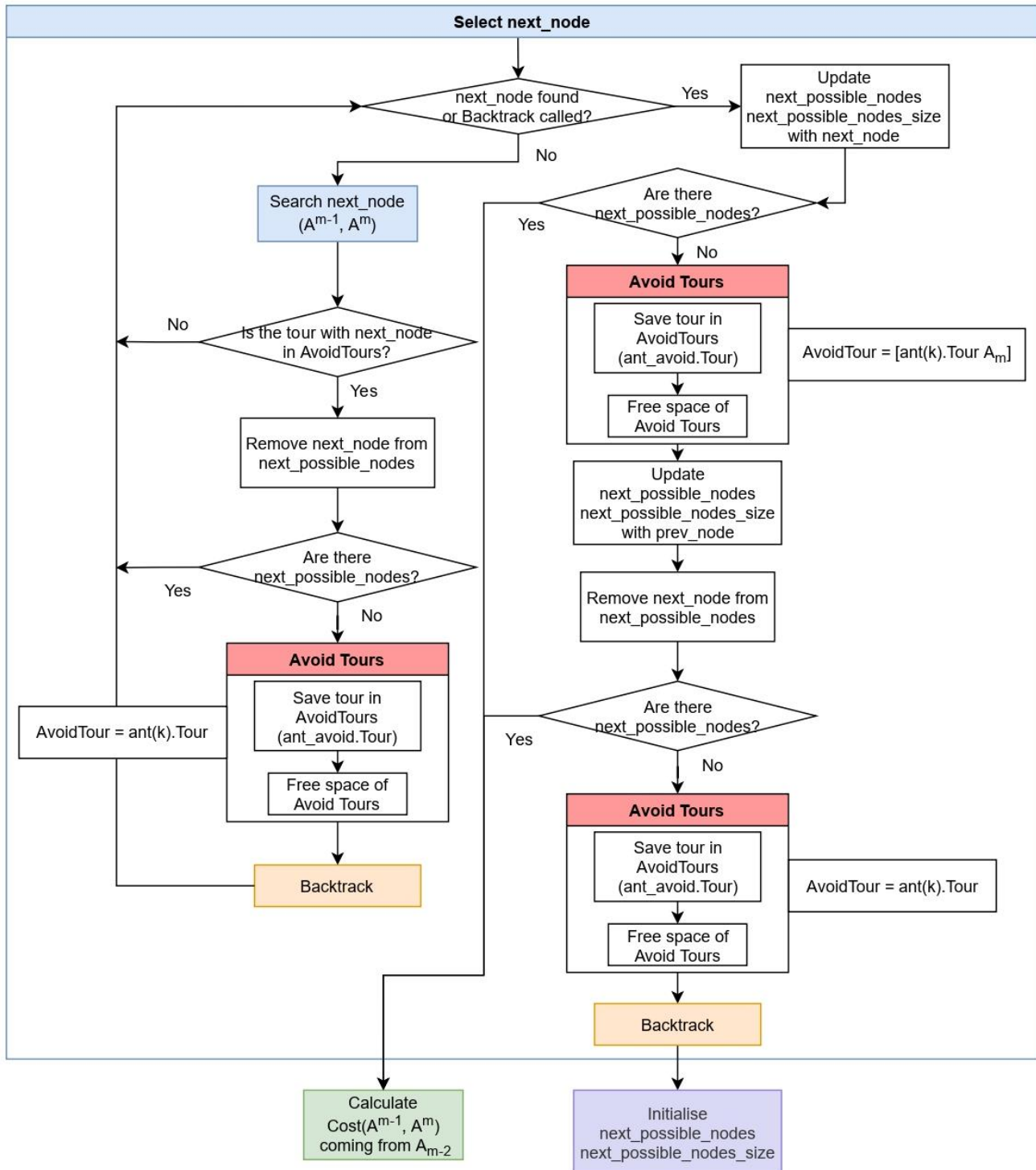
Algorithm 4-2 represents the implemented ACO algorithm main flowchart and Algorithm 4-3 represents the subprocess of selecting next node in the ACO. The main difference of the implemented algorithm with the standard one are the following:

- **Backtracking:** when an ant has selected a node that will not lead to a 12-asteroid sequence (REQ-001), the last asteroid is removed from the tour. A maximum of $n_{bt,max}$ backtracks is let per ant, if this number of removed asteroids is reached, the ant takes the longest tour encountered so far. For more details refer to
- **Avoid Tours:** all the candidate solutions found by the ants that are non-feasible because of REQ-002 are kept in each iteration and considered in the pheromone update.
- **Select next node:** for the selection of the next node, only feasible nodes in the cleaned Search Space \mathcal{G}_c are considered at each step.
- **Calculate Cost:** the algorithm uses the Score Matrix SM to reduce computational cost.
- **Pheromone parameter:** The pheromone is initialised null in all the non-feasible entries. In each iteration the pheromone of the nodes contained in Avoid Tours is decreased.

Algorithm 4-2. Implemented ACO



Algorithm 4-3. Implemented ACO: Select next node



4.5.2.1 Initialise parameters

The pheromone τ and the heuristic parameter η have the same dimensions as the Score Matrix SM , that relates the paths of all the Search Graph \mathcal{G}_c (see 4.2. Score Matrix). Recalling equation (4-4):

$$\tau = [\tau]_{n_{pairs,from} \times n_{ast}} \quad (4-9)$$

$$\eta = [\eta]_{n_{pairs,from} \times n_{ast}} \quad (4-10)$$

Due to the cleaning of the Search Space prior to calling the solver, the non-feasible solutions in the Score Matrix are translated into the pheromone initialisation as a 0-pheromone path. For the feasible candidates, the pheromone initialisation is set to the inverse of the mean average of the Score Matrix feasible entries, uniformly distributing the probability of choosing one node for the start of the solver.

$$\tau_{(i,j),k} = \begin{cases} 1/\overline{SM} & \text{if } \exists SM_{(i,j),k} \\ 0 & \text{otherwise} \end{cases} \quad (4-11)$$

The heuristic parameter is a constant matrix, whose entries are the inverse of the Score Matrix value for each asteroid triplet.

$$\eta_{(i,j),k} = \begin{cases} 1/SM_{(i,j),k} & \text{if } \exists SM_{(i,j),k} \\ \nexists \eta_{(i,j),k} & \text{otherwise} \end{cases} \quad (4-12)$$

$$\eta_{(i,j),k} = \begin{cases} \frac{1}{v \frac{SM_{(i,j),k}}{\Delta v_{leg,max}} + \iota \frac{\Delta ind_{jk}}{\Delta ind_{max}}} & \text{if } \exists SM_{(i,j),k} \\ \nexists \eta_{(i,j),k} & \text{otherwise} \end{cases}$$

4.5.2.2 Initialise population

The first population (i.e., iteration) candidate solutions of the algorithm are generated randomly. For the selection of each node (j, k) a list of possible feasible asteroids k from the previous node (i, j) is generated first. The selected asteroid is a uniformly distributed pseudorandom integer k . Backtracking is

implemented to ensure a sequence of 12 asteroids is completed and therefore, only feasible solutions are chosen from the Search Grid, $N(\xi) \subseteq \mathcal{G}_c$.

After the selection of the n_{ant} candidate solutions or tours ξ^a , the cost of each tour is summed up by accessing to each leg cost in the Score Matrix. The pheromone is updated then as per (4-7).

4.5.2.3 Let population explore – Initialisation

After the first population has randomly explored and deposited the information about the chosen tours, the rest of the populations are guided by the following steps.

Recalling Figure 4-2, it can be seen that all the candidate solutions start at the same point, A_0^0 , that corresponds to Earth. The first node is selected apart in the algorithm because the structure of it is of different dimensions, but the search of the node and the cost calculation follow the same steps as for the choice of the rest of the asteroids.

4.5.2.4 Let population explore – Construct a candidate solution

Once the first asteroid A^1 has been selected, the Score Matrix is accessed at the row corresponding to the previous node (A^0, A^1) and all the feasible next asteroids A^2 are kept in the variable *next_possible_nodes*.

The following process is repeated till the length of the tour is the desired one (12 asteroids L_{tour}^{ast} + Earth) or till a maximum number of backtrackings ($n_{bt,max}$). At position m of the sequence, the next asteroid A^m is searched, i.e., node (A^{m-1}, A^m). The selection of the next node is explained in detail in 4.5.2.6. Construct a candidate solution: Select next node. The cost of the path is given by the entry $((i, j), k)$ of the Score Matrix SM corresponding to the triplet $((A_i^{m-2}, A_j^{m-1}), A_k^m)$. The next feasible nodes A^{m+1} are searched and kept in *next_possible_nodes* and the best (lowest cost) tour so far is kept.

4.5.2.5 Pheromone update

The pheromone is updated after all the ants in the population have explored the Search Grid. The update is done as for (4-7) where $\Delta\tau_{(i,j),k}^a$ for this problem is given by (4-8), where the cost of the tour (length \mathcal{L}_ξ^a) is given by the sum of the Δv of all the legs that compose the tour of ant a (analogue to equation (4-2)):

$$\mathcal{L}_\xi^a = \Delta v_{A^0, A^1} + \sum_{m=2}^{\mathcal{L}_\xi^a} \Delta v_{A_j^{m-1}, A_k^m}^{A_i^{m-2}} \quad (4-13)$$

The pheromone is initialised 0 where the triplets are non-feasible, but during the running of the solver, some tours are kept as tours to avoid (see). After the pheromone has been updated, all the tours to avoid $\xi_{avoid} \notin N$ (*AvoidTours* in the flowcharts Algorithm 4-2 and Algorithm 4-3) are checked. If the length of the tour is 1 (only contains first asteroid) the pheromone is set to 0. Otherwise, the pheromone of each node in ξ_{avoid} is decreased in a distributed way along the tour.

$$\left. \begin{array}{l} \tau_{(i,j),k} = 0 \\ \tau_{(i,j),k} \leftarrow \frac{\tau_{(i,j),k}}{m} \cdot \frac{L_{\xi_{avoid}}^a}{L_{tour}^{ast}} \end{array} \right\} \forall ((i,j),k) \in \xi_{avoid} \text{ with } \begin{cases} L_{\xi_{avoid}}^a = 1 \\ L_{\xi_{avoid}}^a > 1 \end{cases} \quad (4-14)$$

The division of the length of the tour $L_{\xi_{avoid}}^a$ divided by the maximum length of tour ($L_{tour}^{ast} = 12$) makes the factor $\frac{L_{\xi_{avoid}}^a}{L_{tour}^{ast}} \in (0,1]$ a reduction factor. This makes the reduction of pheromone of shorter tours bigger. Moreover, the division by the position in the sequence m , makes the percentage of reduction bigger for the nodes at the end of the tour than at the beginning of it. Both factors allow to **reduce the probability of an ant choosing a shorter tour or a further node too promptly.**

4.5.2.6 Construct a candidate solution: Select next node

In order to select the next node, the ant is given the information about the next possible nodes and chooses one with a probability given by (4-6). Consider the ant has built the next sequence of asteroids:

$$(A_0^0, A_2^1, A_9^2, \underline{A_{23}^3, A_{56}^4})$$

The next possible nodes are all a pair that starts with previous asteroid, i.e., asteroid 56. The information is taken from the Score Matrix row corresponding to the previous pair of asteroids (A_{23}, A_{56}) (underlined in blue in the sequence).

For each of these asteroids, REQ-001 satisfaction is also checked by the following expression (from (4-1)):

$$A^m \leq n_{ast} - (L_{tour}^{ast} - m) \quad (4-15)$$

This condition would ensure reaching a sequence of 12 asteroids if the Search Space was complete. However, as it is filtered the application of this condition does not ensure the ants reaching to a 12-asteroid sequence.

Thus, for the pair (A_{23}^3, A_{56}^4), the next possible nodes are from asteroid 60 (because 57 to 59 do not comply with the Δv constraint REQ-001) to asteroid 150:

$$158 - (12 - 4) = 150$$

Table 4-3. Next possible nodes selection from SM. Example for pair (A_{23}, A_{56})

From \ To	A_1	...	A_{60}	A_{61}	...	A_{150}	A_{151}	...	A_{158}
(A_{23}, A_{56})	x	...	x	$\Delta v_{56,61}^{23}$...	$\Delta v_{56,150}^{23}$	$\Delta v_{56,151}^{23}$...	$\Delta v_{56,158}^{23}$

The roulette wheel or fitness proportionate selection method is used for selecting the following asteroid. It is based on a roulette wheel of the shape of a pie chart, whose areas are proportional to the relative fitness of each candidate, which corresponds to the dividend of the probability expression (4-6):

$$f_{(i,j),k} = \tau_{(i,j),k}^\alpha \cdot \eta_{(i,j),k}^\beta \quad (4-16)$$

The divisor of the probability expression is the sum of the relative fitness of all the candidate nodes:

$$F_{(i,j)} = \sum_{(A_i^{m-2}, A_j^{m-1}, A_l^m)} \tau_{(i,j),l}^\alpha \cdot \eta_{(i,j),l}^\beta \quad (4-17)$$

A uniformly distributed random number $r \in [0, F_{(i,j)})$ is generated. Then, the candidates are ordered in ascending relative fitness. The cumulative sum sequence $s_{(i,j),l}$ is calculated, and the first candidate k with a bigger relative fitness $s_{(i,j),k} > r$ is selected. Thus, the node (j, k) would be selected. A node with higher relative fitness occupies more area in the roulette wheel and thus, has a higher probability to get selected.

4.5.3 Multiobjective formulation of ACO

Multiobjective algorithms are used to build Pareto forms in problems with more than one confronted objective. In the analysed problem, a tendency of the ants towards the prompt selection of **asteroids with higher MOID time indices** was perceived. These asteroids have a smaller cost and thus, **are more attracted by the ants**. In order to counter balance this tendency, and the ants be able to reach a 12-asteroid sequence, the difference between asteroid MOID time indices needs to be taken into account. Thus, the confronted objectives in this case would be:

- Δv : Cost of the leg, related to REQ-002
- Δind : Difference between MOID time index of current asteroid and previous asteroid, related to REQ-001

However, to elaborate a bi-objective approach, an exhaustive enumeration of all the possible paths would be needed. With the aim of avoiding this, the algorithm is formulated as a **unique objective algorithm but with an alternative weighted pheromone calculation that considers the two objectives**.

4.5.3.1 ACO with index reward

Let the new cost of the node be the following weighted sum of individual objective costs:

$$\mathcal{L}_{(i,j),k}^a = v \frac{\mathcal{L}_{\xi}^a}{\Delta v_{max}} + \iota \frac{\Delta ind_{jk}}{\Delta ind_{max}} \quad (4-18)$$

where \mathcal{L}_{ξ}^a is the cost of the tour for the Δv objective given by (4-13) and $\Delta ind_{jk} = k - j$ is the cost for the Δind objective. Both costs are weighted by the parameters v and ι respectively. Leaving it as a function of Δv the following expression is obtained:

$$\mathcal{L}_{(i,j),k}^a = v \frac{\Delta v_{A^0, A^1} + \sum_{m=2}^{\iota_{\xi}^a} \Delta v_{A_j^{m-1}, A_k^m}^{A_i^{m-2}}}{\Delta v_{max}} + \iota \frac{\Delta ind_{jk}}{\Delta ind_{max}} \quad (4-19)$$

Note that now the cost is different for each node in the tour. Both terms are normalised with respect to the maximum value each objective cost can obtain. As well as the cost, related to the pheromone, the heuristic parameter is also changed to balance both objectives using the following expression:

$$\eta_{(i,j),k} = \begin{cases} \frac{1}{v \frac{SM_{(i,j),k}}{\Delta v_{max}} + \iota \frac{\Delta ind_{jk}}{\Delta ind_{max}}} & \text{if } \exists SM_{(i,j),k} \\ \nexists \eta_{(i,j),k} & \text{otherwise} \end{cases} \quad (4-20)$$

The maximum value of the Δv -cost of a tour is:

$$\mathcal{L}_{\xi}^a \leq \Delta v_{max} = L_{tour}^{ast} \cdot \Delta v_{leg,max} = 12 \cdot 9 \text{ km/s} = 108 \text{ km/s} \rightarrow \mathcal{L}_{max} = 108 \text{ km/s}$$

$$\Delta v_{max} = 108 \text{ km/s}$$

The maximum value of the Δind -cost of a node is:

$$\Delta ind_{max} = n_{ast} - L_{tour}^{ast} = 146$$

Giving the same weight to both objectives, the maximum Δv -cost is divided. Being the Δv -cost weight:

$$v = \frac{\mathcal{L}_{max}}{2} = 54$$

For the weight tuning, $n_{ast}/L_{tour}^{obj} = 158/12 \approx 13$ can be taken as an average value of Δind_{ij} . As the ants' tendency is to "jump" promptly to higher MOID time indices, more significance wants to be given to this term. Thus, ι is calculated taking into account the average value of Δind_{ij} , so that in average, the second summing term will equal $\mathcal{L}_{max}/2$ (but can be higher than this value):

$$\iota \frac{13}{\Delta ind_{max}} = \frac{\mathcal{L}_{max}}{2}$$

$$\iota \frac{13}{147} = 54 \rightarrow \iota \approx 611$$

5 SIMULATIONS

This section explains the approaches set-up parameters and test cases are presented, then the results analysis tools are explained and the computational tools to run the test cases are specified.

5.1 Approaches set-up

This section presents the different parameters set-up for comparison and best tuning of the solver. A table summary with all the values used can be seen in (ref). All the parameters that are not mentioned in this section have been tuned that proved to give to the best results of Tena and are considered to be sufficiently validated in his thesis [33].

5.1.1 Execution time and number of runs or rounds

In order to compare the performance of different solvers and configurations, all the solvers are limited to a maximum time of execution. It would be a better approach to limit the number of evaluations of each solver. However, due to the different nature of the heuristics and analytical tools, a fair comparison of the number of evaluations of the solvers is not possible. Thus, the solvers are limited in execution time.

Following Tena's previous work, each of the test cases are simulated in a total of 10 independent runs for each execution time of 10, 30 and 60 minutes [33].

5.1.2 Score Matrix cleaning threshold

As mentioned in 4.3. Cleaning of Search Space, the Score Matrix is cleaned, and therefore the Search Space pruned, by applying a threshold at each leg of 9 km/s. However, the best solver encountered so far in CASTPath was the called ACO *taboo* search, where the ant's last tour leg had a cost bigger than 1 km/s, the tour was completed by adding very high cost nodes, not allowing it to be a candidate solution (see Tena [33] for more information).

It has been reasoned that the cleaning threshold is set at the one imposed by the total Δv requirement REQ-002 to avoid the omission of feasible solutions

with a total Δv smaller than 9 km/s which could have one leg of high cost and the other 10 legs of significantly low Δv . Therefore, it seems reasonable to test the solver with:

- **No filtering**: use of the Search Grid \mathcal{G} to prove the importance of pruning the Search Space prior to the solver execution,
- A filter implemented by a leg cost **threshold of 9 km/s**: the Search Space will be reduced to the cleaned Search Grid denoted as $\mathcal{G}_{c,9}$, and
- A filter implemented by a leg cost **threshold of 1 km/s**: the Search Space will be even more reduced to the cleaned Search Grid denoted as $\mathcal{G}_{c,1}$, in order to compare with previous CASTPath-ACO approaches.

5.1.3 ACO parameters

The different test cases will be run with different ACO parameters to tune the solver with the best parameters. This tuning will also allow to understand the nature of the Search Space, which will be discussed in 7. Discussion.

5.1.3.1 Heuristic importance and trail dependency

Previous ACO CASTPath analysis has proven that the more weight is given to the heuristic parameter η (with exponential weight β) compared to the pheromone τ weight (exponential weight α) in the expression (4-6), the better the ACO solver performs [33]. A sensitivity analysis is repeated in this thesis for the α and β parameters in order to validate this theory independently of the ACO approach.

A sensitivity analysis with the following cases will be tested:

- $\alpha = 1$ and $\beta = 1$
- $\alpha = 1$ and $\beta = 4$
- $\alpha = 1$ and $\beta = 5$
- $\alpha = 0$ and $\beta = 1$

The first two cases were implemented by Tena, for the analysis of the intermediate cases see his thesis [33]. The last two are extreme cases to verify

the importance of the heuristic parameter over the pheromone. Finally, comparing $\beta = 4$ (Tena's best solution tuning) with $\beta = 5$ (not tested by Tena), aims to give an idea of how much the solutions change by a unitary increment of this parameter.

5.1.3.2 Bi-objective cost: index reward

After some tests run with the implementation described in 4.5.2. Implemented ACO algorithm, a prompt selection of asteroids of high MOID time ID was detected. The ants struggled to find feasible solutions that complied with REQ-001 (12-asteroid tours) giving solutions of 9 asteroids as most. This is the reason why the bi-objective approach applied to the pheromone and heuristic was modelled (see 4.5.3. Multiobjective formulation of ACO).

The cost weight parameters are tuned so that it is given more importance to the Δ_{ind} objective than the Δv . In order to understand the objectives influence, the test cases will be the tuned parameters and equalling the parameters:

- $\nu = 54$ and $\iota = 611$ (tuned parameters)
- $\nu = 54$ and $\iota = 54$

Note that the case $\nu = 54$ and $\iota = 0$ has already been implemented in the test runs prior to the bi-objective cost approach.

5.1.3.3 Backtracking

A maximum number of 50 backtracks per ant has been implemented as stop criterium after balancing the execution time of the solver and the solutions encountered. This number had to be tuned while running the solver for the ants to not to get lost in the Search Grid when no more feasible nodes can be found.

5.2 Test cases

After analysing the approaches set-up, the test cases will be the following:

1. Score Matrix (SM) validation test
2. Whole Search: SM not pruned
 - a. $\beta = 1$

- b. $\beta = 5$
- 3. Score Matrix filtered at 9 km/s: $SM \leq 9 \text{ km/s}$
 - a. $\iota = 54$
 - b. $\iota = 611$
- 4. Score Matrix filtered at 1 km/s: $SM \leq 1 \text{ km/s}$
 - a. $\beta = 4 (\iota = 54)$
 - b. $\beta = 5 (\iota = 54)$
 - c. $\alpha = 0$
 - d. $\iota = 611 (\beta = 5)$

5.3 Analysis tools

Ten independent runs for each test case will be executed and statistically analysed by the tools presented in this section. Boxplots and stacked bar graphs will be used to analyse the quantity of feasible asteroid sequences encountered and the quality (score) of those solutions. The quality of the solutions is even more analysed in terms of diversity as for the different asteroids included in the solutions by means of the similarity measure tool developed in this thesis.

5.3.1 Box and Whiskers Plot

Boxplots are standardised statistical distribution graphs for groups of data that give information about the variability of the data. A boxplot consists of a box with an Interquartile Range (IQR) containing three quartiles: lower or first quartile (Q_1), median or second quartile (Q_2), upper or third quartile (Q_3). It can also include *whiskers*, two lines outside the box that indicate the dispersion of the data outside the IQR, with the “minimum” (Q_0) and “maximum” (Q_4) values. Note that these values are not really the global maximum and minimum values of the data set, which will be called highest and lowest data points instead. In summary, the elements of a box and whiskers plot are [42]:

- “Minimum” (Q_0 or 0th percentile): the lowest value excluding outliers, $minimum = Q_1 - 1.5 \cdot IQR$.





- Lower quartile (Q_1 or 25th percentile): the middle number between the lowest value (not the “minimum”) and the median of the dataset.
- Median or second quartile (Q_2 or 50th percentile): the middle value of the dataset.
- Upper quartile (Q_3 or 75th percentile): the middle value between the median and the highest value (not the “maximum”) of the dataset.
- “Maximum” (Q_4 or 100th percentile): the highest value excluding outliers, $maximum = Q_3 + 1.5 \cdot IQR$.
- Outliers: 0.7% of the data, in a normal distribution, the 0.35% at both edges of the plot, that include the lowest and highest data points in the dataset.
- Interquartile Range (IQR): 25th to the 75th percentile. $IQR = Q_3 - Q_1$
- Whiskers: data dispersion outside the IQR between the “minimum” and “maximum” (not the lowest and highest values).

In this thesis, the built-in MATLAB function `boxplot` will be used for the analysis of the different runs for each test case. This plot will allow to analyse whether the solutions obtained have been obtained by chance or the algorithm is robust within the 10 different simulations. The diagrams include the total number of sequences below a certain threshold in the right-hand vertical axis illustrated by the symbol \triangleleft in the graph.

5.3.2 Stacked Bar Graph

A stacked bar graph is a graph used to compare different series of data. A bar per series is used, divided into different sub-elements to show the fraction of each on the total of the series (the bar). In the problem in hand, it will be used to compare 3 series: 10 min, 30 min and 60 min execution times, each of those being fractioned into four different thresholds. Each fraction will include the number of sequences in percentage that fall in that range. Note that some of the cases, as in the Whole Search and $SM \leq 9km/s$ the y axis is in logarithmic scale as the fractions are too small to be seen at linear scale.

The chosen thresholds (except for the Whole Search case) are:

	$\Delta v \leq 8km/s$	exceeds REQ-002
	$8 < \Delta v \leq 9km/s$	meets REQ-002
	$9 < \Delta v \leq 12km/s$	does not meet REQ-002
	$\Delta v > 12km/s$	does not meet REQ-002

Where the first two will be represented in blue as they fulfil REQ-002, and the last two in red as these are not feasible solutions because of the Δv_{max} constraints imposed. The intermediate threshold of 12 km/s is chosen for the non-feasible solutions because there seem to be several solutions in the third threshold.

5.3.3 Similarity analysis

The previous both tools are common statistical graphs used within the mathematics and statistics community. However, a tool to understand and measure the diversity in the solutions is needed as per the scientific interest on the CASTAway asteroid tour. The Similarity measure tool developed and introduced in 4.4. Similarity Measure can be used to first, analyse the diversity the proposed ACO solver gives within its solutions and evaluate the goodness of the set of the solutions by their variability; and second, to compare the easiness of the heuristic to search more broadly within the Search Space, compared to other solvers used in CASTPath.

5.4 Computational tools

All the small tests and analysis of the data were executed on an Intel® Core™ i7-8565U CPU @ 1.80GHz, windows 10 Home 64 bit (10.0) and no multitasking. For the full-scale run cases, Cranfield University's *Crescent High Performance Computer Cluster*, or HPC, has been used. In particular, the simulations were run in two Intel E5-2660 (Sandy Bridge) CPUs giving 16 CPU cores and 64GB of shared memory. Instructions on how to use this Cranfield University service can be found on the University's intranet or more in detail, in Tena [33].

6 RESULTS

This section presents the results, analysis and discussion of the developed solver through different test cases summarised in 5.2. Test cases. A further discussion on the results can be found in the next section 7. Discussion.

6.1 Score Matrix validation test

Table 6-1 presents a test performed to prove the cost uniqueness by the proposed modelling of the Search Space. 10 different sequences were generated randomly (same function as the first colony uses in ACO, see section 4.5.2.2) and their total costs were calculated by accessing triplet by triplet to the Score Matrix (Δv_{SM}) and by calling the function in the CASTPath Toolbox wrapper_combina_CASTPath_MARSSW.m (Δv_{fun}). Results show the values obtained are exactly the same and therefore it can be said that the **uniqueness of the paths cost** in the Search Space \mathcal{G} has been **proved**. Note that all the decimals given by MATLAB are shown to prove null error.

Table 6-1. Test to prove the cost uniqueness with the implementation of the SM and comparison on computational time for using the SM and the wrapper function tool

10 randomly generated asteroid sequences ($SM \leq 9km/s$)													Δv_{SM}	Δv_{fun}
0	11	22	41	61	71	83	92	101	119	127	139	158	13.06614441	13.06614441
0	11	24	39	61	83	97	109	121	127	139	152	158	13.22455344	13.22455344
0	2	17	36	50	86	106	118	129	135	141	152	158	10.14910253	10.14910253
0	2	26	46	54	73	92	97	106	120	131	141	154	11.55330089	11.55330089
0	10	35	51	57	85	92	109	120	135	141	152	158	10.68620602	10.68620602
0	13	22	36	41	56	74	97	106	117	122	134	148	12.93202176	12.93202176
0	5	10	17	35	61	86	97	106	129	135	141	157	11.28897666	11.28897666
0	2	13	31	44	59	72	92	110	115	135	141	158	11.78296788	11.78296788
0	8	27	64	80	94	105	118	121	127	139	152	158	12.73494301	12.73494301
0	9	19	35	51	62	92	106	118	129	141	152	158	11.60452448	11.60452448

6.2 Whole Search

For the whole search case, the Δ_{ind} objective weight was set as null, $\iota = 0$, as in this case the Search Space is so extended that the ants do not struggle to find sequences of 12 asteroids. For the same reason, the backtracking is not implemented as the ants always have nodes to continue their sequence. The solver is initialised by a first random colony that gives 12 asteroid sequences whose legs have a cost of less than 9 km/s ($SM \leq 9 \text{ km/s}$ is used).

Figure 6-1 shows the best Δv at each iteration for the 10 rounds of 30 minutes and 60 minutes, for the following two cases of the heuristic parameter weight: $\beta = 1$ and $\beta = 5$. It can be appreciated that, in general, the solver's best solutions are the ones of the first colony.

For the case of 10 minutes, the solver does not have enough time to improve the best Δv given by the first colony. For the case of 30 minutes round 3 achieves to improve the best Δv , and for the case of 60 minutes round 3 and 7 also achieve an improvement from the 1st colony. It is remarkable the improvement in the case of round 7 (in red) that decreases the best Δv from 29.7118 km/s to 24.3152 km/s. However, this is the round that also starts with the highest best Δv .

Comparing the two cases of β , the number of iterations for both cases seem to remain the same. The improvements on the best Δv are the three for the case of $\beta = 1$. However, these are isolated cases and the boxplots (Figure 6-2 for $\beta = 1$ and Figure 6-3 for $\beta = 5$) and stacked bar graph (Figure 6-4) need to be analysed to compare properly the heuristic parameter influence. Note that, as an exception for the rest of the results presented in the following sections, the right axis in the boxplots indicate the number of solutions below a threshold of 30 km/s (instead of 9 km/s) and the stacked bar graphs thresholds are also changed by higher values (all of them do not meet REQ-002). This is done because the **Whole Search** case, as expected, **gives very high values of Δv** .

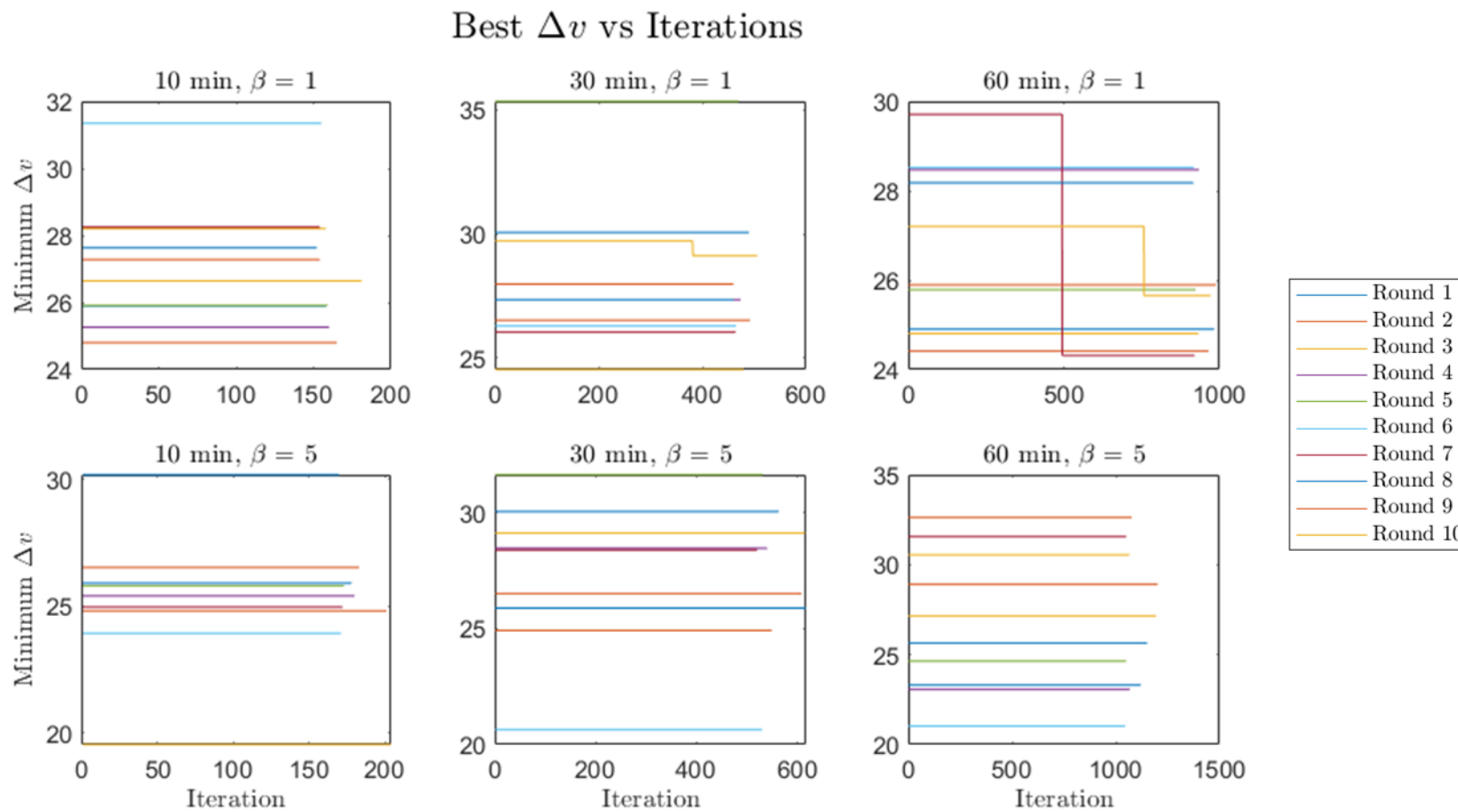


Figure 6-1. Best Δv achieved so far in each iteration in the Whole Search for 10 mins, 30 mins and 60 mins, for $\beta = 1$ and $\beta = 5$

In general, the median and “minimum” values of the $\beta = 5$ boxplot are lower than the ones of $\beta = 1$. It is also noticed that in the case of $\beta = 5$, the ants find more lower outliers, achieving better best- Δv in each series. The execution time does not seem to improve much the values in both cases. In fact, the best- Δv in the case of $\beta = 5$ is achieved for the 10 min execution time series. Moreover, the number of sequences encountered with a maximum threshold of 30 km/s (in orange) does not seem to be related to the execution time or the β value.

The stacked bar graph shows that almost 100% of all the solutions are above the 40 km/s threshold. Note that because of this, the plot’s y axis is in logarithmic scale. This was expected as the Score Matrix is not pruned and there exist very high Δv in the Search Space. All of this implies that **the performance of the solver in the Whole Search case is strongly dependent on the goodness of the first random colony solutions.**

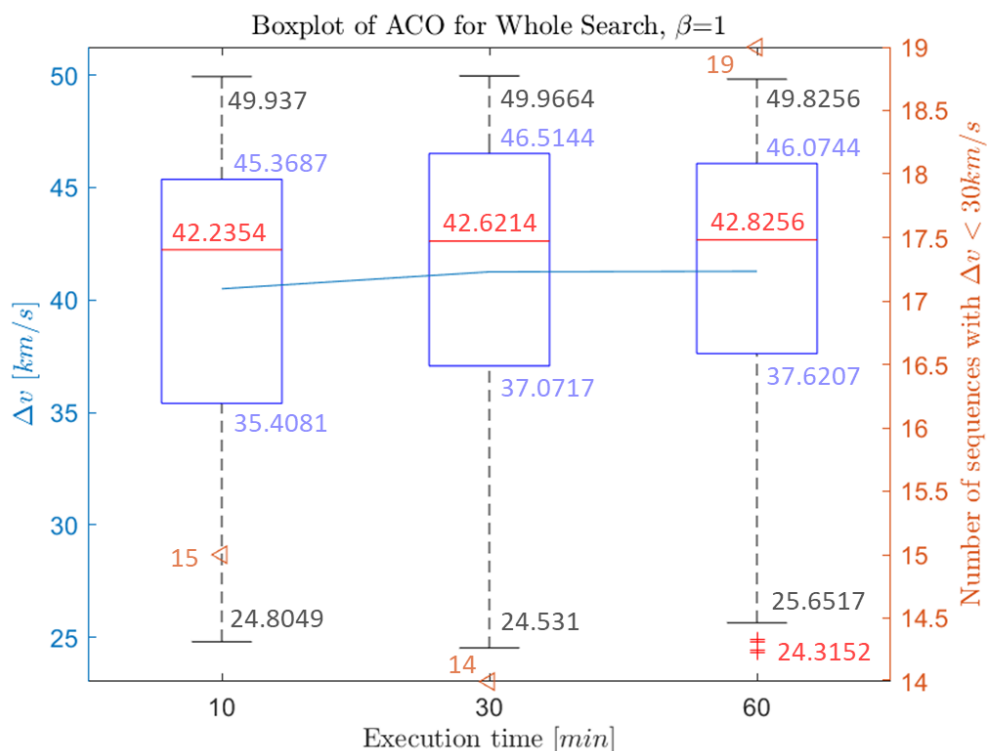


Figure 6-2. Boxplot of results of ACO for Whole Search and $\beta = 1$

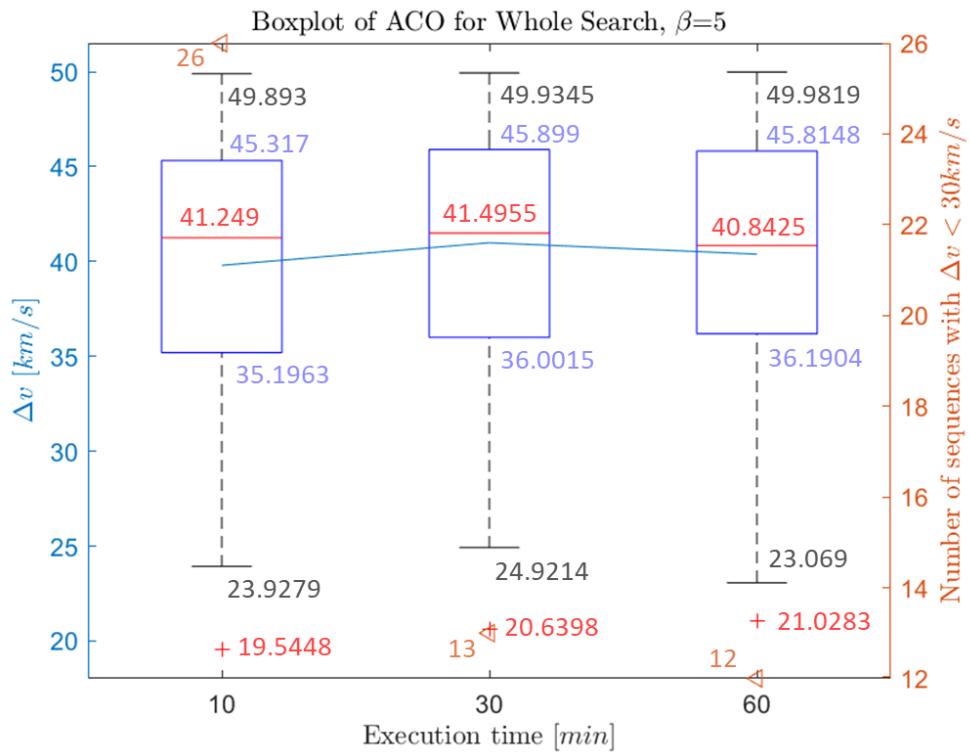


Figure 6-3. Boxplot of results of ACO for Whole Search and $\beta = 5$

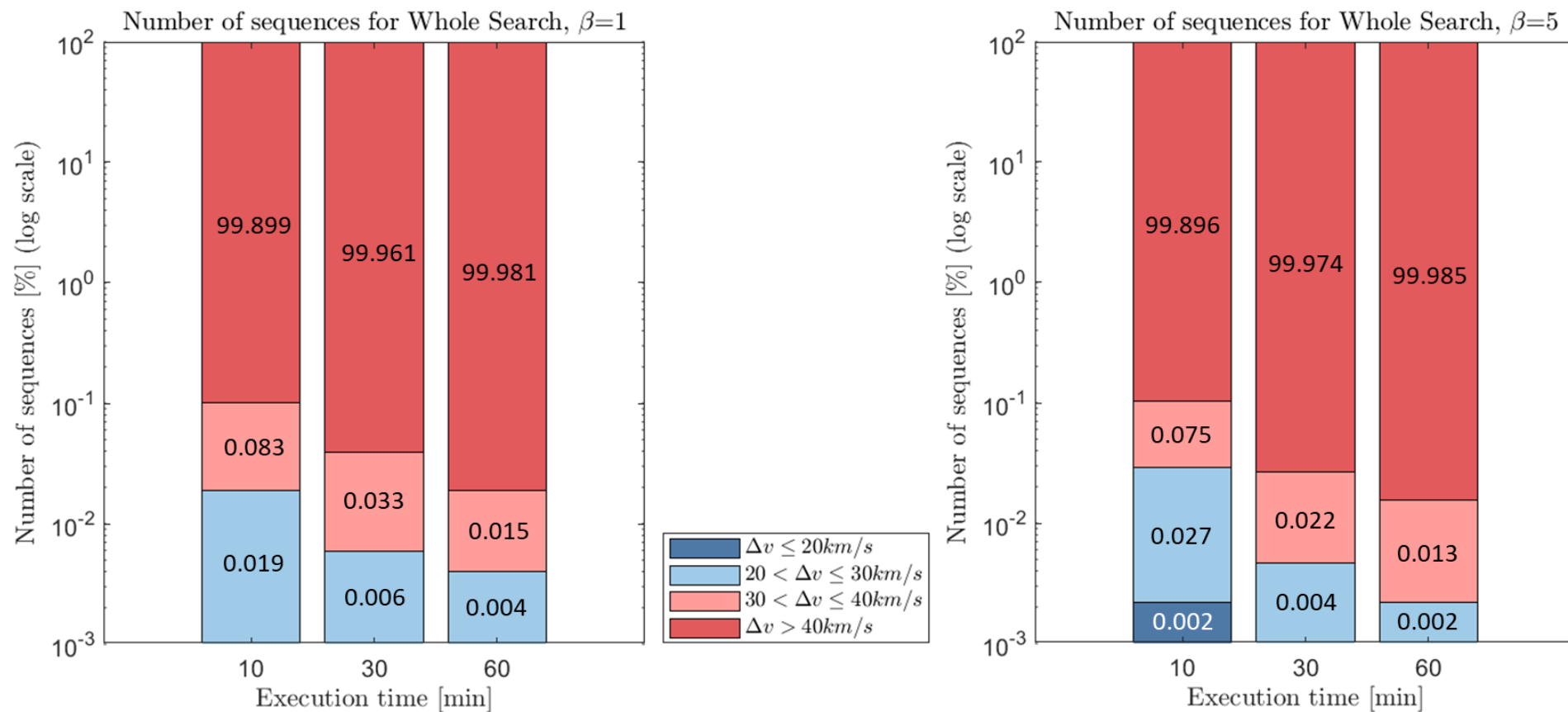


Figure 6-4. Stacked Bar Graph of results of ACO for Whole Search, $\beta = 1$ (left) and $\beta = 5$ (right)

Table 6-2 and Table 6-3 show the best results for the Whole Search case for $\beta = 1$ and $\beta = 5$, respectively. Bear in mind that these best solutions seem to be strongly dependent on the goodness of the score of the first random colony. Nevertheless, it seems that as Tena already proved, a higher heuristic parameter weight β tuned ACO performs better for the CASTAway problem. This fact will be more analysed in 6.4.1. Comparison of $\beta = 4$ and $\beta = 5$ implementing a Score Matrix cleaning.

Table 6-2. Parameters and results of ACO solver for Whole Search and $\beta = 1$

Score Matrix threshold	None (Whole Search)		
Execution time	60 min (best)	30 min	10 min
ACO parameters	$\alpha = 1, \beta = 1, \nu = 54, \iota = 0$		
Number of feasible solutions	-	-	-
Best tour cost Δv	24.3152 km/s		
Best tour sequence (MOID time IDs)	[3 9 62 92 107 111 126 137 142 147 152 158]		
Best tour sequence (Asteroid IDs)	[75773 54602 81229 4876 65326 80662 42999 90769 77579 58896 72195 85792]		

Table 6-3. Parameters and results of ACO solver for Whole Search and $\beta = 5$

Score Matrix threshold	None (Whole Search)		
Execution time	60 min	30 min	10 min (best)
ACO parameters	$\alpha = 1, \beta = 5, \nu = 54, \iota = 0$		
Number of feasible solutions	-	-	-
Best tour cost Δv	19.5448 km/s		
Best tour sequence (MOID time IDs)	[14 39 63 83 97 116 118 131 137 142 147 155]		
Best tour sequence (Asteroid IDs)	[39775 15032 95771 25689 50239 92672 7758 44231 90769 77579 58896 29113]		

6.3 Score Matrix filtered at 9 km/s

Figure 6-5 presents the best Δv found by the ants so far at each iteration for the three different execution times (run in 10 independent rounds each) and for the two test cases of $\iota = 54$ and $\iota = 611$. Note that this diagram is presented to show the tendency of the solver to improve the best Δv , to see concrete values refer to the boxplot, Figure 6-6.

It can be seen that in all the cases, the initial Δv costs are higher, while in the case of $\iota = 611$, not only the costs start at a lower value but the tendency to improve the best Δv is also quicker. **The number of iterations are higher when the Δind cost weight ι is bigger**, which means that the ants need to **backtrack less times**, finding more easily solutions of 12 asteroids.

The case of $\iota = 54$ achieves a better best Δv . However, the boxplots (in Figure 6-6 for $\iota = 54$ and in Figure 6-7 for $\iota = 611$) reveal this value is an outlier, and the quartiles are higher in the case of $\iota = 54$ than in the case of $\iota = 611$. Looking into detail the boxplot for $\iota = 54$, a tendency to find some good outliers below the “minimum” is seen. In fact this case achieves a lower Δv than $\iota = 611$. This is probably because the ants can still find good solutions even if the Δind is higher, which can mean that **increasing the Δind importance gives penalty to differentiated indices that can give good results** and still lead to 12 asteroid sequences. However, it is clearly more difficult for them to find higher amounts of feasible solutions with a lower ι . With $\iota = 54$ only for 60 min execution time 2 feasible sequences are found (null in the other two series), while with $\iota = 611$ 5, 19 and 56 feasible solutions are encountered for each series of 10, 30 and 60 minutes respectively.

Analysing the stacked bar graph for both cases in Figure 6-8 (note that the y axis is again in logarithmic scale), it can be seen that **a higher ι gives a higher percentage of feasible sequences** ($\Delta v \leq 9 \text{ km/s}$, in blue) and a lower percentage of the worst threshold range of $\Delta v > 12 \text{ km/s}$.

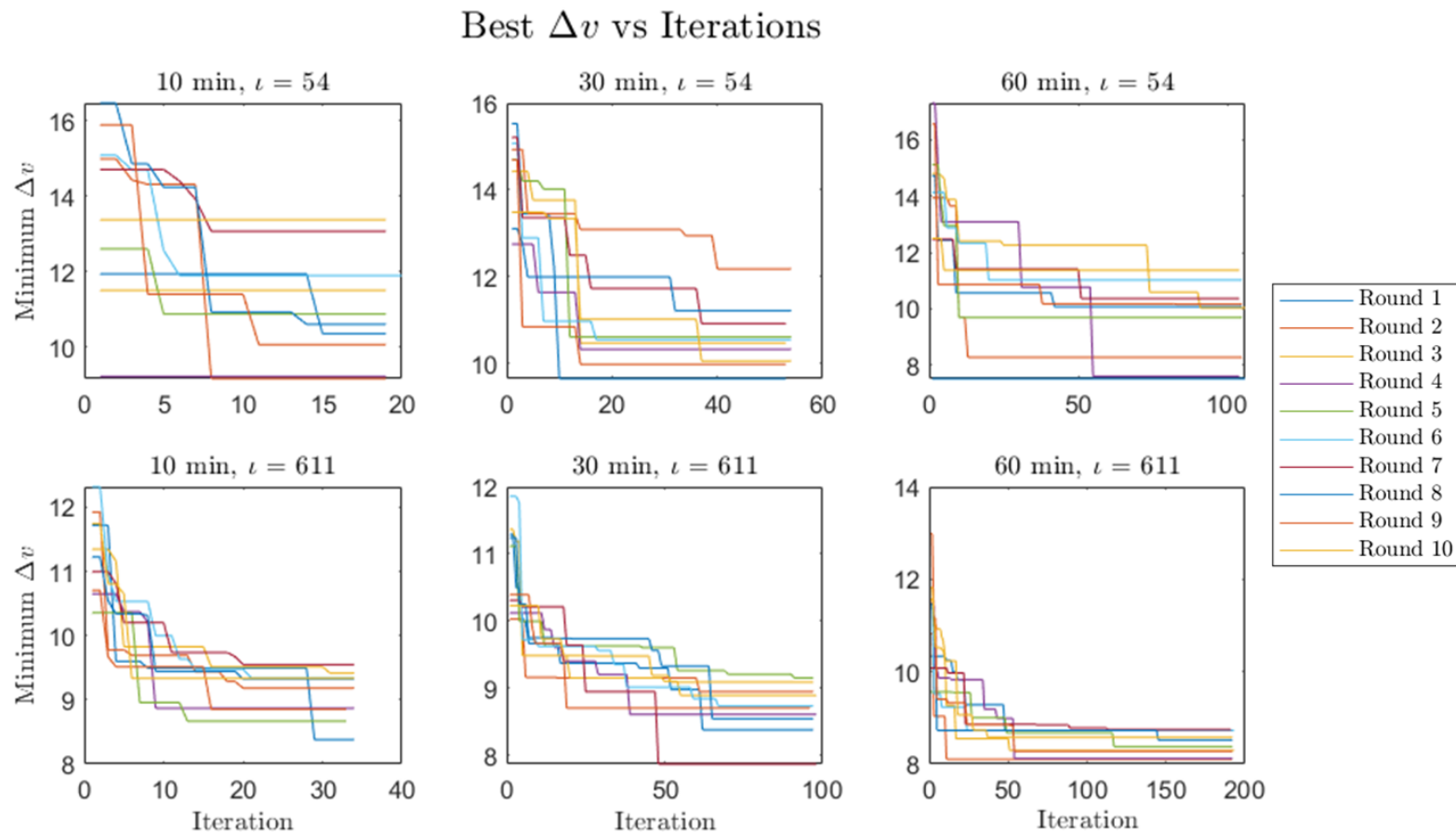


Figure 6-5. Best Δv achieved so far in each iteration of ACO with $SM \leq 9$ km/s for 10 mins, 30 mins and 60 mins, for $\ell = 54$ and $\ell = 611$

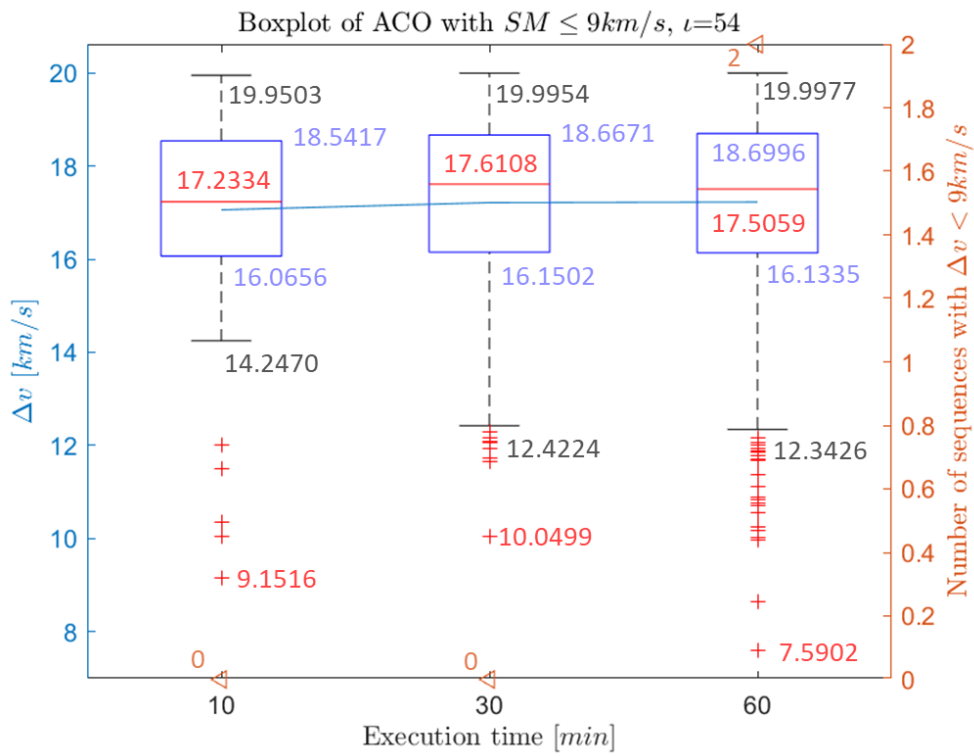


Figure 6-6. Boxplot of results of ACO with $SM \leq 9 \text{ km/s}$ and $\iota = 54$

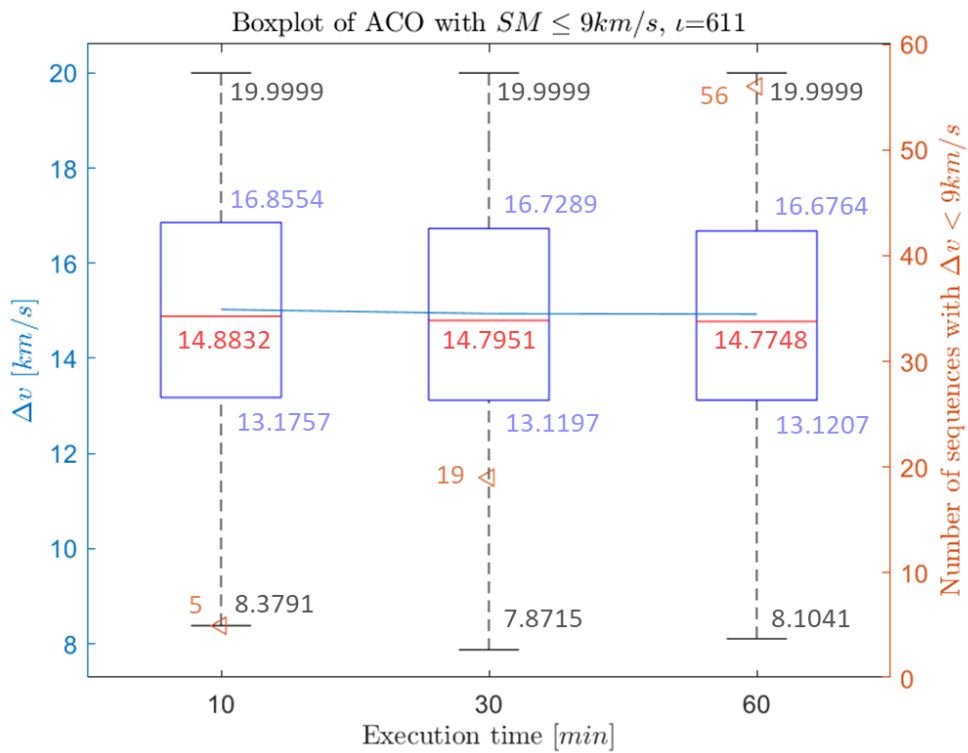


Figure 6-7. Boxplot of results of ACO with $SM \leq 9 \text{ km/s}$ and $\iota = 611$

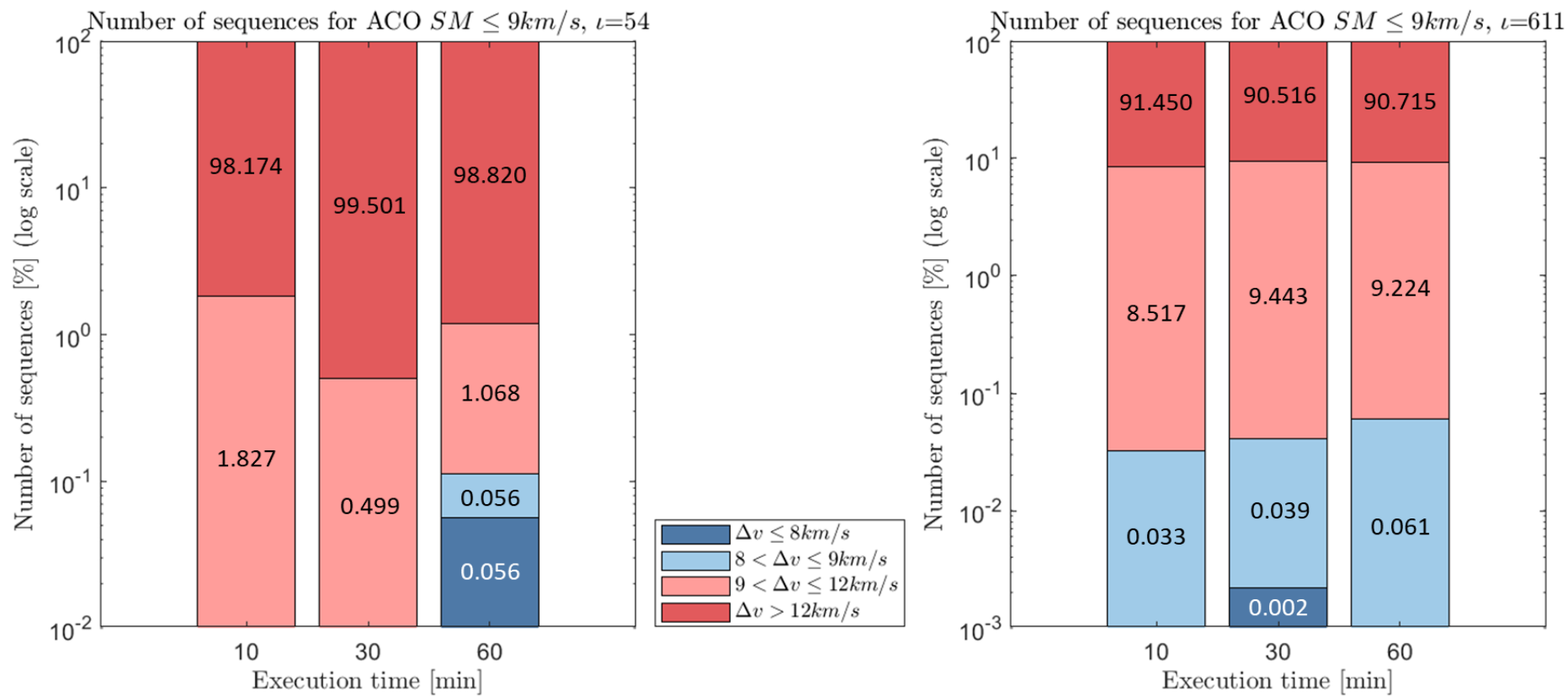


Figure 6-8. Stacked Bar Graph of results of ACO with $SM \leq 9 km/s$ and $\iota = 54$ (left) and $\iota = 611$ (right). Note that the y axis is in logarithmic scale.

Table 6-4 and Table 6-5 present a summary of the best Δv achieved in all the runs for the proposed ACO solver with $SM \leq 9 \text{ km/s}$ and $\iota = 54$ and $\iota = 611$, respectively. There is need to highlight that even if the best tour cost is achieved by $\iota = 54$, $\iota = 611$ achieves many more feasible solutions.

Table 6-4. Parameters and results of ACO solver with $SM \leq 9 \text{ km/s}$ and $\iota = 54$

Score Matrix threshold	9 km/s		
Execution time	60 min (best)	30 min	10 min
ACO parameters	$\alpha = 1, \beta = 5, \nu = 54, \iota = 54$		
Number of feasible solutions	2	-	-
Best tour cost Δv	7.5902 km/s		
Best tour sequence (MOID time IDs)	[6 10 17 35 43 53 99 118 129 135 141 152]		
Best tour sequence (Asteroid IDs)	[37079 7080 56848 42460 79766 29568 60318 7758 80319 88041 82192 72195]		

Table 6-5. Parameters and results of ACO solver with $SM \leq 9 \text{ km/s}$ and $\iota = 611$

Score Matrix threshold	9 km/s		
Execution time	60 min	30 min (best)	10 min
ACO parameters	$\alpha = 1, \beta = 5, \nu = 54, \iota = 611$		
Number of feasible solutions	56	19	5
Best tour cost Δv	7.8715 km/s		
Best tour sequence (MOID time IDs)	[6 10 17 25 33 62 75 102 129 135 141 154]		
Best tour sequence (Asteroid IDs)	[37079 7080 56848 70155 5233 81229 69928 46 80319 88041 82192 59305]		

6.4 Score Matrix filtered at 1 km/s

AS introduced in 5.1.2. Score Matrix cleaning threshold, the last pruning of the Search Space is done for a $SM \leq 1km/s$ following the work done by Tena on his ACO-Taboo solver [33]. First, in 6.4.1. Comparison of $\beta = 4$ and $\beta = 5$ a comparison of the heuristic parameter weight β is done by applying Tena's best tuning parameter ($\beta = 4$) and increasing this value by one unit ($\beta = 5$). The rest of the parameters are $\alpha = 1$, $\nu = 54$ and $\iota = 54$.

As presented in the next section and seen in the Whole Search tests, a higher β performs better. Because of this observation, the trail-dependency of the tour of asteroids (the effectiveness of the pheromone) wants to be checked by removing completely the contribution of the pheromone making $\alpha = 0$. The rest of parameters will be $\beta = 1$ (increasing here β makes no improvement as the heuristic parameter will be the only one contributing to the probability, see (4-6)), $\nu = 54$ and $\iota = 54$.

Finally, as the tests with a $SM \leq 9km/s$ comparing $\iota = 54$ and $\iota = 611$ show that a higher weight of the Δ_{ind} cost objective performs better, a test with $\iota = 611$ with the best $\beta = 5$ is done, being the rest of parameters $\alpha = 1$ and $\nu = 54$.

6.4.1 Comparison of $\beta = 4$ and $\beta = 5$

Figure 6-9 and Figure 6-10 show the boxplots of the results of Δv (in the left axis) of the proposed ACO solver with $SM \leq 1km/s$ for $\beta = 4$ and $\beta = 5$, respectively, along with the number of feasible (as per REQ-002) sequences (in the right axis, in orange, represented by $\langle \rangle$).

The first observation made to both plots is that the tendency of the medians (in red) is to increase from 10 min to 30 min execution time, and then decrease from 30 min to 60 min but with a higher value compared to the median of the 10 min series in both cases. It seems that the ants start with very good solutions thanks to the heuristic parameter information (as the pheromone is initialised equally throughout all the Search Space) even if a random first colony is not executed in this case. Then, the ants explore more widely within the Search

Space encountering a wide range of solutions with better and worse Δv (30 min) and need a bit more of time to finally find several feasible solutions with lower Δv costs (60 min).

In both configurations, the ants can find feasible solutions rapidly in 10 min (7 for $\beta = 4$ and 12 for $\beta = 5$), and the growth of the feasible solutions increases rapidly with the execution time as it achieves more than the double or even more than the triple, x2.35 in the case of $\beta = 4$, and x3.46 in the case of $\beta = 5$, of feasible results in the double of time (60 and 30 mins). This fact leads to think that the ants learn more when the heuristic parameter importance is higher.

Comparing both boxplots, it can be seen that the higher heuristic parameter weight $\beta = 5$ gives better lower quartile values (lower Δv costs) and is able to find 2.43 times more feasible solutions (in 60 mins) than the case with $\beta = 4$. In this vein, the stacked bar graph in Figure 6-11 shows that the $\beta = 5$ case gives more feasible solutions not only in quantity but also in fraction with respect to the non-feasible ones (in red). It needs to be highlighted that **more than the half of $\beta = 5$ solutions** for 10 and 60 minutes of execution time **are feasible** (57.14% and 56.73% respectively).

It is also worth mentioning that for both cases, and only for one series in each of them, only a 1.33% of the solutions scores are higher than 12 km/s. Even if the Score Matrix threshold is established in 1 km/s per leg, so for 12 asteroids the maximum would be 12 km/s, the first A^0, A^1 leg (Earth-first asteroid) also needs to be considered which adds an average of 4.5544 km/s (the maximum value is 6.2889 km/s for leg A_0^0, A_7^1). Thus, the maximum possible value of the tour for $SM \leq 1km/s$ is theoretically 18.2889 km/s.

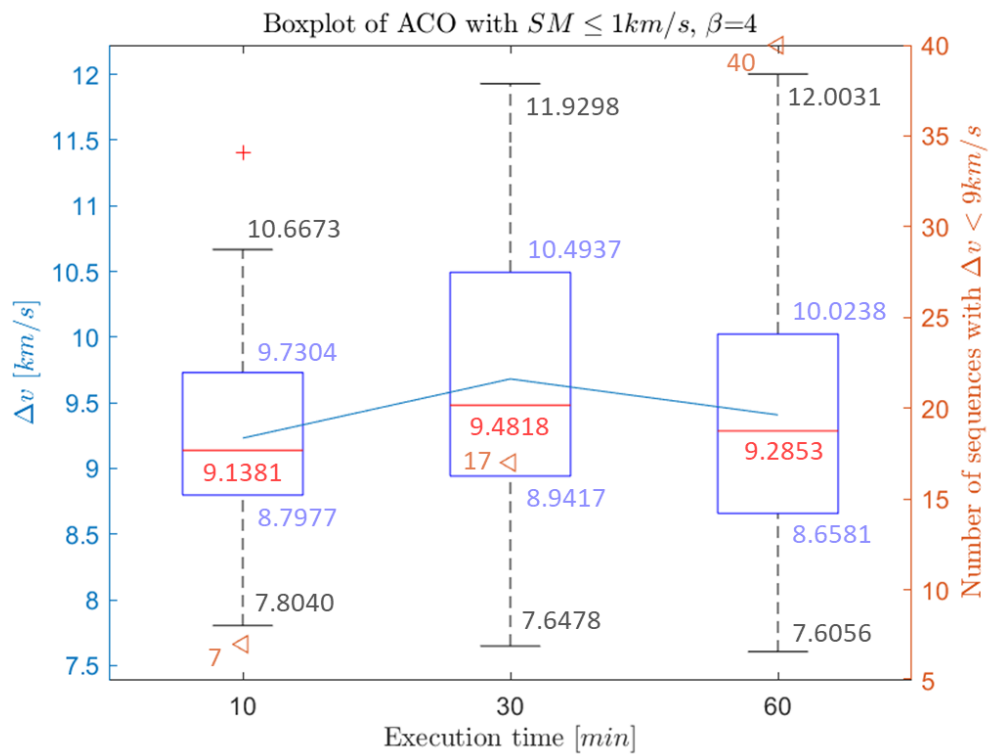


Figure 6-9. Boxplot of results of ACO with $SM \leq 1 \text{ km/s}$ and $\beta = 4$

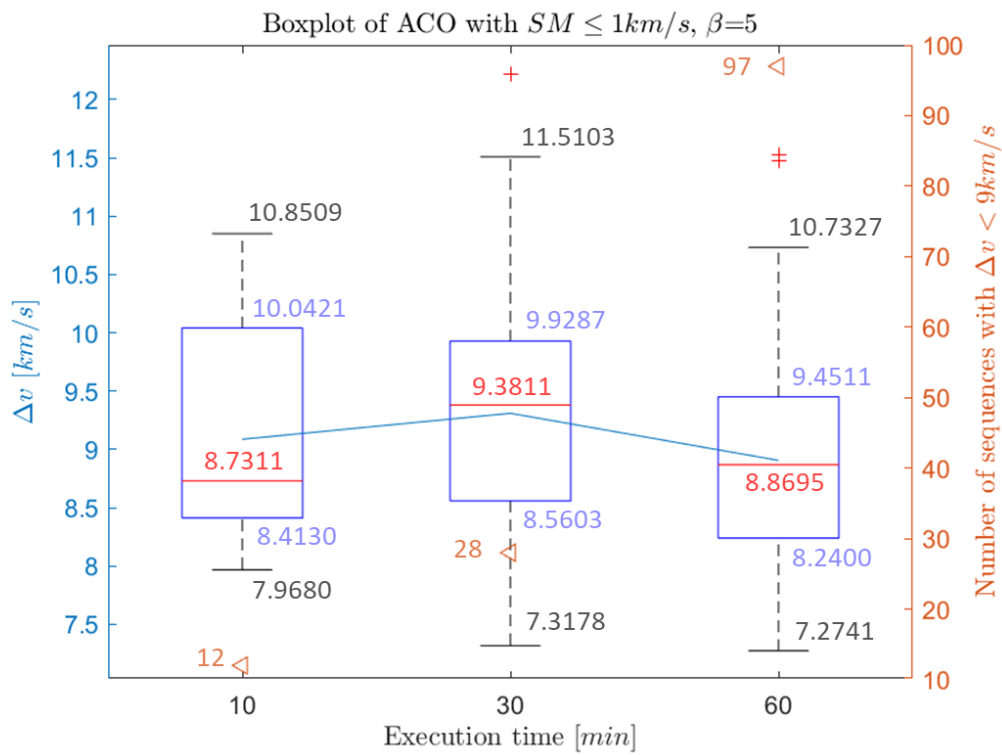


Figure 6-10. Boxplot of results of ACO with $SM \leq 1 \text{ km/s}$ and $\beta = 5$

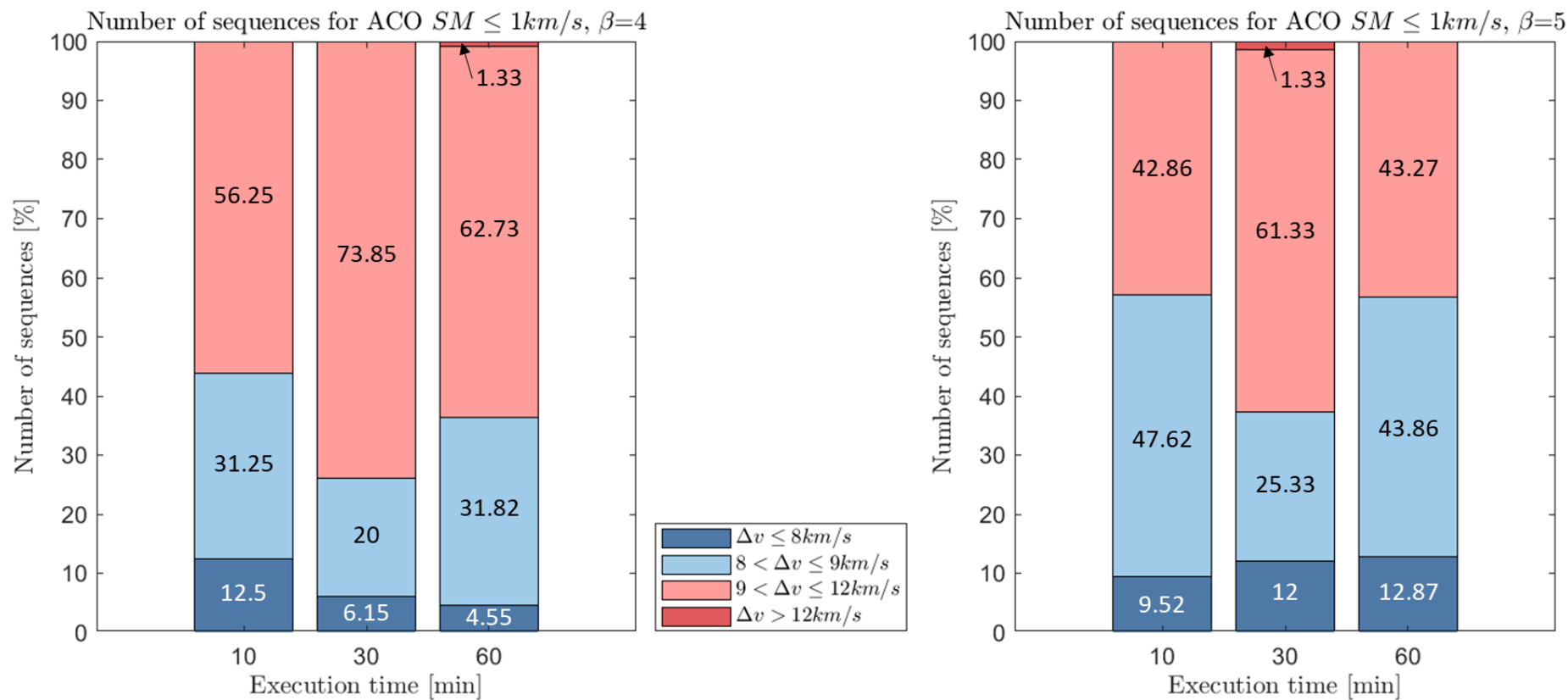


Figure 6-11. Stacked Bar Graph of results of ACO with $SM \leq 1 km/s$ and $\beta = 4$ (left) and $\beta = 5$ (right)

Table 6-6 presents the configuration, number of feasible solutions and best solution for the ACO solver with $SM \leq 1\text{km/s}$ and $\beta = 4$. Table 6-7 presents the configuration, number of feasible solutions and best solution for the ACO solver with $SM \leq 1\text{km/s}$ and $\beta = 5$, which corresponds to the **best** solution with a Δv of **7.2741 km/s** encountered with the presented approach and by any heuristics used **within CASTPath** in 1h of execution time (best previous solution by Tena of 7.6962 km/s using a hybrid solver of ACO with Tabu Search and GA [33]).

Table 6-6. Parameters and results of ACO solver with $SM \leq 1\text{ km/s}$ and $\beta = 4$

Score Matrix threshold	1 km/s		
Execution time	60 min (best)	30 min	10 min
Number of feasible solutions	40	17	7
ACO parameters	$\alpha = 1, \beta = 4, \nu = 54, \iota = 54$		
Best tour cost Δv	7.6056 km/s		
Best tour sequence (MOID time IDs)	[2 14 17 43 54 87 94 103 118 135 141 152]		
Best tour sequence (Asteroid IDs)	[79285 39775 56848 79766 58905 83559 88535 80332 7758 88041 82192 72195]		

Table 6-7. Parameters and results of ACO solver with $SM \leq 1\text{ km/s}$ and $\beta = 5$. Best solution encountered by heuristics in CASTPath

Parameters	1 km/s		
Execution time	60 min (best)	30 min	10 min
Number of feasible solutions	97	28	12
ACO parameters	$\alpha = 1, \beta = 5, \nu = 54, \iota = 54$		
Best tour cost Δv (best overall)	7.2741 km/s		
Best tour sequence (MOID time IDs)	[2 14 17 33 57 85 102 118 129 135 141 152]		
Best tour sequence (Asteroid IDs)	[79285 39775 56848 5233 17646 26797 46 7758 80319 88041 82192 72195]		

6.4.2 Removing the pheromone contribution, $\alpha = 0$

Given the good tendency in all the cases studied of performing better when increasing the heuristic parameter weight, an extreme test was performed removing the pheromone ($\alpha = 0$) maintaining the same Δ_{ind} cost weight as in the previous test cases ($\iota = 54$). Figure 6-12 shows the boxplot result. The results were not the expected ones, as non-feasible sequences were found by the ants. Moreover, in 10 minutes of execution the solver is not even capable of finding any 12-asteroid tour. In 30 minutes, a total of 2 12-asteroid solutions are encountered in rounds 1 and 10, null in the rest of the rounds. In 60 minutes, a total of 3 are found, 2 for round 4 and 1 for round 9. Thus, the boxplot is presented to have a visual representation of this case, but it is not representative of the solver performance. It can be concluded that **without the pheromone contribution ($\alpha = 0$), the solver is not able to find feasible solutions as per REQ-001 (12-asteroid tour) and thus, there exists trail-dependency** in the problem.

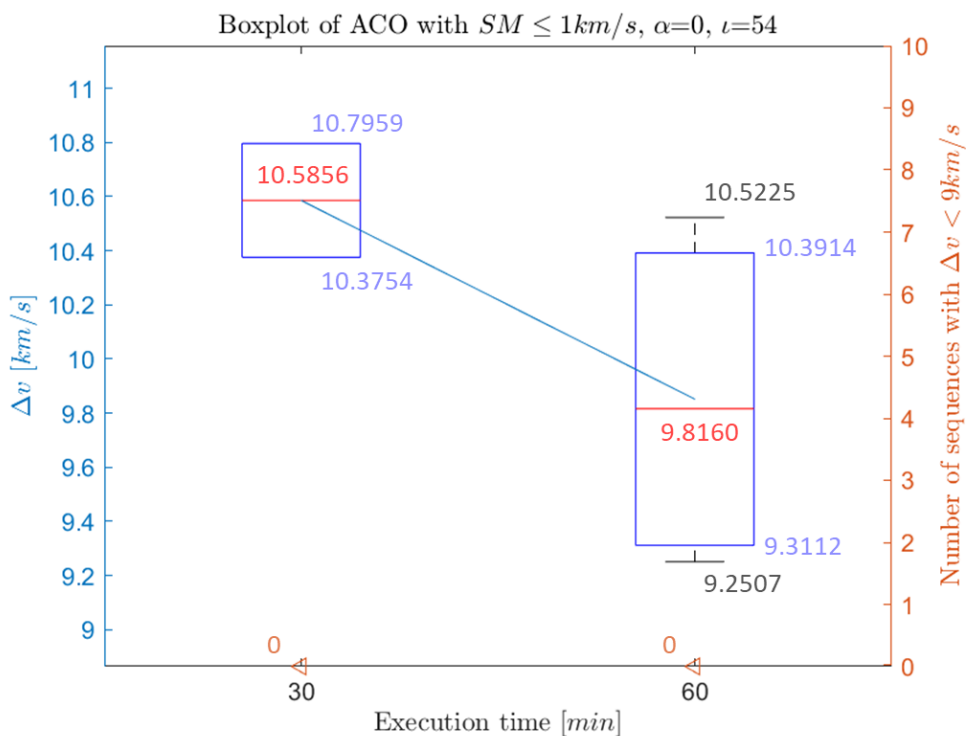


Figure 6-12 Boxplot of results of ACO with $SM \leq 1 km/s$ and $\alpha = 0$. Note that only 2 solutions exist for 30 min and 4 solutions for 60 min

6.4.3 Comparison of $\iota = 54$ and $\iota = 611$

Finally, a test taking the best parameters of each test case was performed: ACO solver with $SM \leq 1\text{km/s}$, $\alpha = 1$, $\beta = 5$, $\nu = 54$ and $\iota = 611$. The boxplot of the results is presented in Figure 6-13. This configuration leads to expect better results than the same configuration with $\iota = 54$ because previous test have performed better with $\iota = 611$. However, the results are quite poor as the solver is not able to find solutions that satisfy REQ-002 ($\Delta v \leq 9\text{km/s}$) and many solutions are close to the theoretical maximum for this SM set-up (18.2889 km/s, explained in section 6.4.1, note that the real maximum is lower). Moreover, on the contrary for the previous test cases, the three series present several higher and lower outliers which implies that the ants seem a bit lost in the Search Space. It can be concluded that, as anticipated in 6.3. Score Matrix filtered at 9 km/s, increasing the Δind importance gives penalty to differentiated indices that can give good results, which can be a bigger penalty in a more reduced Search Space as the one analysed here.

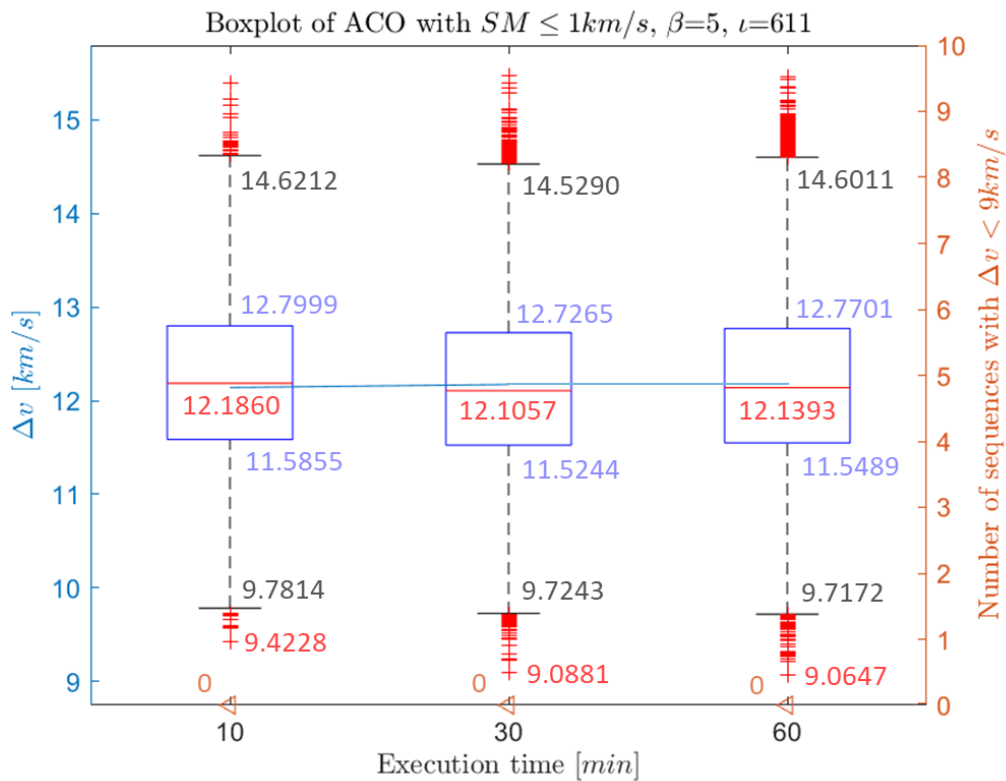


Figure 6-13. Boxplot of results of ACO with $SM \leq 1\text{ km/s}$, $\beta = 5$ and $\iota = 611$

7 DISCUSSION

This section provides a further discussion of the results presented in the previous section 6. Results.

7.1 Comparison of the different set-ups

In this section, a general comparison of the results is made about the index reward, the smoothness of the Search Space and the trail dependency of the nodes.

7.1.1 Index reward

The difficulty of the ants to find feasible solutions as per REQ-001 (obtaining 12-asteroid sequences) has been noticed when dealing with pruned spaces in the tri-structured Search Grid. This has been solved by applying the bi-objective weighted cost approach presented in 4.5.3.1. ACO with index reward. This is due to the “wholes” left in the Search Grid when the pruning of the SM is done. When the SM threshold was lower, and therefore the Search Grid reduced, the ants backtracked more times (for the same setting of parameters). Bear in mind that backtracking was done when the ant detected that there were no possible next nodes to continue with the search, i.e., it had entered or was about to enter a whole in the grid (see Algorithm 0-1 for further details).

However, increasing the index reward weight (ι) does not seem to improve the results scores as much as increasing the heuristic parameter weight (β). In fact, special care has to be taken when tuning these two parameters as it has been seen that a too high index reward weight gives penalty to feasible solutions and there is a point when increasing β in which it is better to not to increase ι .

7.1.2 Smoothness of the Search Space

By smoothness of a search space, the difference in cost in the neighbourhood of the nodes is meant. This is, if an ant is on a node whose leg has a very good cost but when moving not very far away from that node, all the surroundings

have a very different worse cost (and this is repeated in various spots of the grid) it can be said that the search space is very sharp.

The results shown in 6.2. Whole Search, where the solutions were very dependent on the first iterations, lead to think that the space is sharp. Not only this, but the struggle of the ACO solver before implementing the index reward also could be related. It is true though, that ACO algorithms perform better when the Search Space is not very constrained [43].

If this is true, when filtering the Score Matrix even more, the Search Space would be being smoothed as the curve is flattened and thus, this would be the reason why it is easier for the ants to find feasible solutions in the Search Grid $\mathcal{G}_{c,1}$ than in $\mathcal{G}_{c,9}$.

This is a hypothesis that could be tested by creating a plot of the Δv values of a sequence, and then removing just one node, swapping it by another one calculate its Δv and swapping it again by another one, till all the neighbourhood has been swapped. If this plot presents many changes in the surroundings it would mean that the space is not smooth.

7.1.3 Trail dependency of the nodes in the Search Grid

As it has been seen, a higher heuristic weighted ACO (higher β) performs better. This means giving less importance to the pheromone, that is equally laid on a tour instead of node-by-node. This can mean that the Search Grid nodes are not dependent which makes sense if one thinks about the uniqueness of the paths demonstrated when modelling the nodes as pairs of asteroids.

As described in 4.1. Search Space, the tour cost is a tri-dependency of the prior asteroid, the current asteroid and the next one. Thus, it is not necessary to fly by asteroid 10 to reach asteroid triplets 24-132-149, and the cost of the leg 132-149 prior to have visited asteroid 24 will not change because of having visited or not asteroid 10 before. In the same way, it is clear asteroid 24 will not be visited if the leg 10-132 is chosen and therefore, there should exist a small trail dependency.

In this vein, note also that the pheromone update with the novel implementation of the Avoid Tours (see 4.5.2.5. Pheromone update) translates this problem into the Search Graph pheromone deposition. Continuing with the previous example, if the leg x-10-132 is promptly chosen, the ant will probably end up with an Avoid Tour, resulting on a punishment of the leg x-10-132 by evaporating more of its pheromone. Following this reasoning, the bad results when nulling the pheromone ($\alpha = 0$) make sense.

7.2 Comparison with previous work

Previous work on CASTPath has been presented in 2.3. Previous work on CASTPath. This section makes a comparison of the presented approach with the previous results and solvers used in the project.

7.2.1 Use of Score Matrix

The use of a Score Matrix is possible thanks to the novel modelling of the Search Space by triplets of asteroids (pairs of asteroids as nodes) introduced in 4.1. Search Space. In CASTPath previous work the Δv was priorly approximated, P1 was executed and then a optimisation and refinement was done with P2, which implies high computation times and repeated evaluations. This method, while it is not exact and can lead to inaccurate solutions, is a good way of avoiding a high number of evaluations while the solver is running. However, it has the disadvantage of having to run a second analytical solver that computes the real Δv costs of the solution tours.

By modelling the Search Grid in a way that its paths have unique costs, the solver has the real information about the costs of all the asteroid legs and thus, is given a better tool to find better solutions. Moreover, the run of a second analytical solver is not necessary, reducing the total computational time. These both facts have been proven in the results shown in Table 6-1.

It is worth highlighting the importance of balancing between how much the space search is filtered and the amount of feasible solutions that wants to be obtained. The implemented solver finds several solutions when the Search

Space is pruned below a 1 km/s threshold, but this is probably removing some feasible solutions from the Search Space.

7.2.2 ACO Tabu Search

From the different test cases performed by Tena, the ACO Tabu search is comparable due to the approximation of the approach. In fact, the solver proposed in this thesis is inspired by the results and discussion of Tena. Note that this is not the best solution he finds, as already mentioned in section 2.3. Previous work on CASTPath. Figure 7-1 has been constructed from the data provided by Tena [33] on the ACO Tabu search test case with $\alpha = 1$ and $\beta = 4$. It can be seen that the number of feasible sequences found is not very high and the best score is an outlier, meaning that the algorithm is not very robust. This is probably due to the approximation on the cost of the legs as the algorithm basis is very similar. The Search Space was not constructed based on the triple-dependency of the asteroids and thus, the information the ACO solver has in Tena's case is way lower than in the Search Space used in this thesis. This proves that ACO is able to perform much better when it is given more information on the problem.

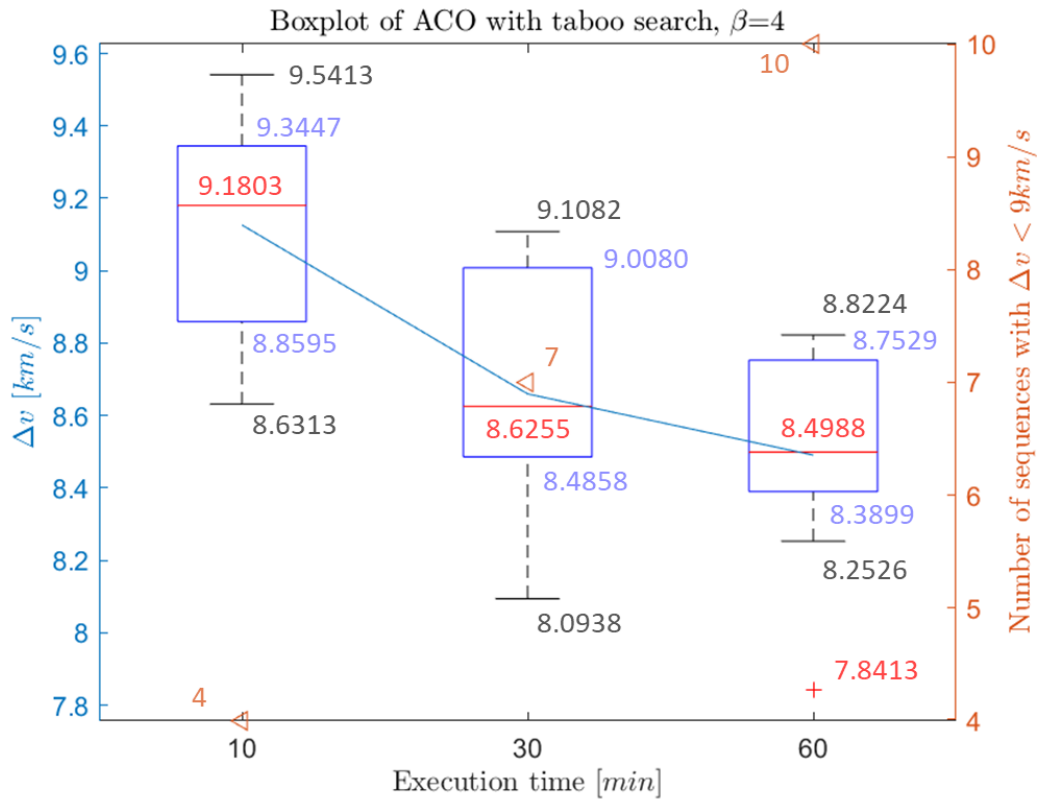


Figure 7-1. ACO Tabu Search results boxplot. Plot made from the data results of Tena [33]

7.2.3 Dynamic Programming

In Bellome et al. [3], a global optimisation solver based on Dynamic Programming was created at the same time as this thesis was developed. Dynamic programming involves the compliance of Bellman's principle of optimality: an optimal policy is independent from the initial decisions (or initial state), i.e., the optimal policy would be the same even if found from intermediate states. Applied to CASTPath, on which it has been demonstrated there exists a trail dependency but only triplet by triplet, this principle holds when working with a Search Space of nodes composed by pair of asteroids. Thus, the trail dependency discussed in 7.1.3. Trail dependency of the nodes in the Search Grid is true for the ACO solver as the ants construct their solutions node by node, ordered in increasing MOID times, but the problem itself is not trail dependant if modelled in pairs, and can be tackled from any node in the Grid.

Bellome et al. presented in [3] the **global optimum** for MOID 0.05 of **6.9772 km/s** with an execution time of 4.7 h. Compared with the best solution encountered by the proposed ACO algorithm in 1 h of **7.2741 km/s**, and taking into account that heuristics do not guarantee global optimality, it can be concluded that the proposed ACO solver is efficient, robust and gives significant good results.

8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

This thesis has proposed a novel metaheuristic solver with ACO for the CASTAway asteroid tour trajectory optimisation problem. A new modelling of the Search Space has been implemented achieving a graph model with unique path costs, due to the triplet dependency and modelling the nodes as pairs of asteroids.

This achievement has made possible the use of a Score Matrix with unique exact information about the leg costs, which is accessible during the solver and, thus, reduces the computational time. Moreover, this achievement deletes the need of dividing the problem into a subproblem P1 with estimated Δv values to find feasible sequences and an analytical subproblem P2 to optimise and give the real Δv values. Instead, a unique problem is solved with exact information about the Δv costs.

Given the importance of finding diverse solutions because of the scientific interest of the CASTAway mission, a similarity measure tool has also been developed. This tool scores the asteroid-by-asteroid and subsequences similarity among the solutions.

The heuristic has been successful on first, fulfilling the two requirements imposed of finding 12-asteroid sequences of a total Δv of 9 km/s as maximum. Second, on tuning the parameters to obtain a total of 97 feasible solutions in 10 independent runs of 1 h with a best score of 7.2741 km/s. And third, on improving the results encountered by a heuristic in CASTPath, so far. It can be concluded that ACO performs better and learns faster when more information is given a priori.

Finally, the finding of the modelling of the Search Space with unique cost paths has enabled the use of Dynamic Programming to find global optima for a given MOID threshold, as the Bellman's principle with this Search Space model is fulfilled. The difference of the best cost by the proposed ACO and the global

optima is just below 0.3 km/s. Thus, it can be concluded that the proposed ACO solver is efficient, robust and gives significant good results.

8.2 Future Work

This work has answered many questions about CASTPath but has opened many others too, enabling many tasks as future work to improve the performance of the solvers or to understand better the nature of the problem (of the Search Space).

As stated in 2.2.2. Metaheuristic algorithms, other heuristics were researched before deciding on focusing the thesis on improving the ACO solver. From the listed ones, the multimodal optimisation looks promising for this project as it can find many local minima and if it finds lots of them, the diversity on the asteroids would be more ensured.

The second interesting heuristic to explore is the multi-objective one. In this work, it can be said (even if it is not strictly the same mathematical approach as the standard one) that a bi-objective approach has been applied to ACO (with the objectives of Δv and Δind , see 4.5.3. Multiobjective formulation of ACO). But apart from this approach, an interesting application of the similarity measure would be to use it while the solver is running to increase the diversity of the solutions. GA tends to give similar results, a bi-objective approach can be used with GA, so using NSGA (normally NSGA-II is used, Non-dominated Sorting Genetic Algorithm II), with the objectives of minimising the Δv and minimising the similarity score.

All the heuristics and deterministic solvers used in CASTPath can be tested using the Score Matrix too.

Finally, the smoothness of the Search Space can be further analysed by creating the plot described in 7.1.2. Smoothness of the Search Space.

REFERENCES

- [1] N. E. Bowles *et al.*, “CASTAway: An asteroid main belt tour and survey,” *Adv. Sp. Res.*, vol. 62, no. 8, pp. 1998–2025, 2018, doi: 10.1016/j.asr.2017.10.021.
- [2] J. P. Sánchez Cuartielles, A. Gibbings, C. Snodgrass, S. Green, and N. Bowles, “Asteroid Belt Multiple Fly-by Options for M-class Missions,” no. September, pp. 26–30, 2016.
- [3] A. Bellome, M. Carrillo, J.-P. Sánchez, J. Del Ser, S. Kemble, and L. Felicetti, “Efficiency of tree-search like heuristics to solve complex mixed-integer programming problems applied to space trajectory design,” 2021.
- [4] Wikipedia contributors, “List of minor planets and comets visited by spacecraft,” *Wikipedia, The Free Encyclopedia*, 2021. .
- [5] ESA, “GTOC Portal,” *ESA*, 2021. https://sophia.estec.esa.int/gtoc_portal/ (accessed Jul. 13, 2021).
- [6] J. T. Olympio, “Optimal control problem for low-thrust multiple asteroid tour missions,” *J. Guid. Control. Dyn.*, vol. 34, no. 6, pp. 1709–1719, 2011, doi: 10.2514/1.53339.
- [7] K. Zhu, F. Jiang, J. Li, and H. Baoyin, “Trajectory optimization of multi-asteroids exploration with low thrust,” *Trans. Jpn. Soc. Aeronaut. Space Sci.*, vol. 52, no. 175, pp. 47–54, 2009, doi: 10.2322/tjsass.52.47.
- [8] A. S. Rivkin *et al.*, “The Main-belt Asteroid and NEO Tour with Imaging and Spectroscopy (MANTIS),” in *2016 IEEE Aerospace Conference*, 2016, pp. 1–14, doi: 10.1109/AERO.2016.7500757.
- [9] M. Di Carlo and M. Vasile, “Low-thrust tour of the main belt asteroids,” *AIAA/AAS Astrodyn. Spec. Conf. 2016*, pp. 1–19, 2016, doi: 10.2514/6.2016-5640.
- [10] MathWorks, “Traveling Salesman Problem: Solver-Based,” *MathWorks*,

2021. <https://uk.mathworks.com/help/optim/ug/travelling-salesman-problem.html> (accessed Jul. 29, 2021).
- [11] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, “Mixed-integer nonlinear optimization,” *Acta Numer.*, vol. 22, no. 2013, pp. 1–131, 2013, doi: 10.1017/S0962492913000032.
- [12] NEOS Guide, “Mixed Integer Nonlinear Programming,” 2020. <https://neos-guide.org/content/mixed-integer-nonlinear-programming> (accessed Aug. 27, 2021).
- [13] M. Schlueter, “Nonlinear mixed integer based Optimization Technique for Space Applications,” no. May, p. 165, 2012, [Online]. Available: <https://core.ac.uk/download/pdf/40015286.pdf>.
- [14] M. Schlueter, S. O. Erb, M. Gerdt, S. Kemble, and J. J. Rückmann, “MIDACO on MINLP space applications,” *Adv. Sp. Res.*, vol. 51, no. 7, pp. 1116–1131, 2013, doi: 10.1016/j.asr.2012.11.006.
- [15] A. Bellome, J.-P. Sánchez, S. Kemble, and L. Felicetti, “Multi-fidelity optimization process complex multiple gravity assist trajectory design,” no. June, pp. 23–25, 2021.
- [16] R. Chai, A. Savvaris, A. Tsourdos, S. Chai, and Y. Xia, “Trajectory Optimization of Space Maneuver Vehicle Using a Hybrid Optimal Control Solver,” *IEEE Trans. Cybern.*, vol. PP, pp. 1–14, 2017, doi: 10.1109/TCYB.2017.2778195.
- [17] N. S. University, “Deterministic vs. stochastic models,” *Department of Statistics*, 2013. <https://www4.stat.ncsu.edu/~gross/BIO560/webpage/slides/Jan102013.pdf> (accessed Jul. 20, 2021).
- [18] NIST/SEMATECH, “Gallery of Distributions,” *e-Handbook of Statistical Methods*, 2013. <https://www.itl.nist.gov/div898/handbook/eda/section3/eda366.htm> (accessed Jul. 20, 2021).

- [19] I. E. Grossmann, "Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques," *Optim. Eng.*, vol. 3, no. 3, pp. 227–252, 2002, doi: 10.1023/A:1021039126272.
- [20] R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," *Math. Program.*, vol. 66, no. 1, pp. 327–349, 1994, doi: 10.1007/BF01581153.
- [21] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Math. Program.*, vol. 36, no. 3, pp. 307–339, 1986, doi: 10.1007/BF02592064.
- [22] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discret. Optim.*, vol. 19, pp. 79–102, 2016, doi: 10.1016/j.disopt.2016.01.005.
- [23] A. H. Land and A. G. Doig, "An Automatic Method for Solving Discrete Programming Problems," in *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, M. Jünger, T. M. Lieblich, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 105–132.
- [24] A. M. Geoffrion, "Generalized Benders decomposition," *J. Optim. Theory Appl.*, vol. 10, no. 4, pp. 237–260, 1972, doi: 10.1007/BF00934810.
- [25] B. K.-S. Cheung, A. Langevin, and H. Delmaire, "Coupling genetic algorithm with a grid search method to solve mixed integer nonlinear programming problems," *Comput. Math. with Appl.*, vol. 34, no. 12, pp. 13–23, 1997, doi: [https://doi.org/10.1016/S0898-1221\(97\)00229-0](https://doi.org/10.1016/S0898-1221(97)00229-0).
- [26] M. Schlüter, J. A. Egea, and J. R. Banga, "Extended ant colony optimization for non-convex mixed integer nonlinear programming," *Comput. Oper. Res.*, vol. 36, no. 7, pp. 2217–2229, 2009, doi: <https://doi.org/10.1016/j.cor.2008.08.015>.

- [27] L. Yiqing, Y. Xigang, and L. Yongjian, "An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints," *Comput. Chem. Eng.*, vol. 31, no. 3, pp. 153–162, 2007, doi: <https://doi.org/10.1016/j.compchemeng.2006.05.016>.
- [28] V. Kenny, M. Nathal, and S. Saldana, "Heuristic algorithms," *Northwestern University Open Text Book on Process Optimization*, 2020. https://optimization.mccormick.northwestern.edu/index.php/Main_Page (accessed Jul. 21, 2021).
- [29] E. Osaba, A. D. Martinez, and J. Del Ser, "Evolutionary Multitask Optimization: a Methodological Overview, Challenges and Future Research Directions," 2021, [Online]. Available: <http://arxiv.org/abs/2102.02558>.
- [30] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 518–538, 2017, doi: 10.1109/TEVC.2016.2638437.
- [31] S. Dasa, S. Maity, B. Y. Qu, and P. N. Suganthan, "Real-parameter evolutionary multimodal optimization-A survey of the state-of-the-art," *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 71–88, 2011, doi: 10.1016/j.swevo.2011.05.005.
- [32] I. Fister *et al.*, "Novelty search for global optimization," *Appl. Math. Comput.*, vol. 347, pp. 865–881, 2019, doi: 10.1016/j.amc.2018.11.052.
- [33] I. Tena Acebo, "Global Optimisation for Large Combinatorial Space Trajectory Design problems: CASTAway case study," Cranfield University, 2020.
- [34] G. Curzi, "Trajectory Design of a Multiple Flyby Mission to Asteroids," *Angew. Chemie Int. Ed.* 6(11), 951–952., no. April, pp. 2018–2019, 2016.
- [35] JPL, "JPL Small-Body Database Search Engine," 2021. .

- [36] B. W. Barbee, "Mission Planning for the Mitigation of Hazardous Near Earth Objects," The University of Texas at Austin, 2005.
- [37] J. J. Lu and M. Zhang, "Heuristic Search," in *Encyclopedia of Systems Biology*, W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, Eds. New York, NY: Springer New York, 2013, pp. 885–886.
- [38] M. Bilal, "A Heuristic Search Algorithm for Asteroid Tour Missions," Cranfield University, 2018.
- [39] M. Dorigo, V. Maniezzo, A. Colorni, and M. Dorigo, "Positive Feedback as a Search Strategy," *Tech. Rep. 91-016*, no. June, pp. 1–20, 1991, [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.6342>.
- [40] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, 1997, doi: 10.1109/4235.585892.
- [41] T. Stützle and H. H. Hoos, "MAX-MIN Ant System," *Futur. Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000, doi: 10.1016/S0167-739X(00)00043-1.
- [42] M. Galarnyk, "Understanding Boxplots," *towards data science*, 2018. <https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51> (accessed Aug. 26, 2021).
- [43] S. Direct, "Ant Colony Optimization," *Science Direct*, 2021. <https://www.sciencedirect.com/topics/engineering/ant-colony-optimization> (accessed Aug. 30, 2021).
- [44] Tutorials point, "Convex Optimization - Polyhedral Set," *Tutorials point*, 2021. https://www.tutorialspoint.com/convex_optimization/convex_optimization_polyhedral_set.htm#:~:text=Advertisements,n%7D (accessed Aug. 27, 2021).

APPENDICES

Appendix A Variables definition

Table A-1. List of variables, values and definition

Variable	Value	Definition
Δv	-	Difference in velocity
t_{MOID}	-	MOID time
A_k^m	-	Asteroid with MOID ID k in position m of the asteroid sequence
$(A_i^{m-2}, A_j^{m-1}, A_k^m)$	-	Asteroid triplet with MOID ordered indices i, j, k
$\Delta v_{j,k}^i$	-	Δv required for the spacecraft to fly by asteroid k from asteroid j flyby, coming from asteroid i
\mathcal{G}	-	Search Grid
\mathcal{G}_c	-	Cleaned Search Grid (input to the solver)
S_i	-	Subspace containing all the feasible pairs of asteroids beginning by S_i
n_{ast}	158	Number of different asteroids in the Search Space (for MOID ≤ 0.05 AU)
SM	-	
N	-	Feasible set of solutions to the problem
L_{tour}^{ast}	12	Length of the tour sequence of asteroids (not considering Earth or Mars)
Δv_{tot}	-	Total Δv of a tour, including Earth - 1 st asteroid leg
Δv_{max}	9 km/s	Maximum Δv imposed for the tour
$n_{pairs,from}$	11,889	Number of pairs of asteroids
Similarity Measure		
Ψ	-	Total similarity score
ψ_{ast}	-	Asteroid-by-asteroid similarity score
w_{ast}	1	Asteroid-by-asteroid similarity score weight

ψ_{sub}	-	Subsequences similarity score
w_{sub}	1	Subsequences similarity score weight
ACO solver		
ξ	-	Candidate solution tour
P	-	Probability of choosing next node
τ	-	Pheromone parameter
η	-	Heuristic parameter
α	1	Exponent weight to pheromone
β	4	Exponent weight to heuristic parameter
ρ	0.5	Pheromone evaporation rate
n_{bt}	-	Backtracking counter
$n_{bt,max}$	50	Maximum number of backtrackings allowed
t_{exec}		Execution time
$f_{(i,j),k}$	-	Relative fitness of triplet (i, j, k)
$F_{(i,j)}$	-	Sum of the relative fitness of all the candidate nodes from node (i, j)
$s_{(i,j),l}$	-	Cumulative sum of relative fitnesses till asteroid l
Δind_{jk}	-	Difference between MOID time IDs $(k - j)$
ν	54	Weight of Δv objective in the cost
ι	611	Weight of Δind objective in the cost

Appendix B Backtrack

Algorithm 0-1. Implemented ACO: Backtrack

