16th International Conference on Space Operations, Cape Town, South Africa – 3 - 5 May 2021.

SpaceOps-2021,1,x1532

Low thrust multi-injection approach for constellation and multi-mission deployment

Vincenzo Maria Salvato[a], Jacopo Prinetto[b], Michèle Lavagna[c]

[a] Department of Aerospace Science and Technology, Politecnico di Milano, via La Masa 34, Milan 20156, Italy, vincenzomaria.salvato@mail.polimi.it
[b] Department of Aerospace Science and Technology, Politecnico di Milano, via La Masa 34, Milan 20156, Italy, jacopo.prinetto@polimi.it
[c] Department of Aerospace Science and Technology, Politecnico di Milano, via La Masa 34, Milan 20156, Italy, michelle.lavagna@polimi.it

Abstract

This work suggests a new approach to fast identify feasible profiles to simultaneously and efficiently inject into orbit multiple space assets by means of a deployer equipped with low thrust control authority. The proposed algorithm, while finding an optimal trajectory to deploy N heterogeneous satellites correctly phased on their operational orbits, aims of being computationally light and fast. The algorithm is proposed to be as flexible as possible in terms of scenarios, being compatible with both single-launch homogeneous space assets constellation and heterogeneous multi-satellites deployment, differing in final orbit insertion and physical properties. A multi objective optimization is preferred, aiming to minimize the fuel consumption and the time to operations. The low-thrust trajectory to follow between two arbitrary orbits in space is found by exploiting a 3-dimensional shape based algorithm, based on non-linear interpolation of consecutive orbits. To perform the most demanding transfers in terms of propellant and time at reasonable costs, i.e. those entailing large plane changes (i.e. Right Ascension of the Ascending Node (RAAN) and inclination), the asymmetrical Earth gravitational field induced secular variation is exploited. The paper will discuss the sensitivity analysis with respect to the variations in the optimization search space with particular attention to the computational load. A critical comparison between the adoption of a pure heuristic optimization algorithm and its hybridisation with a branch and bound approach is also discussed in terms of computation efficiency. Finally, a realistic scenario is presented to highlight the proposed approach performances when stressed in terms of both search space and in presence of constraints. The results show how this approach is suitable to deploy several satellites directly onto their operational orbit with only one launch, even when the orbits deeply differ in their keplerian elements.

Keywords: low-thrust; routing; optimization; constellation; multi-mission; multi-deployment.

Nomenclature

$\mathbf{K}$	Target states matrix	p	Semi-latus rectum
$\mathbf{P}$	Permutations matrix	$R_\oplus$	Earth radius
a	Semi-major axis	$\dot{\Omega}_{sec}$	Secular RAAN shift
e	Eccentricity	t_{wait}	Waiting time
i	Inclination	N_{iter}	Number of iterations
Ω	RAAN	N_{perms}	Number of permutations
ω	Argument of perigee	r	Subsearch dimension
θ	True anomaly	x	Elements still to reach
n	Mean angular motion	h	Elements already reached

i	Initial conditions
f	Final conditions
N_p	Population size
M_{gen}	Maximum number of generations
M_0	Initial wet mass
T	Thrust
I_{sp}	Specific impulse
M_{prop}	Propellant mass consumption
r_{perc}	Partially dominated solutions percentage range
L	Constraint Limit
JD	Julian Date

Acronyms

ADR	Active Debris Removal
GA	Genetic Algorithm
GTOC	Global Trajectories Optimization Competition
KP	Keplerian Parameters
LEO	Low Earth Orbit
MGA	Multiple Gravity Assists
MINLP	Mixed-Integer Nonlinear Programming
MOPSO	Multi-Objective Particle Swarm Optimization
PSO	Particle Swarm Optimization
RAAN	Right Ascension of the Ascending Node
TOF	Time Of Flight
TPA	Two Phase Algorithm
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

1 Introduction

1.1 Objectives

With the advent of CubeSats and SmallSats, which range from 0.01 to 180 kg [1], the need for new techniques to launch in space these classes of satellites is arising. In the incoming years, a substantial increment in the number of small satellites to be launched has been forecasted [2]. Studies suggest that the number of satellites launched in the decade 2019-2028 will have a x4 growth rate compared to the previous decade and that satellites with a launch mass $< 500\ kg$ will account for 87% of such number. In addition to this, it was found that "despite a growing number of operational dedicated launch vehicles, the majority of nano/microsatellites in 2019 chose to leverage rideshare alternatives" [3]. The drawback of piggyback launches is in the fact that usually the small satellites are released on the target orbit of the main payload or in its proximity. Consequently, these satellites would need their propulsive system to be capable to allocate themselves on the correct orbit and with the desired phasing. Also, they would need to wait for a launch whose main payload has a target orbit as similar as possible to their final one. Being such satellites the largest market share, new ways to facilitate their access to space are being investigated. A possible solution to overcome the drawbacks of piggyback payload launches is to develop the last stages of launchers or dedicated vehicles able to carry the small satellites directly on their operational orbit. The objective of the paper is to develop an algorithm that, given a set of N satellites to release in different positions around Earth with a vehicle with a a low-thrust control authority, is capable to define the optimal releasing order and transfer strategy.

1.2 Literature Survey

Finding the optimal hopping path between different orbits means solving a variant of the Travelling Salesman Problem (TSP), which is a particular formulation of the Vehicle Routing Problem (VRP). The latter belongs to the class of the NP-hard problems, meaning that the computational time required to solve them dramatically increases with the size of the problem. Most of the VRP applications to space have been about on-orbis servicing [4], Active Debris Removal (ADR) [5], [6] or Multiple Gravity Assists (MGA) problem [7]. This variant of the TSP can be formulated as a problem that includes two different kinds of variables: continuous-valued variables, such as the ones

describing the state of the spacecraft in time, and binary variables, such as the ones defining the visiting order of the hopping trajectory. In particular, this problem belongs to the Mixed-Integer Nonlinear Programming (MINLP), the area of optimization which deals with non-linear problems with continuous and integer variables. Alternatively, to solve the problem a two-layer optimization scheme can be adopted, as suggested by Conway and Wall [8]. These different options were investigated in the work by Zhang et al. [9], who showed that the two-level optimization presented the worst performance. In light of such results, this work focuses on finding a solution to the MINLP.

Differently from previous works, a new way to approach the VRP is here proposed, that will be specifically applied to multi-deployment mission scenarios. Moreover, the transfers will not be modeled as solutions to the Lambert problem, like in most of the previous solutions of the VRP in space, but they will be continuous-thrust transfers computed through the adoption of a shape-based approach [10].

1.3 Paper outline

In Section 2 the algorithm developed to solve the VRP and the transfer strategy selected are presented, while in Section 3, the optimization approach adopted to solve the problem is reported. The most relevant results are discussed in Section 4. Finally, conclusions are reported in Section 5.

2 Multi-deployment algorithm

First, in Section 2.1, the main structure of the routing solving algorithm is presented. Even though it is here used to find the releasing order of the multi-deployment mission, this routing algorithm is suitable to any routing problem. Afterward, in Section 2.2, the transfer strategy selected for the multi-deployment scenario is described.

2.1 Routing

2.1.1 Routing architecture

The main workflow of the algorithm is shown in Figure 1.

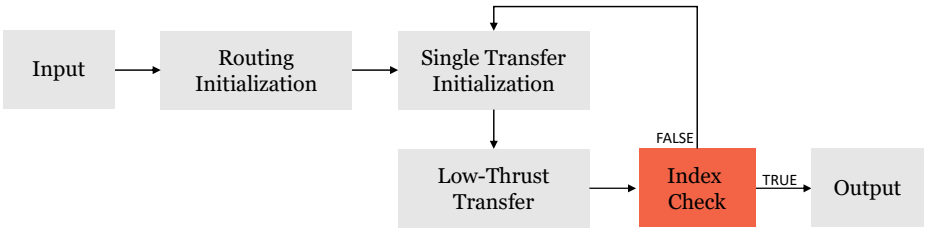

Figure 1: Routing algorithm main workflow.

Input. The main inputs of the algorithm are the initial wet mass of the vehicle M_0, the initial time JD_0, the specific impulse I_{sp}, the initial state KP_0 and the two matrices $\mathbf{K}$ and $\mathbf{P}$. The former is a matrix of dimensions $6 \times N$, where N is the number of orbits onto which to deploy the satellites, or generically the number of destinations of the routing problem. Each column of the matrix contains the Keplerian Parameters (KP) of each orbit. Matrix $\mathbf{P}$ is presented in the next item.

Routing Initialization. In this block, the releasing order is defined. Given N orbits onto which to release the satellites, the optimal path will be one of the possible permutations of the vector $[1, 2, ..., N]$, where each number identifies one of the columns of $\mathbf{K}$. The matrix $\mathbf{P}$ is built outside the algorithm through the operator in Eq. (1), where $x = [1, 2, 3, .., N-1, N]$. The operator builds the matrix $\mathbf{P}$ such to contain all the possible permutations of x and its size will be $N \times N!$.

$$\mathbf{P} = \texttt{perms}(x) \tag{1}$$

Once the matrix **P** is built, only one discrete variable identifying one of the columns of **P** is enough to define the releasing order. The selected column will be referred to as p. Arithmetic overflow might occur when building the matrix **P**. Such issue and more details about the choice and advantages of introducing **P** to solve the VRP are discussed in Section 2.1.2.

Single Transfer Initialization. Once the releasing order is fixed by selecting a column of **P**, the first of the N transfers must be initialized. In this block, the initial mass, current date and target state position are updated. The latter, due to the presence of environmental perturbations, depends on the epoch. In particular, only the J_2 secular contribution of the Earth's gravitational assimetry is taken into account. The target state RAAN is updated according to Eq. (2).

$$\dot{\Omega}_{sec} = -\frac{3nR_{\oplus}^2 J_2}{2p^2} cos(i) \tag{2}$$

Low-Thrust Transfer. In this block, the transfer between the current state and the next one in the releasing order is computed.

Index Check. In this block, a check about the progress of the releasing mission takes place. If the last released satellite does not correspond with the last element of p, the next transfer is initialized. Otherwise, the mission is completed and the final output can be computed.

Output Once the last element of p has been reached, the total propellant consumption and duration of the transfer can be computed. In particular, the total values are the sum of the partial contributions of each transfer.

2.1.2 VRP solution

As aforementioned, the problem belongs to the MINLP family, presenting both discrete and continuous variables. The latter are the ones optimizing the transfers, while the discrete variables are needed to define the route of the releasing vehicle. Since the multi-deployment algorithm must run inside an optimizer in order to find the optimal or near-optimal solutions, it was necessary to find a way to help the optimizer to efficiently evaluate different releasing orders. To speed up the algorithm and guarantee convergence onto a feasible solution, the introduction of the matrix **P** was considered. This solution resulted to be particularly efficient (see Section 2.1.3) since thanks to this choice only one discrete variable identifying the column of **P** is sufficient to define the visiting order of the orbits, allowing to drastically reduce the search space. This variable will be the only discrete variable of the optimization and shall adopt values from 1 to N!, which is the number of columns of matrix **P**.

On the one hand, the introduction of **P** speeds up the convergence of the algorithm with respect to considering N discrete variables to define the visiting order. On the other hand, the building of **P** can require large storage space with increasing values of N. For these reasons the matrix **P** is built only once outside the multi-deployment algorithm and this approach proves ineffective for values of N larger than 11 (e.g. the storage of matrix **P** for $N = 12$ would require 42.8 GB of space). To overcome this limitation, a hybrid optimization approach between a branch and bound method and a heuristic method was implemented, explained in Section 3.

2.1.3 Routing validation

It was interesting to compare the results obtained applying the routing algorithm to a problem whose solution was known in order to validate it and also to assess the algorithm quality and efficiency. The algorithm validation was carried on by solving the problem faced in the 5^{th} Global Trajectories Optimization Competition (GTOC), that was already used for validation in several papers [9] [11]. The scenario of this problem is different from the multi-deployment mission which is the main focus of the algorithm developed in this thesis. However, the two problems share similar features and therefore by applying only really little changes to the algorithm it was possible to apply it to this different problem. The problem deals with three versions of a multiple asteroids rendezvous task, respectively with 4, 8 and 16 targets. The optimization approach explained in Section 3.1 to enlarge the capabilities

of the algorithm only suits multi-objective optimization (due to the branching and bounding criteria chosen). For this reason, the algorithm will only be tested on the 4 and 8 targets cases. The details of the GTOC problem are reported in the work by Zhang et al [9] and are not here reported for the sake of brevity.

The workflow of the algorithm stays the one of the routing algorithm represented in Figure 1, but the transfers will be computed as a single Lambert transfers. Particle Swarm Optimization (PSO) was chosen as computational method for the single-objective optimization to minimize the total Δv of the hopping journey. In particular, the `particleswarm` function of MATLAB Global Optimization Toolbox [12] was used. The default tuning parameters of the algorithm were adopted in this case; a research of the optimal ones might further increase the performances reported in the next paragraphs.

For both the 4 and the 8 targets cases, respectively identified as Case 1 and Case 2, the problem was solved 10 times and the results are shown in Table 2. The results of the routing algorithm (M1) here proposed are compared to the ones found by solving it through two different methods (M2 and M3). The three methods are summarized in Table 1.

Table 1: Methods for routing algorithm validation.

M1	Routing algorithm
M2	Genetic Algorithm (GA) with search enhancement
M3	Two Phase Algorithm (TPA)

The results are compared to the ones found by Zhang et al [9], who tried different computational methods to solve the problem. Only the method which provided the best results is here reported, which are the ones found by performing the minimization through the adoption of a GA with search enhancement (M2). The proposed method will be compared also to the one proposed by Bang and Ahn [11] which consists in a TPA (M3) characterized by a first phase in which some elementary solutions are found which are later used as starting point to solve the TSP.

The results are shown in Table 2, which confirms the validity of the algorithm and also proves its quality in performances. Only the information about the best results found by adopting M3 was available.

Table 2: Results comparison for algorithm validation.

Method	Case 1 [km/s]			Case 2 [km/s]		
	Best	Mean	STD	Best	Mean	STD
M1	6.068	7.558	0.779	17.350	20.989	1.921
M2	6.397	7.307	0.570	19.153	22.978	3.044
M3	6.360	-	-	16.400	-	-

It is interesting to compare not only the results but also the rapidity with which the algorithm finds its final solution. The computation procedure presented in this report ran on a personal computer with an Intel(R) Core(TM) i7-7500 (2.7 GHz) processor and a 16 GB RAM. However, the rapidity of the codes is measured through the evaluation of how many times the Lambert functions are called to find the final solution and therefore independently on the computing machine. For the proposed method M1 the averages of the 10 runs are reported in Table 3 and compared to the number of Lambert calls of the other methods [11]. The results indicate that the computational resource spent by the proposed method is about one order of magnitude smaller than adopting M2 and two than the M3, even though the latter was capable to find a better solution for Case 2.

All things considered, it is possible to deduce that the algorithm works and finds reliable solutions to the problem. Also, it is competitive with respect to similar algorithms in terms of solutions found and especially of computational cost.

Table 3: Number of Lambert routine calls comparison.

Method	Number of Lambert routine calls	
	Case 1	Case 2
M1	88,000	348,320
M2	960,000	7,680,000
M3	3,526,933	25,392,677

2.2 Low-thrust transfer

2.2.1 Single transfer

A 3-dimensional shape-based algorithm [10] was selected for the single transfers since particularly suited to planeto-centric mission scenarios. The shape of the transfer is proposed a priori as a non-linear interpolation of similar and consecutive orbits. A completely analytical shape based approach was necessary due to its really low computational complexity, which makes it possible to evaluate several different trajectories in few seconds. This is of paramount importance for the scope of the multi-deployment algorithm since a really large number of transfers have to be computed as fast as possible.

The shape-based algorithm selected only deals with keplerian motion. At the current state of the art, the environmental perturbations could be accounted a posteriori only in the case in which the perturbing acceleration was at least about one order of magnitude less than the acceleration provided by the thrusters. When dealing with orbits in the Low Earth Orbit (LEO) region, environmental perturbations have an important role, especially in low-thrust since the order of magnitude of the perturbing accelerations are sometimes the same as (if not higher than) the thrust acceleration [13]. For this reason, the impact of the perturbations on the spacecraft trajectories, especially the J_2 effect which is the main contribution, must be taken into account when planning the transfers, since it cannot be simply counteracted and canceled by the spacecraft. The shape-based algorithm is then modified to take into account the J_2 perturbation disturbances. The most accurate way to consider this perturbation would be considering its punctual effect on each revolution around the main attractor. However, this approach would need the integration of the equations of motion which would dramatically increase the computational load. For this reason, only the integral of the effects of the J_2 perturbation are considered in the trajectory computation. After each revolution around the Earth the current state is corrected with the secular effect of the perturbation over that revolution. While correcting the current state with the secular effect of the perturbation introduces some discontinuities in the trajectory and control low, it provides a good estimation of the propellant consumption and transfer duration with a really low computational load, which is the main driver of the algorithm development.

2.2.2 Transfer strategy

To perform the transfers in the most efficient way, a transfer strategy exploiting the secular effect of the J2 perturbation was planned and is here explained. The main idea behind this approach is to change the RAAN of the spacecraft not by thrusting the spacecraft but only exploiting the gravitational perturbation due to the not spherically symmetric mass distribution of the central attractor [14]. The RAAN of a spacecraft on a given orbit can be changed for free by waiting the necessary amount of time without counteracting the J2 perturbation. By doing so, the rate of change of the RAAN would be the one in Eq. (2). Alternatively, the spacecraft can also move to another orbit in order to make the desired change of RAAN happen faster at a cost of a little propellant consumption. This two-legs transfer option turns the transfer problem into an optimization problem whose variables are the KP of the intermediate orbit onto which to stationary for the change of RAAN. The workflow of the transfer strategy is reported in Figure 2 and structured in the blocks described below:

Inputs. The starting orbit KP_1 and the target orbit KP_2 are the two main inputs to the function.

Drifting Orbit Definition. The drifting orbit is defined by the six KP reported in Table 4. As it can be seen

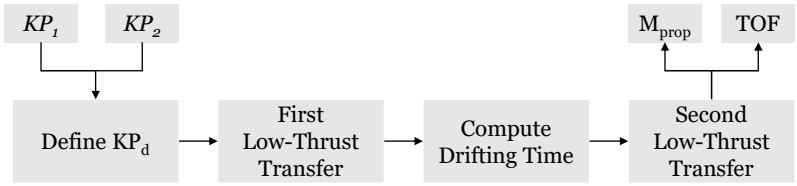

Figure 2: Single low-thrust J2-exploiting transfer scheme.

Table 4: KP of the drifting orbit for low-thrust J2-exploiting transfer strategy.

a_d	e_d	i_d	Ω_d	ω_d	θ_d
var	0	*var*	*free*	$\frac{1}{2}(\omega_1+\omega_2)$	*free*

from Eq. (2), the three parameters which affect the RAAN variation are the semi-major axis a, the eccentricity e and the inclination i. For this reason, not all the KP are considered as variables in order to reduce the size of the optimization problem:

- The semi-major axis is the one which has the greatest impact on the RAAN variation and it is also an element whose value is relatively cheap to control and change, making it the main variable of the optimization.
- Eccentricity affects the RAAN change since a high value means an orbit perigee at a lower altitude and therefore more subject to the gravitational asymmetries of the central planet. However, it is preferred to have a uniform influence of such asymmetry on the spacecraft over the revolutions and therefore this value is chosen to be 0.
- The inclination is a parameter that also greatly affects the RAAN variation but also extremely expensive to change. Therefore, it is considered as variable but only when a change of inclination is necessary from original to target orbit. In that case, the value of the inclination of the drifting orbit is selected between i_1 and i_2. In the case in which i_1 and i_2 have the same value, the inclination is kept the same as the ones of the two orbits unless polar orbits are dealt with. Polar orbits do not experience the RAAN secular variation and, for this reason, it is necessary to depart from 90° inclination to shift the orbital plane.
- Saying that Ω_d is left *free* means that no control is considered on the final value of the RAAN when moving to the drifting orbit. The RAAN of the spacecraft is left to vary according to the J_2 effect. There would be no point or convenience in controlling the RAAN since the goal itself of reaching the drifting orbit is to change such parameter by exploiting the natural perturbations instead of the propellant.
- The argument of perigee ω of an orbit does not affect the secular variation of the RAAN. For this reason, ω_d was not taken as variable of the problem and was arbitrarily set to have a halfway value between ω_1 and ω_2.
- θ_d is *left* free since it affects neither the secular RAAN variation nor the cost or duration of the multi-revolution low-thrust transfer.

First Transfer. Once the drifting orbit is defined, the cost and duration of the first low-thrust transfer are computed.

Drifting Time Computation. Once the target orbit has been reached, it is necessary to compute the amount of time necessary to close the RAAN gap. Generally speaking, knowing the elements of the drifting orbit KP_d and the ones of the target orbit KP_2 it is possible to compute the RAAN gap (Eq. (3a)) at the moment of the arrival on the drifting orbit and the relative drift rate (Eq. (3b)). Once these two quantities are known, it is possible to compute the waiting time (Eq. (3c)) necessary to have $\Omega_2 = \Omega_d$, at a value different from the two original ones.

$$\Delta\Omega = \Omega_2 - \Omega_d \tag{3a}$$

$$\Delta\dot{\Omega} = \dot{\Omega}_d - \dot{\Omega}_2 \tag{3b}$$

$$t_{wait} = \frac{\Delta\Omega}{\Delta\dot{\Omega}} \tag{3c}$$

While this is true for impulsive maneuvers, when dealing with low-thrust transfers the computation of the exact Ω_d at which to depart from the drifting orbit is slightly more complicated. Since low-thrust transfers between the orbits have large times of flight, an amount of relative RAAN shift happens also during the transfer from the drifting to the target orbit. For this reason, departing when the two values of the RAAN are already equal would require some degree of control on such parameter during the transfer to the target orbit to keep such values equal. The most efficient and less expensive way to perform the transfer would be to estimate the relative drift which takes place during the transfer from the drifting to the target orbit and to take it into account when computing the drifting time. To adopt this approach the following steps must be taken:

- A fictitious target orbit is defined with the same parameters as the true target orbit KP_2 apart from the RAAN which is left free, to estimate the amount of shift that would take place during the transfer. Such shift is referred to as $\Delta\Omega_{t,1}$.
- Once computed the transfer to the fictitious target orbit, knowing the Time Of Flight (TOF) of the transfer it is possible to estimate also how much the target orbit shifts during the duration of the transfer. This amount of RAAN shift is defined as $\Delta\Omega_{t,2}$.
- From these two quantities it is possible to compute the amount of relative shift $\Delta\Omega_t$ (Eq. (4a)) and use it to correct the RAAN gap to close (Eq. (4b)). The time necessary is then computed according to Eq. (3c) and with Eq. (2) the new values of the RAAN at the end of the waiting time are updated.

$$\Delta\Omega_t = \Delta\Omega_{t,1} - \Delta\Omega_{t,2} \tag{4a}$$

$$\Delta\Omega = \Omega_2 - \Omega_d + \Delta\Omega_t \tag{4b}$$

Shortly, this correction on $\Delta\Omega$ allows finding the correct Ω_d at which to depart from the drifting orbit, which does not coincide with Ω_2 unless the two orbits are so close or so high in altitude that the J2 effect during the transfer can be considered negligible. Thanks to this estimation, no propellant has to be consumed to control the RAAN because the transfer is planned in such a way to meet the target orbit at the correct value of this parameter.

Second Transfer. Once the RAAN of current and target orbits have been updated after the stationing onto the drifting orbit, the second low-thrust transfer to the target orbit can effectively be computed.

Outputs. At the end, the whole cost of the two-legs transfer is computed. In terms of propellant, the total amount needed to reach the target orbit is the sum of the propellant consumed in the two legs of the transfer. In terms of time, the total duration of the transfer is the sum of the TOFs of the two legs and also the waiting time onto the drifting orbit.

3 Optimization

First, the optimization approach adopted is presented in Section 3.1. Later, more details about how some solutions are discarded and about the introduction of constraints are discussed in Section 3.2 and Section 3.3.

3.1 Optimization approach

A multi-objective optimizations was preferred to minimize the propellant consumption and time duration of the mission. A heuristic optimization method was necessary due to the large search space of the problem. Heuristic methods [15] do not grant to find the optimal solutions to the problem but the low computational effort required to achieve this near-optimal solution makes them very valuable. In particular, population-based methods are more suitable to find promising areas in a large search space [16]. For these reasons, the Multi-Objective Particle Swarm Optimization (MOPSO) [17] method, specially modified to work with discrete variables, will be used to perform all the multi-objective optimizations. The two main tuning parameters of the method are N_p and M_{gen}.

However, while the optimization method selected was the MOPSO, a particular approach to deal with the optimization of the problem was developed and is presented in this section. This approach was at first developed to deal with problem with N> 11, which was a problem after the introduction of **P**. However, it proved to provide better results even when applied to cases of smaller dimension (see Section 4.1).

Instead of dealing with the whole hopping trajectory at once, the problem is broken up into smaller and consequential subproblems. A parameter r which defines the dimension of the subproblem is chosen, defining also the number of iterations necessary to solve the problem of reaching the N orbits. The number of iterations N_{iter} needed to solve the problem can be computed by Eq. (5), where the operator rounds the result of the fraction to the nearest integer greater than or equal to it. In the case in which the remainder after the division of N by r is different from zero, the last iteration is performed with a subset of size r equal to the remainder.

$$N_{iter} = \texttt{ceil}\left(\frac{N}{r}\right) \tag{5}$$

When adopting this hybrid approach, the matrix **P** to give as input to the routing algorithm is built through the use of a different operator than Eq. (1). The new operator is shown in Eq. (6) and builds a matrix which, given a set of N orbits to reach, only contains the permutations of a subset of r elements of the vector x.

$$\mathbf{P} = \texttt{subperms}(x, r) \tag{6}$$

The number of possible permutations N_{perms}, in this case, is not anymore N! but can be found through Eq. (7). The size of the matrix **P** will be $r \times N_{perms}$.

$$N_{perms} = \frac{N!}{(N-r)!} \tag{7}$$

The pseudo code of the optimization approach is presented in Algorithm 1, whose steps are described below, and is represented by the scheme in Figure 3. In the latter, each circle represents a single solution. The grey solutions actually represent several solutions, whose number is unknown a priori due to the heuristic nature of the optimization.

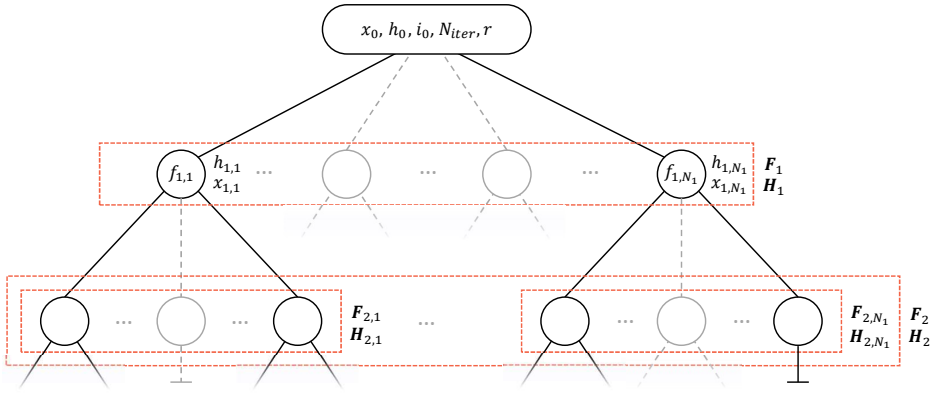

Figure 3: Branch and bound based heuristic approach scheme.

Algorithm 1: Branch and bound based heuristic approach

Data: x_0, i_0, h_0, N_{iter}, r
Result: $\mathbf{F}_{N_{iter}}$, $\mathbf{H}_{N_{iter}}$

```
begin
    [F_1, H_1] = branching(x_0, i_0, h_0, r)
    for j = 2 to N_iter do
        N_{j-1} = length(H_{j-1}(:, 1))
        for k = 1 to N_{j-1} do
            h_{j-1,k} = H_{j-1}(k, :))
            x_{j-1,k} = exclude(x_0, h_{j-1,k})
            i_{j-1,k} = F_{j-1}(k, :))
            [F_{j,k}, H_{j,k}] = branching(x_{j-1,k}, i_{j-1,k}, h_{j-1,k}, r)
        end
        H_j = [H_{j,1}; H_{j,2}; ... ; H_{j,N_{j-1}}]
        F_j = [F_{j,1}; F_{j,2}; ... ; F_{j,N_{j-1}}]
        [F_j, H_j] = bounding(F_j, H_j)
    end
    [F_j, H_j] = boundingpareto(F_j, H_j)
end
```

- The inputs to the algorithm are the vector of all the elements still to be reached x, the vector of initial conditions i, the vector containing the indices of the destinations already reached h. At the first iteration $x_0 = [1, 2, .., N]$ and h_0 is initialized as an empty vector. In addition to these, the number of iterations N_{iter} computed in Eq. (5) and the sub-search dimension r are also given as input to the algorithm.
- The first iteration is performed outside the **for** loop. The operator `branching`, starting from one initial condition, runs the heuristic optimizer and finds a set of partial solutions which make the first branches of the algorithm. The matrix $\mathbf{P}$ at this iteration is computed through Eq. (6), with x_0 as input. The heuristic algorithm only gives as output the solution which respect the bounding criteria, explained in Section 3.2. Each solution is characterised by $f_{j,1}$, $h_{j,1}$ and $x_{j,1}$ vectors. The survived solutions are stored in two matrices:
  - $\mathbf{F}_j = [f_{j,1};\ f_{j,2};\ ...\ ;\ f_{j,N_j}]$ is the matrix containing the final conditions of each solution, including the values of the objectives.
  - Each row of $\mathbf{H}_j = [h_{j,1};\ h_{j,2};\ ...\ ;\ h_{j,N_j}]$ contains the indices of the elements reached by the respective partial solution.
- Now the first **for** loop begins. At each iteration, N_{j-1} is computed, which is the number of the branches coming from the previous iteration. The operator `length` computes the number of elements of the vector of input. Each of the branches represents a solution that must be expanded in the following **for** loop.
- One branch at a time, the iteration is initialized defining $h_{j-1,k}$, $i_{j-1,k}$ and $x_{j-1,k}$. The latter in particular is defined by the operator `exclude`. This operator cancels from x_0 all the elements already reached by that partial solution, identified by $h_{j-1,k}$. The new vector $x_{j-1,k}$ will have dimension $N - r \cdot j$.
- From each branch, a new set of branches is found again through the operator `branching`. At each iteration a different $\mathbf{P}$ is given as input to the routing algorithm, again computed through Eq. (6), each time with $x_{j-1,k}$ as input. The branches are stored in $\mathbf{F}_{j,k}$ and $\mathbf{H}_{j,k}$.
- Once all the branches have been expanded, the solutions are stored in the matrices $\mathbf{F}_j$ and $\mathbf{H}_j$, which include all the branches born from the current iteration.

- Before starting the next iteration, the bounding criteria must be applied to $\mathbf{F}_j$ and $\mathbf{H}_j$. While it is true that each $\mathbf{F}_{j,k}$ is composed by solutions which survived the bounding inside the single optimization, now they must be compared to the all the other set of solutions of the iteration. The bounding criteria are applied by the operator `bounding` and the new $\mathbf{F}_j$ and $\mathbf{H}_j$, containing only the survived solutions, are given as output. In Figure 3 the solutions which do not survive the bounding criteria are indicated with the symbol $\perp$.
- The solutions of the iteration $j-1$ which survived the `bounding` operator, are now expanded themselves.
- After the last iteration, if necessary, a more stringent bounding criterion is applied to the final set of solutions. In particular, through the use of the operator `boundingpareto` only the non-dominated solutions are kept.

3.2 Bounding criteria

The operator `bounding`, given a set of solutions and some bounding criteria, only keeps the solutions that respect the latter while discarding all the others. Choosing to discard all the solutions not belonging to the Pareto front of the sub-problem would lead to the risk of discarding some partial solutions really close to the Pareto front, which might eventually become optimal solutions when extending them. Therefore, a region of solutions must be selected: all the solutions belonging to the Pareto front or within a certain percentage range from one of the solutions in the front can be kept and then extended. These solutions outside the Pareto front but which survive the bounding criterion are referred to as "partially dominated solutions", even though they are actually fully dominated according to the definition of dominated solutions. The value r_{perc}, chosen between 0 and 1, sets the percentage range within which a dominated solution is considered to be only partially dominated. A schematic representation of such solutions is reported in Figure 4. Indicating with $f_{1,D}$ and $f_{2,D}$ the objectives of a dominated solutions and with $f_{1,P}$ and

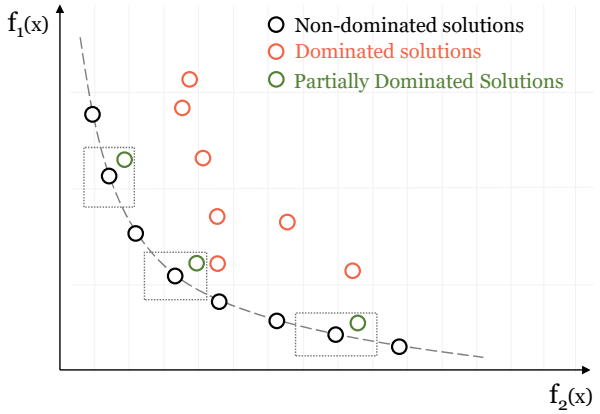

Figure 4: Pareto front and partially dominated solutions.

$f_{2,P}$ the ones of the closest non-dominated solution, a dominated solutions is considered to be partially dominated if both the criteria in Eq. (8) are met.

$$\frac{f_{1,D} - f_{1,P}}{f_{1,P}} < r_{perc} \tag{8a}$$

$$\frac{f_{2,D} - f_{2,P}}{f_{2,P}} < r_{perc} \tag{8b}$$

Graphically, each Pareto solution has a rectangle with sides of length $r_{perc} \cdot f_{1,P}$ and $r_{perc} \cdot f_{2,P}$ which defines the range into which a dominated solution is considered to be partially dominated.

3.3 Constraints

When dealing with engineering problems, it is of paramount importance that an algorithm allows the introduction of constraints to some of the variables. In the routing algorithm, the constraints were introduced by the use of penalty functions.

Penalty functions allow treating of constrained problems as unconstrained problems, introducing an artificial penalty when the constraint is violated, but may create severe slope changes or discontinuities in the solution space, which could interfere with a heuristic optimization algorithm. To help the latter converge onto feasible solutions, it was chosen to introduce a penalty function whose penalty is proportional to the amount of violation of the constraint. For instance, considering a generic objective f_1 and an upper limit L_1, the objective function is modified adding the quadratic loss function $\lambda(f_1, L_1)$ in Eq. (9).

$$\lambda(f_1, L_1) = \max(0, f_1 - L_1)^2 \tag{9}$$

The new formulation of the objective is presented in Eq. (10),

$$f_1 = f_1 + F \cdot \lambda(f_1, L_1) \tag{10}$$

where F is the penalty factor, a scalar greater than 0 and arbitrarily chosen depending on the order of magnitude of f_1.

4 Results and Discussion

In Section 4.1 an example of multi-deployment is reported, solved with both a traditional pure heuristic approach and the branch and bound based heuristic one. Afterward, in Section 4.2, a case of deployment of an existing constellation is dealt with. Due to the relatively high mass of the satellites belonging to existing constellations around Earth, the deployment of only one portion of the constellation is considered.

4.1 Heuristic-hybrid approaches comparison

A generic set of 6 satellites to deploy in LEO was considered. The KP of the orbits into which to insert each of them are reported in Table 5, together with the starting orbit, indicated by the index 0, from which the vehicle

Table 5: Initial states for multi-deployment mission example.

ID	a [DU]	e [-]	i [deg]	Ω [deg]	ω [deg]
0	1.05	0.02	66	0	0
1	1.06	0.01	67	10	5
2	1.07	0.02	66	8	0
3	1.08	0.01	67	328	3
4	1.09	0.03	66	161	0
5	1.10	0	68	22	0
6	1.11	0.05	66	159	20

starts the journey. DU is equal to the radius of the Earth, which was used to non-dimensionalize the semi-major axes. All the KP reported are the ones at the time of the departure from orbit KP_0. The lower and upper bounds of the optimization for this example are the ones in Table 6. The inclinations of the drifting orbits are allowed to vary between the minimum and maximum value of the inclinations of the sets of orbits into which to release the satellites.

The characteristics of the deploying vehicle chosen for this example are presented in Table 7. All the 6 satellites to be released were considered to be nano-satellites of 5 kg mass each.

First, the problem was solved with a pure heurstic approach addressing the $N = 6$ problem directly. $N_p = 20$ and $M_{gen} = 20$ were chosen for the only iteration of the optimization. Afterward, the hybrid approach was carried on with $r = 3$, meaning that two iterations were needed to find the final solutions. Two different values of N_p and

Table 6: Optimization tuning parameters for multi-deployment mission example.

Parameter	lb	ub
a_d [DU]	1.0314	1.300
i_d [deg]	66	68

Table 7: Spacecraft characteristics for multi-deployment mission example.

M_0 [kg]	T [N]	I_{sp} [s]
100	0.5	3500

M_{gen} were chosen for the first and second iteration, reported in Table 8. This was done since the second iteration has a solutions space smaller than the first iteration. While the first iteration has 120 possible permutations according to Eq. (7), the second iteration only has 6, since the matrix **P** in this case only has N! possible columns (Eq. (1)), with this time $N = r = 3$.

Since the final result of a heuristic optimization method strongly depends on the initial solution, comparing only one optimization run to compare the results would not be really significant. For this reason, in order to truly assess whether or not one approach is better than the other, more than one run per each must be considered. For both of them, 10 runs were performed and the 10 Pareto fronts have been merged into one, where only the non-dominated solutions survived. The two Pareto fronts are shown in Figure 5, while the transparent markers represent the clouds

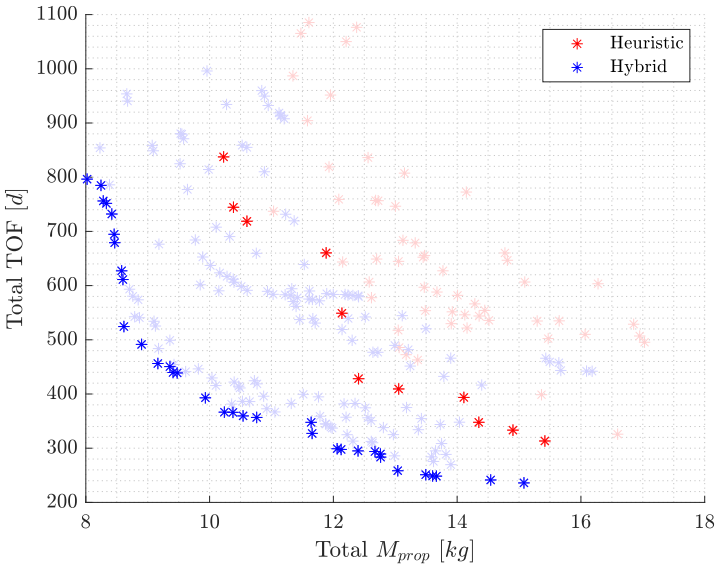

Figure 5: Comparison of optimization approaches results ($M_0 = 100\ kg$; $I_{sp} = 3500\ s$).

of solutions found by the 10 runs of the optimization for both the approaches. It is clear how the Pareto front found from the hybrid optimization approach is composed by far better solutions with respect to the pure heuristic one. Indeed, the blue Pareto front dominates the red one in all its solutions found.

It is important when comparing the two optimization approaches to compare not only the final results but also the computational effort necessary to achieve them. The tuning parameters N_p and M_{gen} were chosen to have a similar duration of the runs between the two approaches. The time is indeed proportional to the function

Table 8: Optimization tuning parameters of hybrid approach for optimization approaches comparison.

	Iter 1	Iter 2
N_p	10	5
M_{gen}	20	10

evaluations necessary to find the solution. The average duration of the 10 runs of the hybrid approach was 225.5 s, while the heuristic one needed an average of 282.3 s per run. It is possible to conclude that the hybrid approach outperforms the pure heuristic one both in terms of results and computing effort. In light of such results, the hybrid optimization approach will be chosen as the standard to perform the minimization of the objectives, also when dealing with problems characterized by $N < 11$.

4.2 Constellation insertion

Here, a possible real application is dealt with: the deployment of a portion of the satellites belonging to the Starlink [18] constellation was considered. The satellites of this constellation have a mass of 260 kg each. The Starlink spacecraft constellation will be spread into 24 orbital planes with an inclination of 53°, on circular orbits with an altitude of 550 km. A possible case may be the replacement of 6 satellites of the constellation, each on a different orbital plane. Supposing that the multi-deployment vehicle is already released on one of the orbital planes of the constellation (the one referred to with ID 0), the planes onto which to release the satellites are reported in Table 9.

Table 9: Initial states for Starlink replacement mission.

ID	a [DU]	e [-]	i [deg]	Ω [deg]
0	1.0862	0	53	0
1	1.0862	0	53	15
2	1.0862	0	53	30
3	1.0862	0	53	45
4	1.0862	0	53	60
5	1.0862	0	53	90
6	1.0862	0	53	105

The initial wet mass M_0 of the releasing vehicle was considered to be 2000 kg, according to mass estimating statistical relationships [19]. The vehicle is considered to be provided with ten RIT 2X Series [20] thrusters, each with a nominal thrust of 171 mN and constant specific impulse of 3500 s.

The optimization of the multi-deployment mission was carried on through the use of the branch and bound based heuristic approach. With the size of the problem $N = 6$ and the size of the subsearch $r = 3$, two iterations were needed to find the final solutions. The tuning parameters of the optimization are reported in Table 10, while the lower and upper bounds of the research are shown in Table 11.

Table 10: Optimization tuning parameters for Starlink replacement mission.

	Iter 1	Iter 2
N_p	15	10
M_{gen}	30	20

The results of one run of the algorithm are presented in the top Pareto front in Figure 6. A constraint on a maximum mission time of 2 years was imposed, reason why all the results do not exceed 700 d of duration. The

Table 11: Lower and upper bounds for Starlink replacement mission.

Parameter	lb	ub
a_d [DU]	1.0314	1.250
i_d [deg]	53	53

bottom subfigure in Figure 6 represents a focus on the fastest solutions of the complete Pareto front of the top one.

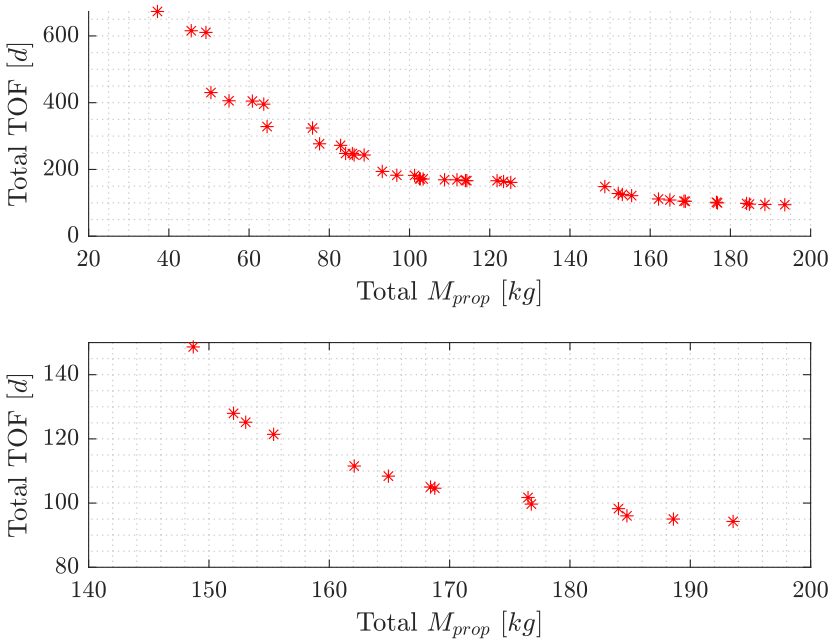

Figure 6: Top: Pareto front for Starlink replacement mission. **Bottom**: focus on fastest solutions ($M_0 = 2000$ kg; $I_{sp} = 3500$ s).

It can be interesting to focus on one solutions to go deeper in detail on the releasing strategy. For instance, the fastest solution is considered. As it could be expected from Table 9, the fastest solutions are characterized by the releasing order $p = [1, 2, 3, 4, 5, 6]$. The releasing details of this solution are reported in Table 12.

Some considerations arise by analyzing the results:

- In four out of the six transfers, the value of a_d was set equal to the upper bound of the solutions space. This suggests that faster solutions, if desired, can be achieved by setting a higher upper bound.
- It can be noticed how the propellant mass required for the last transfers is smaller in magnitude than the one of the initial transfers. This is due to the lightening of the vehicle, whose main contribution is given by the release of the satellites rather than by the propellant consumption. Together with the propellant mass, also the times of flight of the transfers decrease due to the higher acceleration peaks that can be achieved, being the maximum available thrust constant.
- Finally, it is possible to notice how the waiting times t_{wait} on the drifting orbits grow bigger with the going on of the mission. Due to the smaller times of flight, a smaller portion of the RAAN gap is covered during the two legs of the transfers. For this reason, more time must be spent on the drifting orbit to obtain the desired RAAN shift.

Table 12: Release details of the fastest solution for Starlink replacement mission.

	R1	R2	R3	R4	R5	R6
a_d [DU]	1.2136	1.2500	1.1898	1.2500	1.2500	1.2500
i_d $[deg]$	53.00	53.00	53.00	53.00	53.00	53.00
$M_{prop,1}$ $[kg]$	23.44	24.91	13.40	16.14	11.86	7.68
TOF_1 $[d]$	7.1962	7.7478	4.2896	5.0999	3.7778	2.5871
t_{wait} $[d]$	2.8250	0.3743	7.7710	3.1846	13.1871	5.8489
$M_{prop,2}$ $[kg]$	23.16	24.54	13.27	15.92	11.67	7.57
TOF_2 $[d]$	7.1659	7.6668	4.2363	5.0745	3.7877	2.4532

Due to the really high initial mass of the vehicle, an extreme case with 10 thrusters simultaneously firing was considered to get the previous results. It is interesting to study the behaviour of the solutions when changing the number of thrusters the vehicle is provided with. A run of the optimization with the same tuning parameters as before has been performed with 7, 5 and 2 thrusters. The resulting Pareto fronts are plotted in the same figure in Figure 7, where the numbers in the legend represent the number of thrusters that can be fired simultaneously.

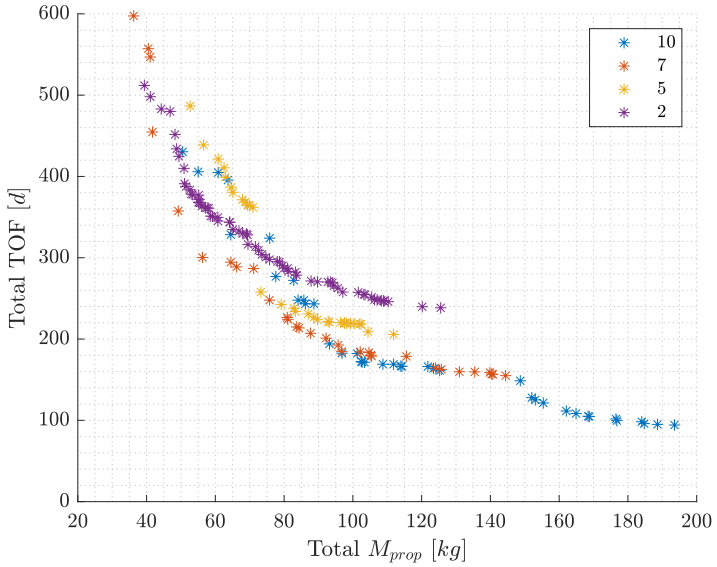

Figure 7: Pareto fronts with different number of thrusters for Starlink replacement mission ($M_0 = 2000\ kg$; $I_{sp} = 3500\ s$).

Respectively, the three configurations lead to a maximum thrust available of 1.1970 N, 0.8550 N and 0.3420 N. As expected, there is no difference in the slow solutions, even due to the maximum mission time duration set to 2 years. The solutions with low propellant consumption and high mission duration are characterized by the choice of a a_d really similar to the semi-major axis of the orbits of the constellation, solutions that can be carried out with each propulsive configuration. The real difference is in the fastest solutions, since the higher the maximum thrust available the faster an orbit with different a_d can be reached. Ten thrusters allow the deployment of the whole set of satellites in about 100 days, while two thrusters configuration requires at least about 250 days to release the six of them.

This comparison is important to understand how the thrust authority does not deeply impact the time duration of the missions, since the greatest contribution to it is given by the amount of time to wait on the drifting orbits to change the RAAN.

5 Conclusions

This paper proposes an algorithm to fast plan and optimize a multi-deployment mission with a low-thrust releasing vehicle. The routing algorithm, thanks to the introduction of the matrix containing the possible permutations as input to the problem, proved extremely effective and fast. Moreover, the branch and bound based heuristic approach increases the performances of the heuristic algorithm by allowing to focus only on the most promising releasing orders of the large search space. Further studies may focus on introducing the J_2 perturbation punctually on the trajectories through an analytical approach.

The results of the simulations present mission durations whose order of magnitude may sound unattractive. However, these values are justified by the choice of the release states of the satellites, characterized by different parameters, and the transfer strategy, which exploiting the J_2 perturbation trades mission duration for low propellant consumption. The only alternatives to release these clusters of satellites are launching them in separate launches or not exploiting the J_2 perturbation to perform the RAAN changes. The former solutions would drastically reduce time to operations if it is possible to arrange several launches to deploy the different satellites. The latter solution would instead dramatically increase the overall propellant consumption of the mission. In light of such considerations, this multi-deployment strategy sounds suitable to the deployment of several satellites whose needed time to operations is not time critical.

References

[1] NASA. *What are SmallSats and CubeSats?* URL: https://www.nasa.gov/content/what-are-smallsats-and-cubesats. //accessed: 30.03.2021 (cit. on p. 2).

[2] Euroconsult. *Satellites To Be Built and Launched by 2028. A complete analysis & forecast of satellite manufacturing & launch services*. 2019, pp. 5–10 (cit. on p. 2).

[3] SpaceWorks. *2020 Nano/Microsatellite Market Forecast, 10th Edition*. 2020 (cit. on p. 2).

[4] Kyle T. Alfriend, Deok Jin Lee, and N. Glenn Creamer. "Optimal servicing of geosynchronous satellites". In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit* August (2002), pp. 1–9. DOI: 10.2514/6.2002-4905 (cit. on p. 2).

[5] Dario Izzo et al. "Evolving solutions to TSP variants for active space debris removal". In: *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference* (2015), pp. 1207–1214. DOI: 10.1145/2739480.2754727 (cit. on p. 2).

[6] Nicolas Bérend and Xavier Olive. "Bi-objective optimization of a multiple-target active debris removal mission". In: *Acta Astronautica* 122 (2016), pp. 324–335. ISSN: 00945765. DOI: 10.1016/j.actaastro.2016.02.005 (cit. on p. 2).

[7] Kristina Alemany and Robert D Braun. "Survey of global optimization methods for low-thrust, multiple asteroid tour missions". In: (2007) (cit. on p. 2).

[8] Bradley J. Wall and Bruce A. Conway. "Genetic algorithms applied to the solution of hybrid optimal control problems in astrodynamics". In: *Journal of Global Optimization* 44.4 (2009), pp. 493–508. ISSN: 09255001. DOI: 10.1007/s10898-008-9352-4 (cit. on p. 3).

[9] Jin Zhang et al. "Analysis of multiple asteroids rendezvous optimization using genetic algorithms". In: *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings* (2015), pp. 596–602. DOI: 10.1109/CEC.2015.7256945 (cit. on pp. 3–5).

[10] Jacopo Prinetto and Michelle Lavagna. "Elliptical shape-based model for multi-revolution planeto-centric mission scenarios". In: *Celestial Mechanics and Dynamical Astronomy* 133.1 (2021), pp. 1–24. ISSN: 15729478. DOI: 10.1007/s10569-020-10001-9 (cit. on pp. 3, 6).

[11] Jun Bang and Jaemyung Ahn. "Two-phase framework for near-optimal multi-target Lambert rendezvous". In: *Advances in Space Research* 61.5 (2018), pp. 1273–1285. ISSN: 18791948. DOI: 10.1016/j.asr.2017.12.025 (cit. on pp. 4, 5).

[12] MATLAB *Optimization Toolbox*. The MathWorks, Natick, MA, USA. 2020b. URL: https://it.mathworks.com/products/global-optimization.html (cit. on p. 5).

[13] P. Fortescue, G. Swinerd, and J. Stark. *Spacecraft Systems Engineering*. Wiley, 2011. ISBN: 9781119978367 (cit. on p. 6).

[14] Stefano Silvestrini et al. "Design of Robust Passively Safe Relative Trajectories for Uncooperative Debris Imaging in Preparation to Removal". In: *2020 AAS/AIAA Astrodynamics Specialist Conference*. 2020, pp. 1–18 (cit. on p. 6).

[15] Judea Pearl. *Heuristics Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company, 1984, pp. 3–4 (cit. on p. 9).

[16] Christian Blum and Andrea Roli. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". In: *ACM Computing Surveys* 35.3 (2003), pp. 268–308. ISSN: 03600300. DOI: 10.1145/937503.937505 (cit. on p. 9).

[17] Víctor Martínez-Cagigal. *Multi-Objective Particle Swarm Optimization (MOPSO)*. MATLAB Central File Exchange. 2020. URL: https://www.mathworks.com/matlabcentral/fileexchange/62074-multi-objective-particle-swarm-optimization-mopso (cit. on p. 9).

[18] ESA. *Starlink Satellite Constellation of SpaceX*. URL: https://directory.eoportal.org/web/eoportal/satellite-missions/s/starlink. //accessed: 30.03.2021 (cit. on p. 14).

[19] Wiley J Larson and James Richard Wertz. *Space mission analysis and design*. Tech. rep. Torrance, CA (United States); Microcosm, Inc., 1992 (cit. on p. 14).

[20] ArieneGroup. *Electric Ion Space Propulsion Systems and Thrusters*. URL: https://www.space-propulsion.com/spacecraft-propulsion/propulsion-systems/electric-propulsion/index.html. //accessed: 30.03.2021 (cit. on p. 14).