Acta Futura 11 (2018) 57-70
DOI: 10.5281/zenodo.1139246

Acta Futura

GTOC 9: Results from University of Strathclyde (team Strath++)

Carlos Ortega Absil[†], Lorenzo A. Ricciardi[†], Marilena Di Carlo[†], Cristian Greco[‡], Romain Serra[†], Mateusz Polnik[†], Aram Vroom[‡], Annalisa Riccardi[†][*], Edmondo Minisci[†], Massimiliano Vasile[†]

[†] Department of Mechanical and Aerospace Engineering, University of Strathclyde, 75 Montrose Street, G1 1XJ Glasgow (United Kingdom)
[‡] Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft (The Netherlands)

Abstract. The design and planning of space trajectories is a challenging problem in mission analysis. In the last years global optimisation techniques have proven to be a valuable tool for automating the design process that otherwise would mostly rely on engineers' expertise. The paper presents the optimisation approach and problem formulation proposed by the team Strathclyde++ to address the problem of the 9th edition of the Global Trajectory Optimisation Competition. While the solution approach is introduced for the design of a set of multiple debris removal missions, the solution idea can be generalised to a wider set of trajectory design problems that have a similar structure.

1 Introduction

The Global Trajectory Optimisation Competition (GTOC) [1] is a yearly worldwide challenge that was initiated by the European Space Agency in 2005 with the aim of advancing the field of research on global optimisation techniques for space mission design. During the years the challenge has been the breeding ground for the testing and development of new computational intelligence techniques for the design of a variety of trajectory design problems. This year challenge, *The Kessler run* [2], has been to design a set of non-concurrent missions to deorbit 123 debris on Low Earth Orbit (LEO), requiring multiple launches within an available mission time frame. The only manoeuvres allowed to control the spacecraft trajectory are instantaneous changes of the spacecraft velocity. The problem objective function J is the sum over all missions of a constant term, the launch cost, and a quadratic term on the sum of propellant mass and de-orbiting kits required for the mission. Nevertheless, during the competition, the constant term was increasing linearly with submission time. Moreover constraints on propellant mass, minimum pericentre of all trajectory arcs, time between rendezvous and time between active missions have to be considered in the problem formulation.

The paper presents the optimisation techniques and the solution approach adopted by the team Strathclyde++ that ranked 6th over the 69 teams that registered to the competition (see Table 1, where $\mathbf{N_L}$ is the number of launches and $\mathbf{N_d}$ the number of debris removed). Section 2 is dedicated to present an overview

[*]Corresponding author. E-mail: annalisa.riccardi@strath.ac.uk

Team	$\mathrm{N_L}$	$\mathrm{N_d}$	score
JPL	10	123	731.2756
NUDT Team	12	123	786.2145
XSCC-ADL	12	123	821.3796
Tsinghua-LAD	12	123	829.5798
NPU	13	123	878.9982
Strathclyde++	14	123	918.9808

TABLE 1. *Final rank GTOC9*

on the overall problem solving methodology designed for the problem, Section 3 presents the different fidelity dynamical models used for the combinatorial search strategies, presented in Section 4, as well as final solution optimisation/local refinement, presented in Section 6. Section 5 presents an evolutionary approach adopted to recombine and improve solutions. Section 7 and 8 present results and conclusions.

2 Solution approach

The solution to the problem was found by using a three-step process that included both low fidelity and high fidelity models, as well as global and local optimisation solvers to converge to an optimal and feasible solution. As a first step, a Beam Search algorithm and a sequence patching method have been used to generate initial guesses (debris sequences and the initial guesses for the departure time from each debris) for multi-launch debris removal campaigns. The combinatorial algorithms used a low fidelity model to calculate the required ΔV for each transfer and estimate the final mission cost. A set of these solutions have been used as initial population for an evolutionary optimisation approach that, by optimising the times of transfers, was able to modify the order of the debris in the sequences themselves as well as to improve the distribution of initial mass among the launches. After the generation of these first-guess campaigns, a second step was used to obtain the solution in the required format, i.e. specifying every ΔV impulse required. In this step, for each debris-to-debris transfer returned by the combinatorial search, the time of application of each impulse as well as their magnitude and direction has been obtained by means of global and local optimisation algorithms using different fidelity models. The bounds on departure time from each debris and ΔV components have been set based on the values returned by the combinatorial search, and constraints applied in a strict sense regarding mission time, position and mass. These sets of mission trajectories constituted final solutions to the problem. As a third step, the entire launch sequence has

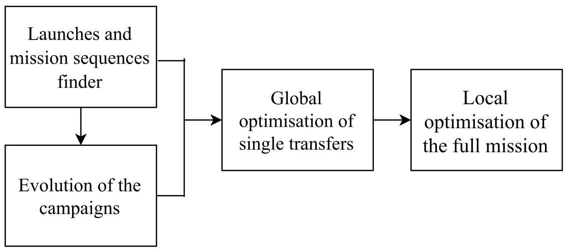
Launches and mission sequences finder
Evolution of the campaigns
Global optimisation of single transfers
Local optimisation of the full mission

FIGURE 1. *Flowchart of the solution approach.*

been optimised locally with the high fidelity dynamical model to exploit the correlation between subsequent transfers and reduce the propellant consumption further while ensuring that the constraint tolerances were met.

A flowchart of the solution approach is shown in Figure 1. In the following sections, the various components of this approach are described in detail.

3 Impulsive models

Low Fidelity estimation

In order to have a good but fast approximation of the cost of a transfer between pairs of debris, a low fidelity model, neglecting the J_2 perturbation, was used. For the sake of simplicity, the transfer was divided into two parts: an in-plane part with associated cost ΔV_i, modifying only the shape of the orbit, and an out-of-plane part, changing only the direction of the angular momentum for a cost of ΔV_o. The phasing was not included as it comes with no extra cost under Keplerian dynamics assuming there is no time constraint on the rendezvous. Given the low eccentricity of all the debris, the departure and target orbits were approximated to be circular. Thus the cost of the in-plane part can be obtained from a classic Hohmann transfer:

$$\begin{aligned}\Delta V_i = \quad & \left|\sqrt{\frac{2\mu}{r_1} - \frac{2\mu}{r_1 + r_2}} - \sqrt{\frac{\mu}{r_1}}\right| \\ & + \left|\sqrt{\frac{\mu}{r_2}} - \sqrt{\frac{2\mu}{r_2} - \frac{2\mu}{r_1 + r_2}}\right|,\end{aligned}$$

where μ is the Earth gravitational constant, while r_1 and r_2 denote the radius of respectively the initial and final orbits. As for the change of plane, it was computed as a single manoeuvre modifying both the inclination and the right ascension of the ascending node at the same time [3]:

$$\Delta V_o = 2\sqrt{\frac{\mu}{r_*}}\sin\left(\frac{\Theta}{2}\right),$$

with

$$\cos(\Theta) = \cos^2(i_*) + \sin^2(i_*)\cos(\Omega_2 - \Omega_1),$$

where the starred variables are determined between 1 and 2 according to the minimal cost.

High fidelity computation

A high fidelity estimation of the cost of the transfer between pairs of debris was obtained by solving a constrained global optimisation problem using Multi-Population Adaptive Inflationary Differential Evolution Algorithm (MP-AIDEA) [4]. In order to reduce the number of variables and, therefore, facilitate convergence to the global optimum, the maximum number of allowed manoeuvres was set to $n_{\Delta V} = 5$ (despite the rules of the competition allowed a maximum number of impulses for transfers between debris was 7). It was proved empirically, on a subset of significative transfers, that such assumption was not deteriorating, but rather improving, the quality of the optimal solution found for same number of functions evaluations.

The vector $\mathbf{y}$ of optimisation variables for the global optimisation problem includes the time of applications of each impulsive manoeuvre and the three components of the $\Delta\mathbf{V}$ vector, for a total of $n = 4 \cdot n_{\Delta V} = 20$ variables for each debris to debris transfer. The constrained optimisation problem was formulated as:

$$\begin{aligned} \min_{\mathbf{L}\leq\mathbf{y}\leq\mathbf{U}} \quad & F(\mathbf{y}) = \sum_{i=1}^{n_{\Delta V}} \Delta V_i(\mathbf{y}) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \\ & a_i(1-e_i) \geq 6600\,\text{km} \qquad i = 1,\ldots,n_{\Delta V} \\ & \mathbf{x}(t_f) = \mathbf{x}_D(t_f) \end{aligned} \tag{1}$$

where $\mathbf{y}$ is the vector encoding the 20 optimisation variables, $\mathbf{x}$ is the state vector of the spacecraft, $\mathbf{x}_D$ is the state vector of the targeted debris and t_f is the time at the end of the transfer. The second constraint imposes the perigee of the orbit of the spacecraft after each impulse to be higher than 6600 km. MP-AIDEA is run for a total of $n_{FEv} = 10^6$ function evaluations for each transfer. One function evaluation consists in the propagation from the initial time to the time of the first impulsive manuever, the application of the maneuver, a propagation until the time of the second manuever, and so on, until the final time t_f. For the first 7e5 function evaluations a non-expensive dynamical model was used, in which it was assumed that the spacecraft's mean orbital elements a, e and i remain constant between two impulses, Ω and ω change according to their secular variations due to J_2 [3], while M changes according to $M = M_0 + \overline{n}\,(t - t_0)$ where $\overline{n}$ is the mean motion perturbed by J_2 [5]:

$$\overline{n} = n\left[1 + \tfrac{3}{2}J_2\left(\tfrac{R_\oplus}{p}\right)^2\sqrt{1-e^2}\left(1 - \tfrac{3}{2}\sin^2 i\right)\right]$$

$R_\oplus$ is the Earth's radius and $p = a(1-e^2)$. The best solutions obtained at the end of this stage were then used to initialise the population for the next phase of the optimisation process, where the complete high fidelity dynamics, including osculating J_2 effects, was considered. In this phase, the dynamic equations were integrated with an 8-th order Adam-Bashforth-Moulton algorithm with a fixed step-size. At the end of the global optimisation, a local search was run from the best solution obtained; the Matlab solver *fmincon* with active-set algorithm was applied to problem 1.

Figure 2 shows a comparison between outputs of the low and high fidelity models for a large number of different sets of inputs. On average, the former tends to overestimate the total ΔV for a transfer. However, there is still a number of outliers whose cost is significantly more expensive than predicted, motivating for a safety margin to be used in the broad combinatorial searches.

4 Combinatorial search

Full campaign

This section presents the algorithms used in the first step of the solution process to focus on the combinatorial component of the problem. At this stage, a solution is considered to be a list of couples $\{(D_j, t_j)\}$ defining the itinerary in terms of debris to visit and time of transfer, and with a predicted cost J. If it contains all the target debris, this is referred to as a first-guess campaign. By considering a new launch as a particular case of transfer, a complete first-guess campaign can be built incrementally in a tree-like fashion.

The approach presented in this section was used to generate first-guess campaigns that eventually consti-

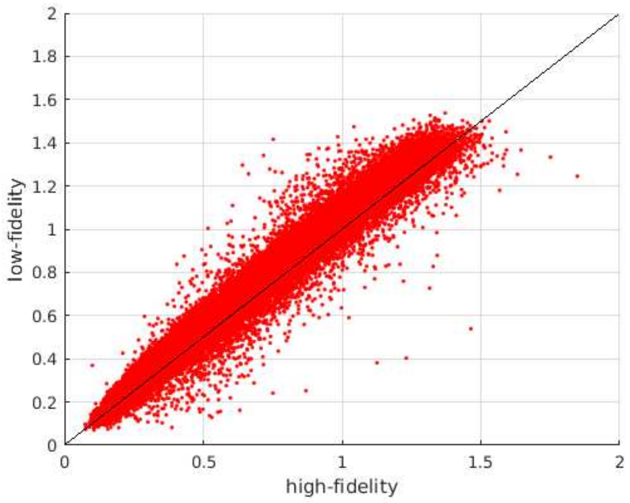

FIGURE 2. *Cost comparison (in meters per second) on 44683 cases between guess from Keplerian model and value from* J_2 *dynamics.*

tuted individuals in the population of the method presented in section 5, in some cases before and in some cases after refinement with the high fidelity models in Section 3 and the procedure in Section 6. This population contained the best first-guess campaigns found, but it contained as well sub-optimal and mass-infeasible solutions that presented remarkable features with respect to the best. Namely reduced number of launches, better homogeneity mass per launch, or overall longer missions. These were obtained with modifications on the baseline approach, that will be mentioned along the section.

Construction of the tree

A node S encodes a partial itinerary $\{(D_j, t_j)\ ,\ j \leq n\}$, the estimated ΔV cost of each transfer that is not a launch, a set of non-visited target debris NV and a set of available time instants for a new launch TL, where with n is noted the number of debris already visited in the partial itinerary. Branching of a node consists in appending to the itinerary the couple (D_{n+1}, t_{n+1}), with either

- a transfer to a debris $D_{n+1} \in NV$, satisfying the time and mass constraints associated to a transfer from $(D_n\ ,\ t_n)$,
- or a new launch to a debris $D_{n+1} \in NV$, with $t_{n+1} \in TL$,

and consequent update of NV and TL. Note this methodology advances chronologically in building the sequence of each single launch mission, but can decide to place a launch at $t_{n+1} < t_n$ if TL allows it. This is for example the case in which the sequences are wrapped in time as will be discussed in the next subsections

Beam Search

The base tree exploration heuristic of choice was the Beam Search (BS). Methodologies based on BS have been successfully applied in other GTOCs [6] [7]. This baseline was selected primarily due to the fact that upper bounds on its time and space complexity are easily controlled.

The Beam Search is a non-exhaustive search that is derived from the textbook implementation of Breadth-First Search (BFS) [8] by considering a fixed maximum number of nodes for branching at each level of depth. This number corresponds to the beaming factor Be, a hyperparameter of the process. Figure 3 illustrates a comparison of BS with BFS and Depth-First Search (DFS).

In addition to pruning at each level of depth, a pre-pruning at the parent level is also conducted, i.e. the number of branches of each node is limited by the branching factor Br. Br can be used to bound further the complexities of the search. Besides, this practice

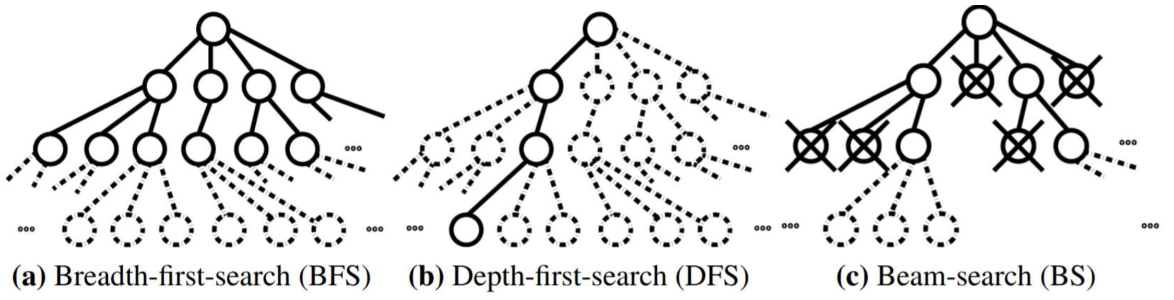

FIGURE 3. *Different tree search strategies in comparison. Dotted nodes are yet to be explored. Crossed out nodes are pruned and will not be branched [7].*

enforces that the offspring of at least Be/Br distinct nodes is represented in the next depth level.

Node fitness

Pruning requires definition of a sorting criterion for the nodes as a mean of prioritisation, i.e. a fitness function. This might or might not be the same for beaming and for branching, and might or might not involve a stochastic process. The definition of these criteria will condition deeply the performance of the search.

The baseline approach used a single fitness function, the quantity J_h, that represents the estimated cost of a hypothetical launch campaign that complies with the itinerary and needs an extra launch for each target debris of NV. This quantity was derived from the ΔV of each transfer as predicted by the Low Fidelity model in section 3.

Nevertheless, first-guess campaigns obtained with modified cost functions proved to be of special interest for the seeding of the approach presented in Section 5. Some examples of modified cost functions that found representation in the population that evolved into the final submission are listed below:

- Adding a penalisation on the standard deviation of the mass budget per launch, or of the ΔV budget per transfer.
- Sorting the nodes alphanumerically: first by the number of targets visited in the partial itinerary (decreasingly), then by the minimum number of targets visited in a single launch (decreasingly), and only then by J_h (increasingly).
- Considering several definitions of a per-debris rarity bonus: according to its appearance in a database of long single launch missions that exploit only close-to-optimal transfers, or according to its appearance in large clusters in a time-series clustering of the target debris RAAN.
- Computing J_h with an increased cost per launch (tripled).

Additional heuristics

In the Kessler run problem, a solution needs to visit all the target debris. This fact poses an issue for incremental approaches such as the ones described hereby; different launch missions will be in competition for a fraction of the reachable targets, hence greedy approaches risk to exhaust the search space in early iterations, leading to unexpensive single launch missions that cannot be aggregated to form complete campaigns. This effect was mitigated using heuristics that enforce some kind of diversity amongst the itineraries represented by the nodes branched at a given depth level, namely:

- *Pruning of twin transfers:* a limited number of transfers n_t to the same target debris is appended to node S during its branching. Also a minimum time separation Δt_t is enforced between each of them. This avoids an overpopulation of slight time variations of the same debris sequence. All results were obtained with $1 \leq n_t \leq 4$.
- *Pruning of twin campaigns:* a maximum number of nodes $n_s > n_t$ visiting the same subset of debris is branched at each depth level. n_s is automatically increased in case this criterion leaves less than Be candidates in the level, to avoid overpruning in single-root searches. This controls the

population of permutations of the same debris sequence. n_s has to be set in relation to Be, a typical value is $n_s = 20$.

Variations

Upon this baseline, a family of problem-specific techniques was conceived. These can be classified in two conceptual variations:

– *Cyclic Beam Search*: this variation considers, for the itinerary $\{(D_j, t_j)\,,\, j \le n)\}$, that TL only contains the first launch date available as imposed by t_n and the problem constraints. When this is unfeasible, TL is restored to contain the first available launch date in the mission timeline. In other words, if the Cyclic Beam Search is fed as root the itinerary $\{(D_0, t_0)\}$, the leaves will be first-guess campaigns that start at t_0 and wrap around time in a ring permutation of their chronological order.

– *Concurrent Beam Search*: a meta-algorithm on the method above, consists in a scheduler that manages the branching of N Cyclic Beam Searches. Each N-tuple of nodes shares NV in a competitive fashion, and each of them is assigned a segment of the mission timeline where it can search for transfers or launches. At each level of depth of the meta-algorithm, one of them is allowed to append a couple (D, t) to its itinerary.

Note these methods can be applied to either the computation of single launch sequences or complete campaigns, as well as to the expansion of partial itineraries if these are fed as roots. Few runs of the Concurrent Beam Search found solutions of better overall quality than few runs of the Cyclic Beam Search. However the increased computational cost of a single run and sensibility to inisialisation of the former translated into the team producing a larger variety of high-quality solutions with the latter. Over 90% of the first-guess solutions eventually used to seed the approach presented in section 5 were generated with the Cyclic Beam Search. Attempts at improving the launch heuristics were conducted, by selecting for TL values inferred from a database of very unexpensive transfers, and resulted in a drop of performance.

Initialisation

For the Cyclic Beam Search, the properties of the search space and algorithm allowed for a brute-force initialisation approach; a search was initialised with roots in the form $\{(D_0, t_0)\}$, with as many D_0 as target debris but a single t_0 for them all. This was repeated with t_0 in a monthly discretisation of the available mission timespan. This practice was found to give better results than initialising each search with various values of t_0. Furthermore, as data was gathered, heavier searches in terms of computational resources were conducted by increasing Be and Br, and priority of execution given to the searches with promising values of t_0. Some individuals were obtained by means of a set of light single-root searches.

For the concurrent beam search, even for moderate values of N, naïve initialisation of all sub-searches from all debris results impractical, since the possibilities grow combinatorially. To overcome this limitation, a multi-variate time-series clustering was conducted in the features $x_j = \cos(\Omega_j(t))\,,\, y_j = \sin(\Omega_j(t))\,,$ where $\Omega_j(t)$ is the RAAN of debris j at time t, and pre-pruning conducted in terms of size of the cluster. The clustering algorithm of choice was Partition Around Medioids, using segments of 75 days, Euclidean and Penrose distances and number of clusters selected by means of Silhouette Width in each segment. Searches were initialised randomly from $N = 3$ clusters. Yet other options considered for the Concurrent Beam Search but not explored in depth during the competition timeframe are its initialisation by means of N non-intersecting itineraries corresponding to independent launches, and solution of a single-objective optimisation problem for the designation of N launch sites in terms of RAAN and time of launch, maximising the total number of reachable debris.

Precomputations

All searches operated on a memory-loaded time-discretised precomputation of the ΔV cost of all debris-to-debris transfers, as predicted by the Low-Fidelity Model in Section 3. The resolution of the snapshots was of 0.6 days. With this modelling that does not take phasing into account, analysis pointed towards the underestimation of the time of flight as an important source of error, primarily in relation to the J_2 drift. Hence, a zero-order approximation of the time of flight was used as correction in the computation of the ephemerides of arrival. This time offset was set to 1.0 days. Furthermore, margins on the ΔV prediction and transfer windows were considered for seamless interaction with the subsequent of the solution pipeline; these safety parameters were tuned until the proportion of valid solutions after refinement was satisfactory.

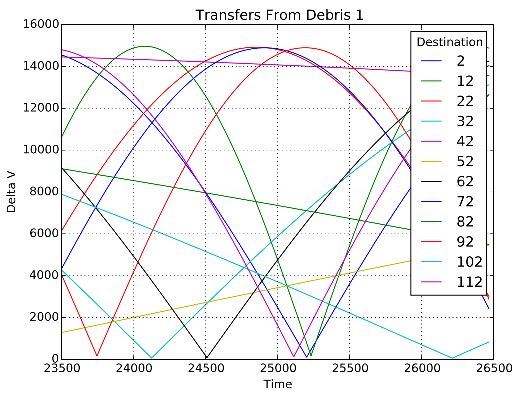

FIGURE 4. *Velocity required to reach a destination debris from a reference debris at a point in time.*

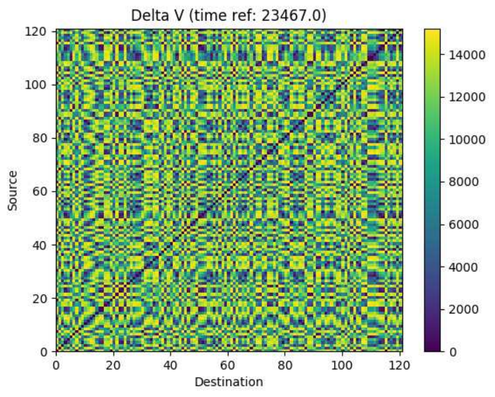

FIGURE 5. *Velocity required to transfer between debris at a reference time point.*

Sequence patching

The goal of a sequence patching is to assemble a launch campaign out of feasible sequences built from cheap debris to debris transfers. Such transfers occur periodically within certain windows over the course of a mission time schedule. For example, the figure 4 plots ΔV required to transfer from debris 1 to another debris considered in the competition. For readiness of the plot, every 10th target debris is reported in the figure. Moreover, as the figure 5 suggests, cheap transfers can be dispatched from other debris too. Therefore, all possible transfers below a predefined ΔV threshold were precomputed and aggregated into sequences. The occurrence of each debris in the database is sensitive to the choice of the threshold. Small treshold values can prevent certain debris to debris transfer to appear in the database, given the campaign time limits.

A sequence is described by time windows within its transfers occur and debris intended for removal. The order in which debris are visited is not relevant for the patching algorithm and can be established later using a cost optimiser.

Building a launch campaign can be modelled as finding a clique in an unidirectional graph $G(V, E)$ where V is the set of sequences and E the set of edges. Two sequences are connected by an edge if they target distinct subsets of debris and do not overlap in time. Finding a maximum clique is a well known NP-hard problem [9] with efficient solvers available open source [10]. With this approach we found that no full campaign can be patched using the data set of $85e4$ sequences. Larger datasets can be obtained by increasing the ΔV parameter. It has to be noted that, the clique construction approach can become impractical for datasets containing more sequences due to memory considerations. Such datasets were processed using a depth-first search.

To accelerate the patching algorithm sequences were sorted according to an index function that took into account a relative cost of a debris removal and its frequency among all sequences. Furthermore, the depth-first search was started from sequences that remove the rarest debris first to significantly reduce the number of sequences that later can be added to a partial campaign.

The results obtained from the sequence patching algorithm heavily depend on the quality of the initial data set. Final campaigns obtained from patching a data set containing $2.2e6$ elements covered up to 116 debris without launches dedicated for a single debris removal.

5 Evolution of solutions

Limitations of Beam Search and Sequence Patching. In the last few days of the competition generating a more competitive campaign became extremely challenging. In the last few days of the competition generating a more competitive campaign became extremely challenging. The best submission so far, Solution 4 (2), was using too many launches, thus penalising the final score. Using different heuristics and heavier searches with the Beam Search allowed to generate heterogeneous campaigns of similar cost, but none of them was better than Solution 4, even though a number of them presented lower number of launches. Attempts at reducing the overconstrained combinatorial search resulted in first-guess solutions that did not respected the maximum mass constraints and the detailed trajectory optimisers weren't able to restore mass feasibility without a heavy increase in cost. Manually fixing those trajectories was time consuming and sometimes simply not possible, while using larger datasets for the Sequence Patching algorithm was becoming computationally intractable, even employing pruning strategies. At that point, as a last resort, an entirely different campaign generation approach was conceived taking into consideration the limitations of the other two and the difficulties encountered when further refining those solutions.

Since all the debris had to be visited, a strategy able to generate full campaigns was sought. This is because such approach had no embedded mechanism that was greedily promoting sequences of easy to reach debris at the expense of leaving out a few scattered and expensive ones. While a grid of 0.6 days, for the Beam Search, at start, was considered sufficiently fine yet not too much to be a problem, later the need to operate on a pre specified time grid seemed too restrictive. Hence an algorithm able to continuously optimise the times and deal with also a set of discrete optimisation variables (debris ID) was considered highly desirable, if at all possible.

Reformulation of the problem. To accommodate all these requirements, the low fidelity campaign building problem was reformulated as a constrained multi objective optimisation problem operating only on real variables:

$$\begin{aligned}
&\min_{\mathbf{L_t}\leq\mathbf{t}\leq\mathbf{U_t}} \mathbf{J}^*(\mathbf{t_s}(\mathbf{t})), \qquad \mathbf{t}=(t_1,...,t_j,...,t_{123})\\
&\text{s.t.}\\
&\mathbf{t}_s = sort(\mathbf{t})\\
&t_{s_1}\in M_1\\
&M_i=\{t_{s_{j+1}}|t_{s_{j+1}}-t_{s_j}\leq 30, t_{s_j}\in M_i\}\\
&5\leq t_{s_j+1}-t_{s_j}\leq 30 \qquad \forall s_j\in M_i,\forall M_i\\
&t_{s_k}-t_{s_l}\geq 43 \quad \forall l\in M_i,\forall k\in M_{i+1}
\end{aligned} \tag{2}$$

where t_i is the departure time from debris i, $\mathbf{J}^*$ is the bi-objective function that has as first objective the original objective function and as second objective the maximum mass constraint violation; M_i is the $i-$th mission and t_{s_j} is the jth sorted time of transfer, coming from the transformation $\mathbf{t}_s = sort(\mathbf{t})$.

Advantages of this formulation With this encoding, internally called Time Shuffler, each debris could be freely associated to a time between the minimum and maximum epoch allowed for the mission ($\mathbf{L_t}$ and $\mathbf{U_t}$). Sorting of the vector $\mathbf{t}$ allowed to automatically and implicitly define the overall sequence of debris visited (by storing the sort index vector), while the constraints allowed to automatically distinguish between different missions. In facts, once the times were sorted, missions M_i automatically emerged from the differences between consecutive times: sequences of debris separated each by less than 30 days defined a mission, while the union of missions defined a full campaign. As a result, all possible campaigns could be uniquely defined by the vector of times $\mathbf{t}$, without needing to explicitly track the debris IDs and thus no discrete variable at all. This also halved the number of optimisation variables, with a drastic reduction of the size of the search space.

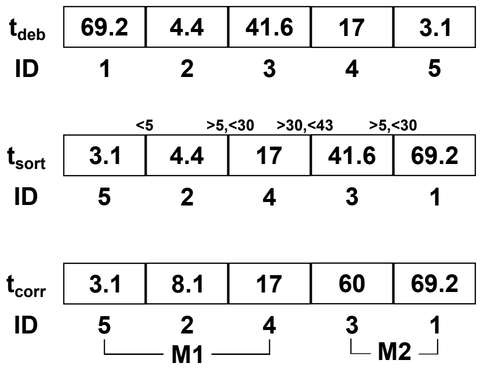

FIGURE 6. *Scheme of the Time Shuffler encoding and a possible time feasible solution*

Once the structure of the campaign was decoded, all time values could be simultaneously changed to satisfy the debris to debris and mission time constraints, provided no change in debris order was allowed. The 43 days of margin between missions included 5 days for the removal of the first debris of a mission (t_i represents the departure time, so the spacecraft has to arrive there 5 days before to apply the deorbiting kit) and 8 days of safety margin, as the transfers were considered instantaneous at this level but not with the full dynamics employed in the refinement stage. A graphical representation of the Time Shuffler encoding, together with a time feasible solution, is given in Figure 6.

Once the structure of a campaign was given and the time constraints were satisfied, it was possible to compute the resulting ΔV of each debris to debris transfer with the low fidelity estimation. Mass constraints were not directly imposed. Instead, the maximum mass vi-

Solution ID	Submission	N. Launches	$\hat{J}$	Relevant improvements in solution process
1	10 April	26	1713.07	Beam Search in the first 100 mission days.
				No thorough trajectory refinement.
2	20 April	18	1133.94	Cyclic Beam Search.
				Improved high fidelity model.
				Added single and multiple-shooting refinement.
3	24 April	16	1059.54	Improved Cyclic Beam Search heuristics.
				Improved low fidelity model.
				Improved global optimisation on high fidelity model.
4	26 April	16	1028.72	Further relaxation of search overconstraints.
5	30 April	14	967.49	Added evolution algorithm to solution process,
				Small population of best submitted solutions.
6	30 April	14	945.15	Multi-objective formulation of evolution
7	1 May	14	918.98	Larger population including diverse features.

TABLE 2. *Evolution of the solution process and quality of some of the submissions. Column $\hat{J}$ computed with $C_0 = 54.945$ as if submitted at the time of submission of solution 7 (best submitted).*

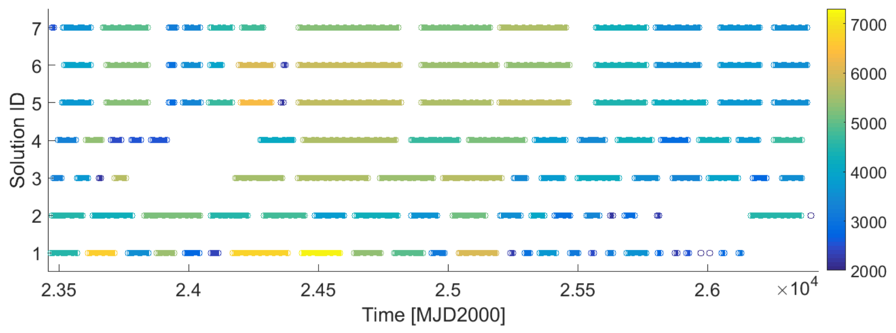

FIGURE 7. *Launches and debris removal epochs of the solutions in Table 2. Colour relates to initial mass of each of the launches.*

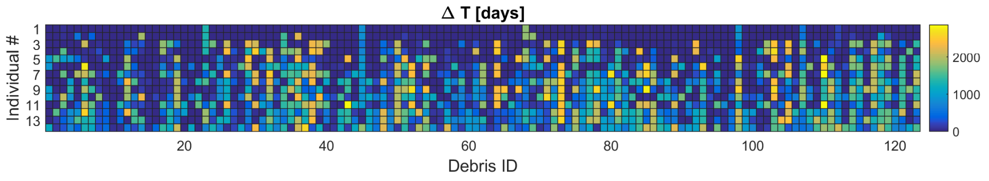

FIGURE 8. *Difference in time at debris between individuals in the initial population and final solution obtained after evolution. Most similar individuals on top.*

olation was considered as a second objective. The reason for this multi-objective approach was that this way a single run of the optimiser could return both the best mass feasible campaign, and a number of mass unfeasible campaigns with even better score, thus allowing us to choose which campaign to refine (i.e. improving an already mass and time feasible solution or attempting to make mass feasible a promising time feasible solution). Moreover, this was thought to have beneficial effects in the overall search, because promising search areas with temporarily mass unfeasible solutions were not getting outright discarded. Note that the multi-objective formulation was introduced only after a previous single objective approach managed to provide improved campaigns.

Implementation details Problem 2 was tackled with the MACS algorithm [11] with a bi-level approach: on the upper level, the evolutionary heuristics of MACS generated possible solutions $\mathbf{t}$, which were sorted, decoded and made feasible by a lower level simply enforcing the constraints and returning to the outer level feasible solutions with the original ordering, similarly to what was done in [12, 13, 14]. Initial trials were performed with totally random initial guesses for $\mathbf{t}$, and resulted in mass and time feasible campaigns with values of $\hat{J} \approx 1900$, rivalling submitted Solution 1 of Table 2 in just a couple of hours of runtime and no thorough trajectory refinement. MACS was then seeded with the best 14 solutions coming from the combinatorial search, including previously submitted solutions and promising mass unfeasible solutions, and was run for 10^7 function evaluations and standard parameters (for a runtime of approximately 6 hours). To get even better results, every 100 iterations of the outer level, the inner level did not just enforce time constraints but also performed a gradient based optimisation of the campaign cost function. Note that this gradient based refinement, with the low fidelity model but on the whole campaign simultaneously was only made possible by the Time Shuffler encoding. The fact that this reformulation of the original problem allowed us to evolve better solutions from those found by the Beam Search and that the detailed trajectory optimisers were then able to further refine those solution, confirmed that the whole approach was effective and solid. Unfortunately, this whole approach arrived too late in the competition, and since the trajectory refinement pipeline took approximately 6 to 8 hours of computational time, it wasn't possible to run it more extensively. Moreover, the generic metaheuristics employed in MACS were probably not particularly suited for this specific problem, so better performance could be expected with problem specific metaheuristics.

6 Solution refinement

As last step, the solutions of the single transfers between debris computed by the high fidelity model presented in Section 3 are refined by a local optimiser handling an entire mission of multiple transfers in order to meet the constraint tolerances and further reduce the propellant consumption. Two steps are employed for this process. In the first one the mission is optimised using a single-shooting method. For each transfer, the optimisation variables are the same ones defined in Section 3, but the total number of variables is now $n = 4\ N\ n_{\Delta V}$ where N is the number of transfers in the mission. The problem is solved using Matlab *fmincon* with the active-set algorithm.

In the second step, the solution obtained by the single-shooting is used as first-guess for a direct multiple-shooting algorithm, using WORHP as sparse nonlinear programming (NLP) solver [15], employed to reduce the numerical integration error and improve the convergence performance.

Each transfer between two debris objects is modelled as a multi-phase problem with discontinuous linking conditions, i.e. the instantaneous velocity change ΔV. In a single phase, there is no continuous control to optimise and also a single discretisation interval could be used. Nonetheless, m sub-intervals are introduced to reduce the integration errors and to enhance the numerical solution of the boundary value problem, restoring the original purpose of shooting techniques. In particular, this precaution was necessary because of long time-scale trajectories subject to a sensitive dynamics. Indeed, a single transfer could last up to 25 days, which translates in hundreds of revolutions in the fast LEO dynamics under the effect of the full J_2 disturbance. Hence, the number of free parameters per transfer sums up to $n = 4n_{\Delta V} + 6(n_{\Delta V} - 1)(m - 1) + 3(n_{\Delta V} - 2)$, where the first term describes the time and three vector components of the impulsive manoeuvres, the second one concerns the initial condition of each sub-interval within a single phase, while the latter deals with the position variables after each ΔV, i.e. the linking conditions on position. Successively, each transfer is connected to the next one by means of a coasting phase, i.e. the de-orbit phase at the debris, with continuous

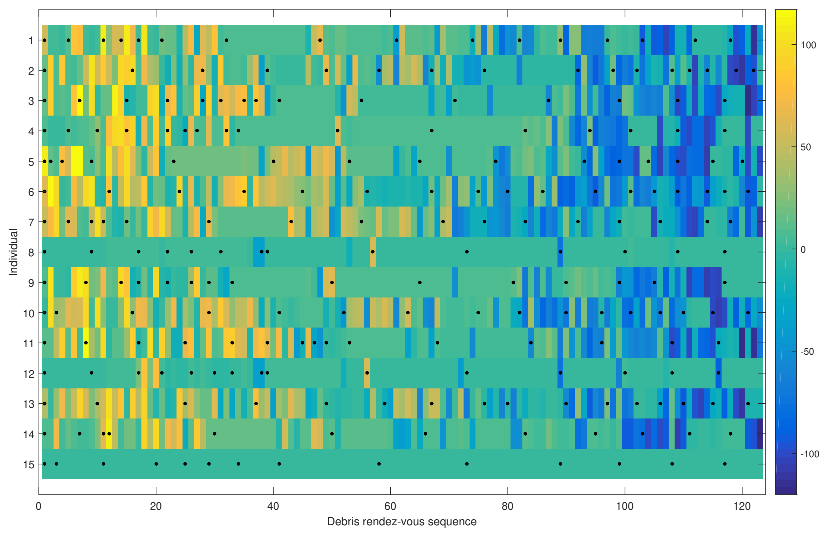

FIGURE 9. *Individual sequence similarity. Each row represents an individual's debris rendez-vous sequence, dots represent new mission launch. Colour details how many positions the rendez-vous with that debris ID is shifted in the sequence of the final submission. Individual 15 is the final submission.*

full linking conditions. In order to enhance the computational efficiency, the sparsity patterns of the associated *Jacobian* and *Hessian* matrices, resulting from the multiple-shooting transcription scheme, have been derived and exploited in the NLP step. The employed settings result in about 400 free variables per transfer, about 5% as percentage of non-zero elements for the objective's gradient, and lower than 0.1% for the constraint's Jacobian and Hessian matrices. Furthermore, the full-J_2 dynamical model has been augmented with the associated variational dynamics, and the system of equations numerically propagated using a Runge-Kutta 4 integrator, to compute the gradient information. This approach resulted in a decreased computational load and a more accurate derivative computation with respect to the finite-difference approach [16].

7 Results

Table 2 details the number of launches and cost of several of the solutions submitted ordered by submission date, together with the associated relevant improvements on the solution process. For fairness in the comparison, the cost is computed as if they had all been submitted at the time of the last submission.

Figure 7 details the mission timeline for the solutions in Table 2 as well as the initial mass of the spacecraft in each of the launches. It can be observed how an increase in the quality of the solution is associated to an increase in homogeneity in the mass of each of the independent launches and to better coverage of the mission time frame – note the gaps in the timelines of solutions 1 to 4. These two features derive from using incremental combinatorial approaches too greedy in terms of ΔV of each transfer, that lead to inexpensive missions that cannot be aggregated into competent campaigns. The gaps are caused by the search exhausting the available debris before exhausting the available mission time, thus failing to explore a region of the search space. Whereas solutions were generated using some of the heuristics detailed in Section 4 that mitigated this effect, this was always at the expense of the final objective function value. A similar phenomenon was encountered regarding the number of launches – solutions of as few as 13 launches yet suboptimal to Solution 4 were generated before Solution 5. This tendency ends with the introduction of the evolution of solutions in the pipeline, in Solutions 5 to 7. Besides a reduced number of launches and the lack of the aforementioned gaps in the mission timeline,

Mission	Start date	End date	Number of debris	Debris IDs
1	08/04/2064	18/04/2064	2	97, 44
2	20/05/2064	06/09/2064	8	109, 66, 28, 42, 102, 5, 72, 110
3	21/10/2064	14/04/2065	9	115, 7, 63, 67, 70, 48, 37, 104, 31
4	02/07/2065	02/08/2065	5	76, 52, 64, 53, 74
5	03/09/2065	02/11/2065	4	50, 118, 35, 113
6	09/12/2065	05/03/2066	5	114, 80, 116, 49, 117
7	11/04/2066	04/07/2066	7	34, 106, 26, 33, 2, 108, 6
8	16/11/2066	07/12/2067	17	4, 8, 43, 73, 55, 10, 9, 95, 65, 14
				93, 19, 90, 21, 100, 69, 30
9	03/03/2068	26/11/2068	15	81, 75, 87, 3, 45, 86, 105, 96, 46
				82, 41, 119, 57, 24, 32
10	01/01/2069	15/09/2069	16	1, 54, 62, 40, 89, 0, 99, 112, 15
				121, 59, 98, 27, 107, 20, 61
11	04/01/2070	17/07/2070	10	58, 23, 39, 122, 17, 12, 71, 16
				60, 68
12	24/08/2070	10/02/2071	9	13, 111, 120, 103, 94, 78, 85, 56, 83
13	28/04/2071	30/09/2071	9	25, 38, 77, 47, 11, 29, 101, 22, 91
14	21/11/2071	31/03/2072	7	18, 88, 36, 92, 51, 79, 84

TABLE 3. *Details of the final submitted campaign: start and end date, number of debris removed and debris' IDs.*

these campaigns also show an increased homogeneity in terms of initial mass of each of the launches. This proves the synergy obtained between the first and second stages of the solution process described in Section 2.

Figure 8 shows the difference in time of arrival at debris between the 14 individuals used by MACS as initial population for the evolution process, and the final solution submitted, Solution 7. Figure 10 presents the same information in terms of shifted positions by considering the debris rendez-vous sequence of each individual in chronological order. Bands of a similar colour indicate sections of the sequence that have been translated and/or permuted. Figure 9 details the zeroes of Figure 10, i.e. when the i-th rendez-vous in chronological order of a seed campaign matches with the final one. Note that many individuals are not mass-feasible initially, but present large similarity with the final submission, that was mass-feasible after applying the high fidelity models.

The time shifts in Figure 8 will alone define the itinerary of a campaign, hence these differences can be taken as a first indicator of the similarity of the chromosomes of different campaigns. It can be observed that there is mainly one individual that serves as backbone for Solution 7, although some other individuals present high similarity. Further analysis, as in Figure 10, confirms that a large part of the final submission itinerary can be traced back to a single individual by means of small shifts and permutations. This backbone individual is number 1 in Figure 8, number 8 in Figures 10 and 9.

The algorithm manages nevertheless to enhance the quality of the backbone individual, presumably by extracting information from other individuals in the population. For instance, the algorithm extracts several debris rendez-vous from the beginning of missions (new launches), and places them elsewhere in Solution 7. It also manages to insert a debris visit that required a dedicated launch within a short sequence. In a number of cases, the largest changes with respect to the backbone individual, can be traced back to smaller shifts and/or permutations with respect to other individuals. In other cases, the information flow is not apparent, as Solution 7 exploits some debris-to-debris transfers that are not represented in any individual of the initial population.

Figure 11 is a representation of the evolution in time of the RAAN of the final submitted campaign. It can be observed that the solution generally follows the natural J_2 drift as expected. Table 3 reports details of the 14 missions of the final submitted campaign. The final re-

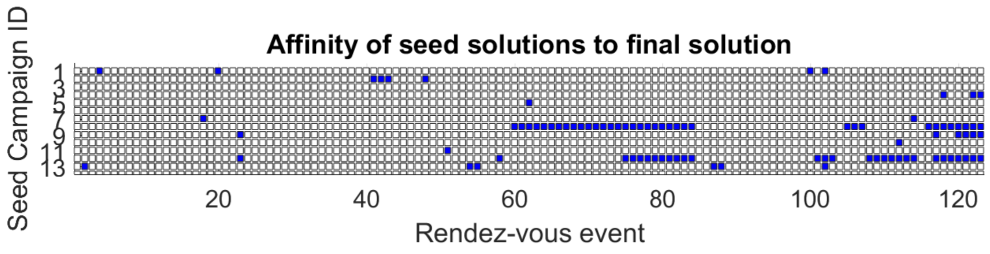

FIGURE 10. *Debris order affinity between seed solutions and final solution. A dot represents when the ID of the debris i-th rendezvous in chronological order of a seed campaign matches with the final one.*

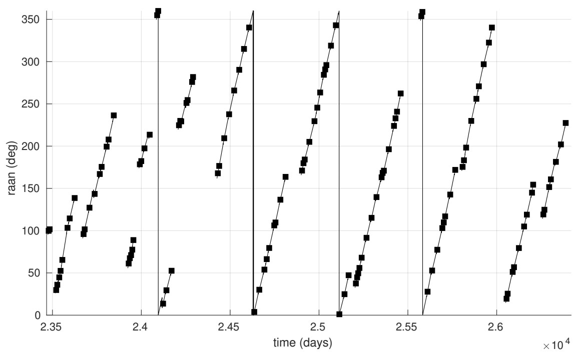

FIGURE 11. *Evolution in time of the RAAN of the final submitted solution.*

sults[1] and the complete seeding population debris IDs[2] can be downloaded from the provided links.

8 Conclusions

The paper presents the approach developed by team Strathclyde++ for the solution of the 9^{th} GTOC problem. The proposed methodology separates the combinatorial component of finding the optimal sequence of debris removals within the mission timeline, from the continuous problem of finding the best set of manoeuvres for each transfer and assuring feasibility. In particular, it introduces a continuous formulation of the combinatorial problem that allowed a population-based global algorithm to evolve a set of first-guess solutions and generate new ones, thus overcoming the manifest limitations of incremental combinatorial approaches. The fundamentals of the proposed methodology can be generalised to a family of multiple rendezvous problems, and are of special interest for the design of missions in which the set of available targets needs to be exhausted.

1 `http://icelab.uk/wp-content/uploads/2017/05/FinalSubmission.zip`

2 `http://icelab.uk/wp-content/uploads/2017/10/debris_order_evolutionary.xlsx`

9 Acknowledgment

The team would like to thank Archie-WEST, the supercomputing centre of West of Scotland for the support and computing resources, and the colleagues at the University of Strathclyde, namely Gianluca Filippi, Francesco Torre and Victor Rodriguez, for the always insightful discussions about the problem.

References

[1] GTOC portal. `https://sophia.estec.esa.int/gtoc_portal/`. Accessed: May 2017.

[2] D. Izzo and M. Märtens. The Kessler Run: On the Design of the GTOC9 Challenge. *Acta Futura*, 11:11–24, 2018.

[3] D. A. Vallado. *Fundamentals of astrodynamics and applications*. Springer Science & Business Media, 2001.

[4] M. Di Carlo, M. Vasile, and E. Minisci. Multi-population inflationary differential evolution algorithm with Adaptive Local Restart. In *2015 IEEE*

Congress on Evolutionary Computation (CEC). 25-28 May 2015, Sendai, Japan.

[5] K. F. Wakker. *Fundamentals of Astrodynamics*. Institutional Repository, Delft University of Technology, 2015.

[6] A. E. Petropoulos, E. P. Bonfiglio, D. J. Grebow, T. Lam, J. S. Parker, J. Arrieta, D. F. Landau, R. L. Anderson, E. D. Gustafson, G. J. Whiffen, P. A. Finlayson, and J. A. Sims. GTOC5: Results from the Jet Propulsion Laboratory. *Acta Futura*, pages 21–27, 2014.

[7] D. Izzo, D. Hennes, L.F. Simões, and M. Märtens. Designing Complex Interplanetary Trajectories for the Global Trajectory Optimization Competitions. In *Space Engineering: Modeling and Optimization with Case Studies*, pages 151–176. Springer, 2016.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2014.

[9] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The Maximum Clique Problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, 1999.

[10] Google Optimization Tools, Google Operations Research Team. `https://developers.google.com/optimization/`. Accessed: April 2017.

[11] L. A. Ricciardi and M. Vasile. Improved archiving and search strategies for Multi Agent Collaborative Search. In *International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control, EUROGEN, 2015*. 14-16 September 2015, Glasgow, UK.

[12] L. A. Ricciardi, M. Vasile, and C. A. Maddock. Global Solutions of Multi-objective Optimal Control problems with Multi Agent Collaborative Search and Direct Finite Elements Transcription. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*. 24-29 July 2016, Vancouver, Canada.

[13] L. A. Ricciardi, M. Vasile, F. Toso, and C. A. Maddock. Multi-Objective Optimal Control of Ascent Trajectories for Launch Vehicles. In *2016 AIAA/AAS Astrodynamics Specialist Conference*. 13-16 September 2016, Long Beach, CA, USA.

[14] M. Vasile and L. A. Ricciardi. A Direct Memetic Approach to the Solution of Multi-Objective Optimal Control Problems. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. 6-9 December 2016, Athens, Greece.

[15] D. Wassel, F. Wolff, J. Vogelsang, and C. Buskens. The ESA NLP-Solver WORHP - Recent Developments and Applications. In *Modeling and Optimization in Space Engineering*, pages 85–110. Springer, New York, 2013.

[16] S. Kemble. *Interplanetary mission analysis and design*. Springer, 1st edition, 2006.