DOI: 10.5281/zenodo.1142857
ISSN: 2309-1940

Acta Futura

Issue 11

GTOC9: The Kessler run

Advanced Concepts Team

http://www.esa.int/act

European Space Agency
Agence spatiale européenne

Publication:	Acta Futura, Issue 11 (2018)
Editors-in-chief:	Dario Izzo Luís F. Simões
Associate editors:	José M. Llorens Montolio Pacôme Delva Francesco Biscani Camilla Pandolfi Duncan James Barker Guido de Croon Daniel Hennes
Published and distributed by:	Advanced Concepts Team European Space Technology Center (ESTEC) 2201 AZ, Noordwijk The Netherlands http://www.esa.int/gsp/ACT/publications/ActaFutura Mail: act@esa.int

Cover image: An impression of the Low Earth Orbit debris environment. Copyright ©2018 European Space Agency

ISSN: 2309-1940
D.O.I.:10.5281/zenodo.1142857

Typeset with LaTeX 2_ε

Contents

Foreword

The ninth edition of the Global Trajectory Optimization Competition took place during April-May 2016. After a one month long registration period, scientists were presented with an interplanetary trajectory optimization problem with a complex solution landscape. The theme chosen to elaborate the competition problem was "active space debris removal" and the competition was named "The Kessler Run" after the famous chain effect that can lead to catastrophic consequences in overpopulated and uncontrolled orbital environments.. It is imagined that in the year 2060 a serious explosion triggered the Kessler effect compromising the Sun-synchronous orbital environment. Fortunately, not all is lost, it is imagined that scientists isolate a set of 123 orbiting debris pieces that, if removed, would allow to restore the orbital environment functionalities. Participating teams were asked to design a series of space missions that could cumulatively remove all the debris pieces. Each mission cost depended on the spacecraft mass and a base cost that was set to be increasing during the competition timeframe as to stimulate early submissions and create a race effect.
The innovative format of the competition, with an online leaderboard where all teams could monitor the other teams performances and progress, created quite an enjoyable event which was both entertaining for outsiders and informative for the scientists who battled up to the very last second to secure a better solution score. A total of 320 scientists divided in 69 teams from 125 different institutions and 19 different countries registered to the competition. A total of 36 teams submitted a solution to the challenge and more than 1200 missions were planned to remove those 123 debris pieces.
This special issue of the Acta Futura journal gathers the contributions of the teams that accepted to write about the methods developed and used during the competition to design the missions that would cumulatively remove all debris pieces. The papers include contributions from all top ranking teams as well as from the competition organizers, allowing this issue to hopefully be of great interest in the years to come to mission designers as well as to the automated debris removal community.

Dario Izzo
(Competition Organizer)

Rank	Team Name	Missions	Debris Removed	mission cost in MEUR
1	Jet Propulsion Laboratory	10	123	731.2756
2	NUDT Team	12	123	786.21452
3	XSCC-ADL	12	123	821.37966
4	Tsinghua-LAD	12	123	829.57987
5	NPU	13	123	878.99821
6	Strathclyde++	14	123	918.9808
7	DLR	14	123	949.85389
8	Missions Learners	14	123	964.51134
9	The Aerospace Corporation	14	123	1004.4860
10	Team Jena	15	123	1022.9063
11	UT Austin	15	122	1044.1787
12	NJU Team	16	123	1047.9685
13	EFLMAN TEAM	14	119	1107.6936
14	CU Boulder	17	123	1150.8439
15	CAS-NUAA	14	123	1182.0632
16	MTU-UoM	16	122	1192.7433
17	NSSC-THU	16	122	1210.3333
18	Brute WORHP	18	123	1229.5475
19	The Goonies	15	122	1238.6396
20	NablaZeroLabs	16	123	1267.7501
21	TYSE	16	123	1336.8590
22	TM	18	123	1490.9659
23	Occitania	22	120	1493.8567
24	ARGoPS	20	123	1512.6017
25	Personal team	23	123	1588.5770
26	GO to space	20	112	1819.1391
27	UofI and Goddard	23	123	1951.6797
28	LSPirates	20	105	2164.2321
29	Astro-ASAP-UC3M	13	85	3141.1951
30	Cal Poly SLO	39	84	4467.8746
31	Team STAR Lab	12	57	4481.7781
32	Nicolas RAVE	13	18	6453.0254
33	National University of Colombia	2	7	6511.5471
34	MeltedCode	1	5	6594.1105
35	AMSS_GTOC	1	4	6619.3569
36	Bremen optimizers	1	2	6760.20

TABLE 1. *The final rankings achieved during the ninth edition of the Global Trajectory Optimization Competition (GTOC9).*

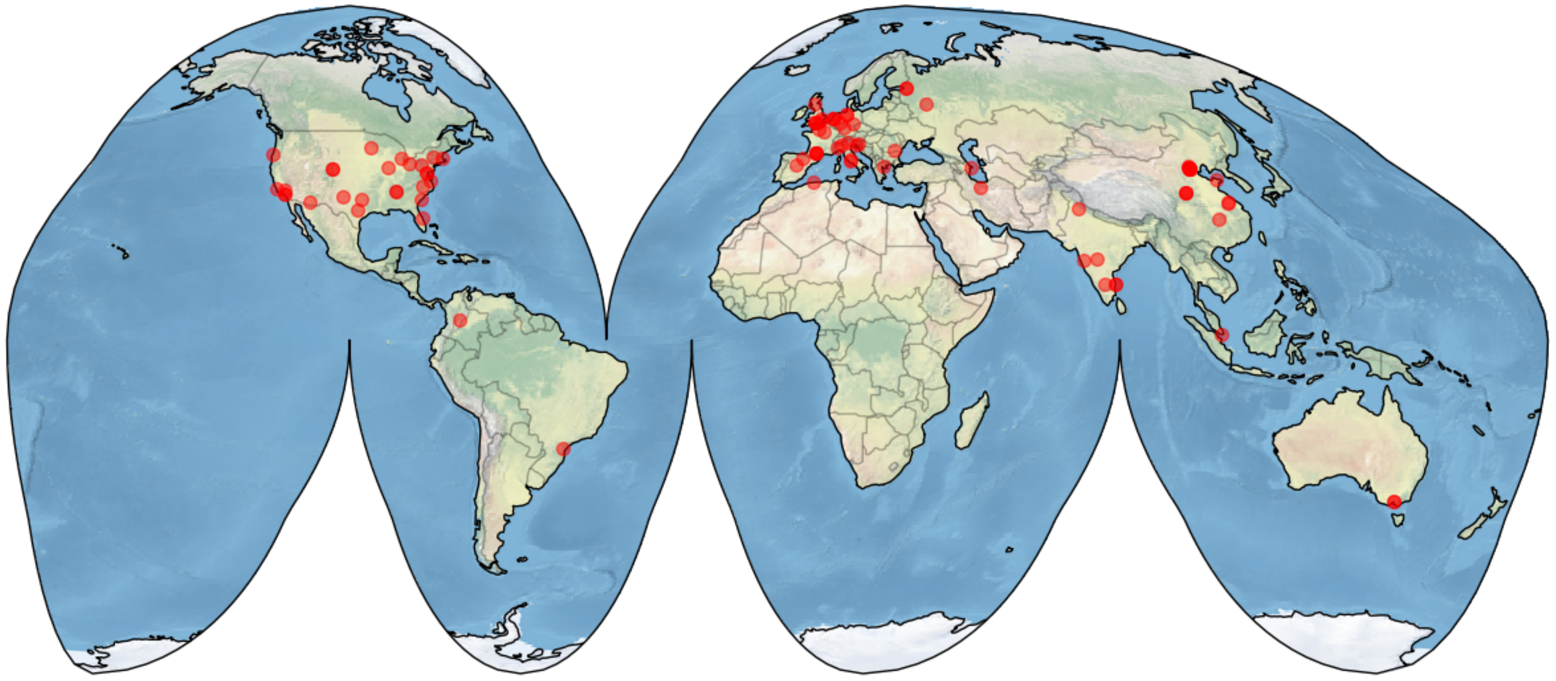


FIGURE 1. *The geographical distribution of the teams registered for the ninth edition of the Global Trajectory Optimization Competition (GTOC9).*

Acta Futura 11 (2018) 11-24
DOI: 10.5281/zenodo.1139022

Acta Futura

The Kessler Run: On the Design of the GTOC9 Challenge

Dario Izzo[†][*], Marcus Märtens[‡]

[†] Advanced Concepts Team, European Space and Technology Center (ESTEC)

[‡] Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Delft, The Netherlands.

Abstract. As organizers of the 9th edition of the Global Trajectory Optimization Competition we were tasked to design a challenging trajectory optimization problem to be solved during the duration of one month. We gave ourselves the goal to create a problem which was accessible enough for newcomers, deep enough for experts, unconventional in its objective function and which allowed for a real-time format of the competition. This document describes our design process from the initial idea up to the final form. We document some of our experiments conducted to learn about the solution landscape of an earlier simplified version of the problem and show how this exploration helped us in tuning the problem parameters to create a balanced and challenging task.

Figure 1. *The GTOC trophy as of May 2017.*

1 Introduction

The Global Trajectory Optimization Competition (GTOC) started in 2006 [1] with the intent to attract interest in the fascinating and difficult problem of optimizing interplanetary trajectories and to advance its methods. The winners are presented with a trophy they will keep up to the following edition (see Figure 1) and are asked to organize the following edition according to their own schedule and rules. During the years, private citizens, academic institutions and companies have competed in the various editions making GTOC an established and prestigious event. The GTOC web portal [2] collects and presents various resources and features an exhaustive collection of scientific publications related to the various competitions, which shows how it has stimulated the research in this field.

As winners of the 8th GTOC, organized by the Jet Propulsion Laboratory [3, 4], we accepted the honour (and burden) to organize the following edition: GTOC9. Starting from our experience with the complexity of the

[*]Corresponding author. E-mail: dario.izzo@gmail.com

design of multiple debris removal missions [5] we decided to design the GTOC9 problem around that scenario, trying to leverage on the under-explored challenges that the differential motion of the right ascension of the ascending node (RAAN) introduces when multiple debris have to be selected for removal.

This document describes the trajectory problem chosen and its evolution as we tuned its various parameters to ensure a challenging and interesting event. It is organized as follows: Section §2 introduces our main goals in developing the challenge. The following Section §3 presents the finalized formal mathematical description of the problem. Section §4 describes the results of a dry-run we made on a simplified (and preliminary) version of the problem and the lessons learned from that exploration of the solution landscape. We then conclude in Section §5 discussing briefly the design of the real-time aspect of the event.

2 Problem Requirements

When we started thinking about the GTOC9 challenge, we defined a set of goals by inheriting some from previous editions organizers and some from our own view on what GTOCs can (and should) be:

1. From a mathematical point of view, the problem has to be a global optimization problem with multiple local minima. Its complexity has to make it highly unlikely for the different participants to produce the same solution so that the final spread of returned trajectories should be as diverse as possible.

2. The objective function needs to be unusual in the sense that many of the problem details should be perceived as novel to ensure that no participant has a clear advantage because he has worked on some similar issues before.

3. The development of novel ideas and use of original algorithms should give a competitive advantage with respect to the reuse of well established methods. No standard approach should be immediately applicable and different strategies should be equally likely to return promising solutions.

4. The mission design problem to be tackled should go beyond being an academic exercise, and aim for real world relevance

5. The entry level for participation should be as low as possible, allowing exploration of the problem at different levels of complexity by participants with different levels of expertise and effort invested within the given time span of 4 weeks.

6. The problem solutions have to be easily and objectively verifiable.

7. A clear winner has to be declared soon after the competition ends.

8. If possible, the competition format should be innovated.

We thus started from these objectives to design an active debris removal mission, relying on the knowledge that some instances of this trajectory design problem map to complex variants of the Travelling Salesman Problem and thus belong to the class of $\mathcal{NP}$-hard problems (nondeterministic-polynomial complexity) [5].

While a number of scientists addressed, also recently, the problem of the design of missions able to remove multiple debris at once [6, 7, 8, 9, 5] (to name just a few of the works consulted), we had the impression that there was still a good amount of space for improvement left in this specific area of trajectory design. In particular, the combinatorial problem of debris selection (i.e. the problem of selecting what pieces of debris to remove from a potentially large set) was far from being explored satisfactorily, especially for long removal sequences. Considering long sequences imposes a quite interesting challenge, since the differential J_2 effect results in the evolution of a complex RAAN distribution. From these thoughts, the basic idea for the GTOC9 problem was born.

3 The Kessler Run

To construct a scenario where the removal of long chains of multiple debris is relevant, we thought to look into a possible future scenario where the Kessler Syndrome [10] triggered a significant damage to an important, and crowded, orbital environment such as that of the Sun-synchronous satellites. The following storyline emerged:
"It is the year 2060 and the commercial exploitation of Low Earth Orbits (LEOs) went well beyond the trillion of Euros market size. Following the unprecedented explosion of a Sun-synchronous satellite, the Kessler effect triggered further impacts and the Sun-synchronous

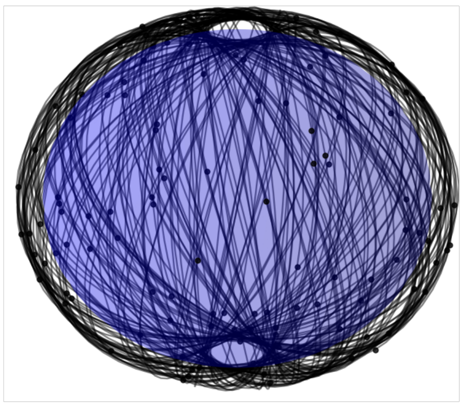


FIGURE 2. *Visualization of the M orbits of the M debris at some fixed epoch. The earth is visualized as the transparent blue sphere. See also `https://youtu.be/zvxZx-QnqQ0`*

LEO environment was severely compromised. Scientists from all main space agencies and private space companies isolated a set of 123 orbiting debris pieces that, if removed, would restore the possibility to operate in that precious orbital environment and prevent the Kessler effect to permanently compromise it. You are thus called to design a series of missions able to remove all critical debris pieces while minimizing the overall cumulative cost of such an endeavour.”[1]

Finally, we decided to name the competition “The Kessler Run” after a famous Star Wars [11] nonsensical quote from Han Solo claiming his ship, the Millennium Falcon, did “The Kessel run in less than 12 parsecs”.[2]

Formal Problem Definition

We report here parts of the original document [12] containing the official description of the problem which was made public and sent to all 69 teams who registered to GTOC9 by the 1st of April 2017.

Design n **missions** able to cumulatively remove $M = 123$ orbiting space debris moving along Keplerian orbits perturbed by the mean J_2 effect as detailed in Appendix A. Figure 2 shows a visualization of those orbits while Figure 3 presents the histograms for their orbital parameters.

One **mission** is a multiple-rendezvous spacecraft trajectory where a subset of size N of the M orbiting debris is removed by the delivery and activation of N de-orbit packages. In between debris visits, the spacecraft dynamics are Keplerian and subject to the full J_2 perturbation. The equations of motion are reported in Appendix A. The following cost function needs to be minimized:

$$J = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \left[c_i + \alpha \left(m_{0_i} - m_{dry} \right)^2 \right] \tag{1}$$

where C_i is the cost charged by the contracted launcher supplier for the i-th **mission** and it is composed of a base cost c_i (increasing linearly during the competition time frame) and a term $\alpha \left(m_{0_i} - m_{dry} \right)^2$ favouring a lighter spacecraft. At the beginning of the i-th mission, m_{0_i} denotes the (total) spacecraft mass and m_{dry} its dry mass. Each kg of launch mass saved thus results in a discount over the mission cost (but also in a less capable spacecraft).

The i-th mission starting epoch is denoted with t_i^s and its end epoch with t_i^f. A mission starts with a launch delivering, at t_i^s, one spacecraft at a chosen debris and ends when all the N de-orbit packages on-board have been delivered and activated. An orbiting debris is considered as removed if: a) its position and velocity vector at some epoch t coincides with the spacecrafts position and velocity vector and b) the spacecraft stays in proximity of the debris for the following $t_w \geq 5$ [days] while delivering and activating a de-orbit package of mass $m_{de} = 30$ [kg].

Afterwards, the spacecraft is free to ignite its propulsion system again and leave towards the next debris (note that only in-between debris transfers the spacecraft is subject to the full J_2 perturbation and its dynamics is described by the equations of motion reported in Appendix A. During the removal operations (i.e. for a time t_w) the position and velocity of the spacecraft must be considered to be those of the debris as computed by the ephemerides).

The basic cost c_i of each mission (i.e. not including the α term), increases linearly during the competition month and is computed as follows:

$$c_i = c_m + \frac{t_{submission} - t_{start}}{t_{end} - t_{start}} (c_M - c_m)$$

where $t_{submission}$ is the epoch at which the i-th mission is validated and t_{end} and t_{start} are the end and

[1]The decision to target 123 debris was, at this stage, purely arbitrary. Merely the result of keyboard proximity (the same way some people pick passwords such as “QWERTY”.

[2]https://www.youtube.com/watch?v=nmyvFEkJSE4

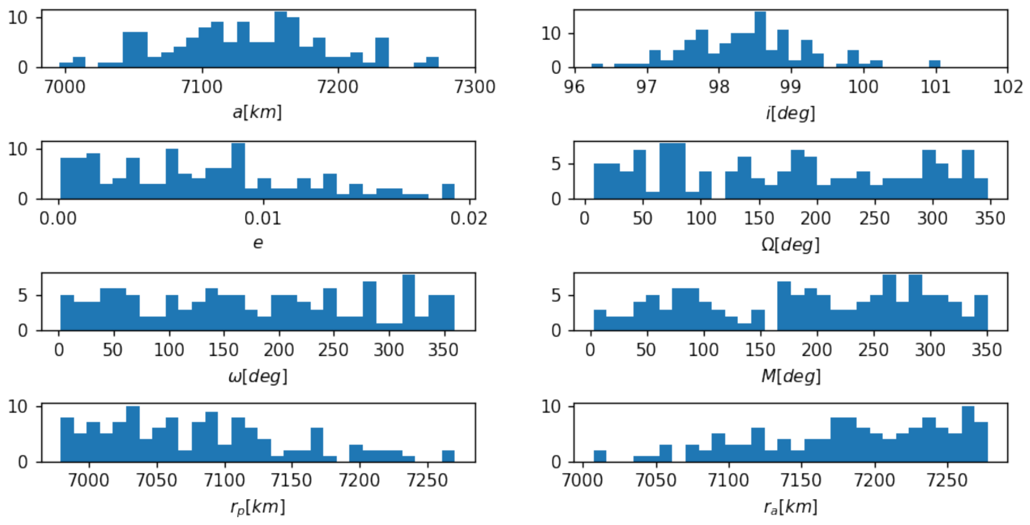


FIGURE 3. *Histograms for the various orbital parameters of the M debris orbits.*

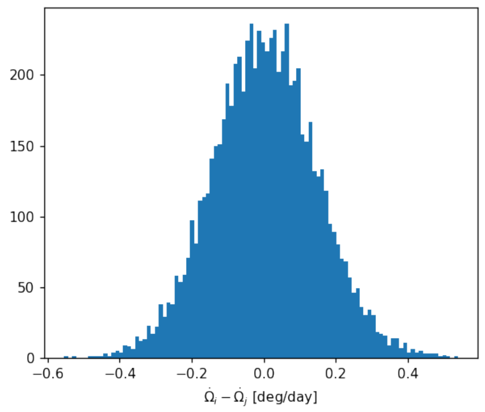


FIGURE 4. *Histogram for all differential RAAN drifts between pairs of debris.*

the beginning epochs of GTOC9. The minimal basic cost c_m is 45 MEUR, while the maximum cost c_M is 55 MEUR. Each orbiting debris not removed by any of the missions is considered, at the end of the competition, removed by a dedicated launch with a fixed cost of $c_{penalty} = 55.0018$ MEUR

Spacecraft

Each spacecraft's initial mass m_0 is the sum of its dry mass, the weights of the $N \geq 1$ de-orbit packages to be used and the propellant mass: $m_0 = m_{dry} + Nm_{de} + m_p$. All spacecraft have a dry mass of $m_{dry} = 2000$ [kg] and a maximum initial propellant mass of $m_p = 5000$ [kg]. Less propellant may be used, in which case the launch costs will decrease. Each de-orbit package has a fixed weight of $m_{de} = 30$ [kg].

Allowed Manoeuvres

The only manoeuvres allowed to control the spacecraft trajectory are instantaneous changes of the spacecraft velocity, its magnitude being denoted by ΔV. After each such manoeuvre, the spacecrafts mass needs to be updated by Tsiolkovsky's equation:

$$m_f = m_i \exp\left(-\frac{\Delta V}{v_e}\right),$$

where $v_e = I_{sp} g_0$. A maximum of 5 impulsive velocity changes are allowed within each transfer (leg) between two successive debris, excluding the departure and arrival impulse.

Operational Constraints

The debris removal operations during each of the multiple-rendezvous trajectories are complex and demand some control over the schedule of the debris visits:

1. The overall time between two successive debris rendezvous, within the same mission, must not exceed 30 days. Thus, if the arrival epoch at a debris a is t_a and the arrival to the next debris b is t_b, then $t_b - t_a \leq \Delta t_R = 30$ [days].

2. Different missions cannot be operated in parallel and a time of at least $\Delta t_M = 30$ [days] must be accounted for between any two consecutive missions so that $t_j^f + 30 \leq t_i^s$ [days] if $t_i^s > t_j^s$ for all $i \neq j$.

3. All mission events (arrivals, departures, maneuvers) have to take place in an allowed time window. Thus, for every event happening at epoch t_{event}, it has to hold that $23467 \leq t_{event} \leq 26419$ [MJD2000] (corresponding to a window of 8 years).

4. The osculating orbital periapsis r_p must not be smaller than $r_{p_m} = 6600000$ [m]. For simplicity, this condition is only checked immediately after arrival, departures and at deep space manoeuvres, but never in-between those events.

Table 1 summarizes the values of the problem constants and parameters used for GTOC9 and for the simplified version of the problem, which we introduce in the following section.

	value (simple version)	units
α	$2.0 \cdot 10^{-6}$ (0.0)	MEUR$/kg^2$
c_m	45 (55)	MEUR
c_M	55 (55)	MEUR
Δt_R	30	$days$
Δt_M	30	$days$
t_w	5	$days$
m_{de}	30	kg
m_{dry}	2000 (1000)	kg
m_p	5000 (2000)	kg
r_{p_m}	6600000	m
μ	$398600.4418 \cdot 10^9$	m^3/sec^2
J_2	$1.08262668 \cdot 10^{-3}$	$-$
r_{eq}	6378137	m
I_{sp}	340 (492)	sec
g_0	9.80665	m/sec^2
Day	86400	sec
Year	365.25	$days$
JD = MJD2000 + 2451544.5		$--$
MJD = MJD2000 + 51544		$--$

TABLE 1. *Problem constants and conversions. The values in parenthesis were used during a dry-run and were later adjusted. JD is the Julian date. MJD the Modified Julian Date and MJD2000 the Modified Julian Date 2000.*

4 Preliminary Solution Space Analysis

A concern in the design of a GTOC challenge is to ensure that the parameters of the selected problem are well balanced to provide an interesting experience. It seemed necessary for us to gain some insight into possible solution landscapes and an idea about the general complexity of the problem to be able to make informed choices on the parameters. Thus, we performed an internal one week long dry-run on a simplified version of the problem.

The Simplified Problem

During our dry-run, we neglected the quadratic and mass dependent term of the objective function by setting $\alpha = 0$. We also considered a fixed cost for each launch independent of the time of submission by setting $c_m = c_M$, effectively ignoring the competitive aspect of the problem to reward early submissions. Based on these decisions, the cost function in Eq.(1) reduces to its simplified version indicated with a subscript S: $J_S = n$, that is to minimize the number of missions needed for a complete removal of all debris.

During our preliminary exploration, we also relaxed the constraints on the total time of flight, by having $23467 \leq t_{event} \leq 27119$, effectively giving a 10 year time window to find solutions. The set of the $M = 123$ orbiting debris was also generated with a different distribution of orbital parameters (one of the outcomes of the dry-run was to introduce a more challenging set of debris orbits). To further simplify, we restricted the trajectories to avoid any deep space maneuvers and thus having the spacecraft only thrust at departure and arrival of debris. The launching systems defined by I_{sp}, m_{dry} and m_p were also different than in the final description (compare Table 1). Apart from these modifications, the basic challenge, i.e. finding a favorable combination of debris removal sequences, remained still the same.

A Set Cover Problem?

Consider the set $\mathcal{S}$ of all possible missions (i.e. multiple debris removal missions) that can be flown with the given spacecraft. The simplified problem is easily mapped to a set cover problem (see Appendix B) with some additional constraints: the universe U is the set of debris, while each valid mission within the time constraints defines a subset $S \in \mathcal{S}$ containing the debris it removed. As each debris can only be removed once, a disjoint set cover problem has to be considered and sets of conflicting missions (i.e. mission overlapping in time or violating the Δt_M time gap) must not be part of the solution. This can be incorporated in the integer linear programming (ILP) formulation of the set cover problem (see Appendix B) by adding additional inequality constraints, as detailed later.

We conclude that the simplified GTOC9 problem maps onto a disjoint set cover problem, whose solution results in the selection of a minimal amount of non-conflicting valid missions. Due to the $\mathcal{NP}$-hardness of set cover and ILPs in general, the computational complexity of the simplified problem is already high and brute force approaches are unlikely to be a feasible option. Clearly, it is impossible to compute the entire set $\mathcal{S}$, and even if it was possible the dimensionality of the corresponding set cover problem would be much too large to be directly solvable. Consequently, if one wants to follow this solution strategy, one has to reduce the problem-size to a small collection of subsets $\mathcal{S}$ reflecting possible debris removal sequences. Since each subset in $\mathcal{S}$ will correspond to an actual mission, we will interchangeably talk about subsets and missions, using $\mathcal{S}$ also as a notion for a collection of missions with the following, vaguely defined, qualities:

1. Each debris should be removed at least once by some element of $\mathcal{S}$.
2. There should be as little conflicting missions in $\mathcal{S}$ as possible.
3. $\mathcal{S}$ needs to be small enough to support a fast convergence of an integer linear programming solver.
4. $\mathcal{S}$ should be large enough to allow for the existence of good solutions.
5. The size of each set in $\mathcal{S}$ should be as large as possible since removing more debris within a few missions is more cost-effective than removing only a small amount of debris by a high number of missions.
6. $\mathcal{S}$ should also contain short missions since they are less likely to overlap and thus avoid constraint violations by making it easier to assemble disjoint sets for the final solution.

Some of these qualities are plainly contradictory and while the quality of the solution rises and falls with $\mathcal{S}$, it is far from trivial to generate a favorable collection of such missions.

We implemented a beam search algorithm to construct the elements of $\mathcal{S}$ which we call single mission beam search: BS. Beam search [13], [14] is a tree search algorithm which has been established as a helpful building block for solving past GTOCs [15]. The strength of beam search is that it allows to inspect large search spaces created by combinatorial decisions by ranking partial solutions and exploring only the most promising further.

By defining a beam width bw, only the bw best partial solutions are maintained at each level of the tree while all others are pruned. Each of those bw solutions is than expanded to compute multiple possible next steps (e.g. the next transfer to all reachable space debris) to generate the next level of the search tree, which will be pruned down to the highest ranking bw solutions again. Adjusting the beam width bw thus allows to balance solution exploration versus a given computational budget. A beam width of $bw = 1$ corresponds to a greedy search which only picks the optimal (partial) solution at each step. An unlimited beam width $bw = \infty$ corresponds to a breadth-first search that would exhaustively explore all possible combinatorial decisions.

Incremental Searches

Before using the BS to construct $\mathcal{S}$ and thus solve the resulting set cover instance, we first used BS, by comparison, to generate full solutions to the simplified problem. Beginning with the set of 123 debris as target set U, BS is used (starting at a random debris and at the earliest possible epoch) to remove the largest possible subset S_1 of debris. Afterwards, BS is called again on the reduced set of targets $U \backslash S_1$ to generate S_2, starting again from a random debris and at an epoch accounting for the Δt_M gap between missions. This approach is referred to as *incremental* BS.

Figure 5 shows the score distribution of various full solutions created by multiple runs of a greedy search (incremental BS with a beam width of 1) and incremental BS with a beam width of 30, denoted by

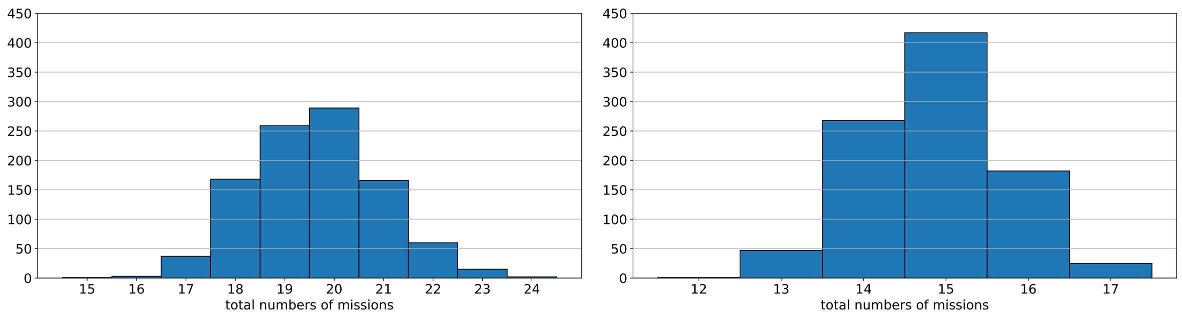


FIGURE 5. *Score J_S (total numbers of missions) achieved by 1000 different runs of (left) incremental greedy search and (right) incremental BS with $bw = 30$.*

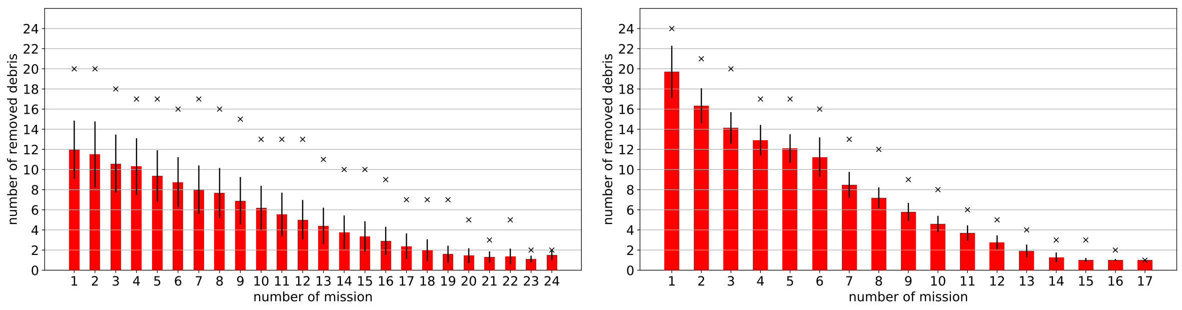


FIGURE 6. *Number of debris removed per mission in 1000 different runs of (left) incremental greedy search and (right) incremental BS with $bw = 30$. The black crosses mark the biggest positive outliers, i.e. a trajectory that would be most likely part of the final solution.*

BS30. The greedy search produces solutions with $J_S = 19.5(\pm 2.87)$ on average. Its best solution scores $J_S = 15$, but only 1 out of 1000 runs was able to discover such a solution. A small beam width of 30 is already enough to improve the quality: BS30 produces solutions with $J_S = 14.5(\pm 1.71)$. The top solution with $J_S = 12$ appeared only once out of 1000 runs.

Although it is possible to increase the beam width further in the hope to find better solutions, it comes at a higher computational cost and does not address a fundamental flaw of the incremental approach: While more and more debris get removed, efficient transfers between the remaining set of debris are becoming increasingly difficult to find. This difficulty arises because of the ascending nodes movement. Consequently, removing debris without a look-ahead strategy or some global perspective leaves debris clouds with evenly distributed RAANs, resulting in the average ΔV cost per transfer to increase tremendously.

Figure 6 shows the number of debris removed per successive mission in 1000 runs of the incremental greedy search and the incremental BS30. It is clear to see how the average length of the removal sequences decreases as the debris cloud becomes thinner. Note how this is the result of the search algorithm used, not a fundamental property of the problem solution. The globally optimal solution is, instead, expected to have no correlation between launch date and number of debris removed per mission.

Set Cover Beam Search

The issue of the thin debris cloud disappears if we take the set cover approach and thus use beam search no longer incrementally, but always on the complete debris cloud to generate a potentially overlapping and conflicting set of mission candidates $\mathcal{S}$. We grid the remaining time window for missions at b different time epochs and run, starting at each of those epochs, BS for each of the 123 starting debris, returning the k best trajectories to get a high quality sample of possible trajectories. This results in $123 \cdot b \cdot k$ missions constituting our collection $\mathcal{S}$

which is subsequently used to create a disjoint set cover problem instance.

Once we have the set cover instance, an ILP-solver can be applied to determine the *optimal* solution by selecting non-conflicting missions with disjoint removal sequences. However, our first attempts to solve the complete problem via this approach failed due to the large problem dimension which was unsuitable for the ILP solver we used: SCIP [16]. Thus, it was necessary to create instances of set cover which were still solvable in practice.

The Hybrid Strategy

Since the issue of thin debris clouds becomes only critical at the late stages of the incremental searches, we opted for a *hybrid strategy* applying first an incremental BS to reduce the size of the debris cloud and, finally, solving the remaining, lower dimensional disjoint set cover problem. In particular, we find the first 4 missions by incremental BS with a beam width of $bw = 100$, removing 21, 19, 20 and finally 11 debris. This leaves 52 debris to be removed and reduces the available time window by 1533 days (accounting for the ΔT_M time gap between missions). We divided the remaining time window equally into $b = 300$ epochs (which is roughly one new mission each 7 [days]) and started BS from each of these epochs and for each of the remaining debris as a start debris, returning the $k = 2$ best missions out of each run. Consequently, we considered 31200 missions, and thus an $\mathcal{S}$ with 31200 subsets defining the disjoint set cover problem.

To ensure a feasible final solution, we have to account for mission conflicts (i.e. missions that overlap in their time of flight or violate the Δt_M time gap). The direct way to approach this is to introduce one constraint for each pair (x_i, x_j) of possible (conflicting) missions, allowing at most one of them to be selected ($x_i + x_j \leq 1$). Given the large amount of sets created, we quickly realized that constructing this amount of constraints makes our set cover instances intractable. Thus, we had to relax our constraints as followed: once again, we divided the remaining time window, but this time equally in c epochs $t_1^*, \ldots, t_c^*$. For any epoch t, let $E(t) \subseteq \mathcal{S}$ be the collection of subsets from $\mathcal{S}$ whose corresponding mission takes place at t (accounting for the Δt_M time gap in between missions as well). Then, we can substitute the pairwise intermission constraints by a constant number of c constraints:

$$\sum_{j: S_j \in E(t_i^*)} x_j \leq 1 \qquad \forall i \in 1, \ldots, c$$

As a consequence of these new constraints, we might find *invalid* solutions, i.e. missions which overlap at epochs in between those selected c epochs. However, increasing c allows to minimize the amount of possible overlap while balancing the feasibility of the set cover instance by keeping the number of constraints low. We decided to set c to 200, which restricted the maximum possible overlap for mission times to roughly 10 [days]. As it turns out, the trajectories found by BS are most of the time stable enough to be moved a few days to the past or the future without compromising the set of removed debris. Thus, a violation of these new and softer constraints was (most of the time) repairable by some simple post processing of the selected trajectories.

To summarize, our hybrid strategy constructs a set cover instance of 31200 variables and 252 constraints. To solve this instance, we used a non-commercial solver named SCIP [16]. SCIP deploys pre-solving techniques by default before it tries to find the optimal solution for the LP relaxation. This pre-solving step simplified the instance to contain only 11355 variables and 233 constraints.

SCIP returned a solution corresponding to an ensemble of 11 missions when tasked with the disjoint set cover variant of the problem, which is already a slight improvement to the results obtained by incremental BS. However, after softening the problem to a basic set cover problem, a solution of only 10 missions was generated. There were only 4 debris removed multiple times within those missions. Three out of these defects could be repaired by simply dropping the last leg of a mission, as they were (coincidentally) removed as last debris. Only one debris was part of the middle legs of two missions. By exchanging one deorbiting of this debris by a deep space maneuver and applying some manual adjustments, this defect could be repaired as well. Additionally, due to the softer constraints, some missions in the solution generated were still conflicting as they had a gap Δt_M smaller than 30 [days]. After these conflicts were repaired by moving the conflicting missions a few days apart while maintaining their removal sequences, it was finally possible to obtain a valid solution removing all 123 debris with only 10 missions. The number of removed debris for those missions was $21, 19, 20, 11, 11, 8, 7, 10, 7$ and 9 debris at last, which shows that the thin debris cloud problem was successfully mitigated following this approach.

Figure 7 shows a visualization of the hybrid strategy and the resulting full solution. The gridded search that was used to create the sets for each remaining debris at nearby epochs is visible as the black area at the upper right corner.

An additional trick that was deployed to allow for better solutions was to have the last debris of some of the fixed 4 missions be variable. For example, if the last leg of the first mission could go to two possible debris, a or b, we saved two versions of this mission, one with the last leg to a and one to b. However, we left a and b as valid targets for later searches, artificially increasing the density of the remaining debris cloud. The sets of all possible combinations of those *optional* debris were used in the set cover problem as special variables, which could be picked without increasing the objective function while we still accounted for overlapping missions. The isolated horizontal lines of dots visible in Figure 7 show these debris, which were still part of the search in the set cover stage of our hybrid strategy, but were selected by the solver to be removed as part of the earlier missions.

Lesson Learned

The dry-run provided us the necessary feedback in terms of problem complexity and suitability for a GTOC to realize a number of important properties on the problem and its potential solutions. Firstly, we had reasons to believe that most solutions would be compressed in the $J_S = 9 - 11$ area which would then result in assigning the victory to the team submitting that score the earliest. We also did not want the final competition days to be spent to improve some top solution from (say) 9 to 8. So we decided to add the quadratic term ($\alpha > 0$) to the objective function and thus introduced an interesting trade-off between heavy and light spacecraft and, at the same time, have the score not directly linked to the number of submitted missions. We liked the idea that solutions with more missions could potentially score better than solutions with less missions.

A second change we made to the problem was to create a debris cloud with a larger average ΔV per transfer between debris pairs. This was done by enlarging the spread of inclinations, semi-major axes and eccentricities. We tried to tune this change such that it would be difficult to find a solution shorter than 12 missions while obtaining a good score.[3] The number of debris $M = 123$, proved of sufficient complexity, since exhaustive searches in the trajectory space were infeasible and the problem needed to be reduced in size before global solvers like SCIP could be applied to directly find solutions. Since our set-cover solution was taking 7.57 years to remove all debris, we now could also decide on the windows for the entire removal sequence and fixed it to 8 years. Finally, we choose the values for c_m, c_M and α so that the total cost increase during the competition of a solution with 12 missions would roughly cover the cost of one launch of a heavy spacecraft, i.e. ≈ 110MEUR. By doing so, we ensured that a team submitting its solution on the last day and cutting down the number of missions by one, while approximately keeping the same overall ΔV needs, would likely win over an early submission.

5 A real-time competition

One of the defining aspects of our challenge was certainly the on-line submission system, used for the first time for a GTOC event, which allowed for a very different (and hopefully enjoyable) experience. We had this idea in mind from the very beginning, as well as the awareness of the challenges and risks that coding an automated validation system brings forth in the field of optimal control and trajectory optimization. Fortunately we are also the creators and maintainers of the Kelvins on-line platform [17] which was created with the intend to host both data-mining competitions (for which the verification is a more standardized task) and algorithmic competitions (for which the verification has to planned case-by-case).

At the very beginning, we looked into automated validation of low-thrust trajectories, only to realize that we could not develop code reliably validating trajectories without making use of a detailed thrust profile (resulting in large files), or introducing tolerances that would be deemed as unacceptable. Fortunately the challenge we had in mind for GTOC9 was suitable also for a spacecraft powered by chemical propulsion and we thus decided to delay our study on automated validation of low-thrust trajectories and set-up the problem around that type of spacecraft.

A real-time competition set-up offers also new opportunities for the problem itself. While we were rather

[3]A hint for this could be found in the competition subtext: Can you do it in less than 12 parsec?

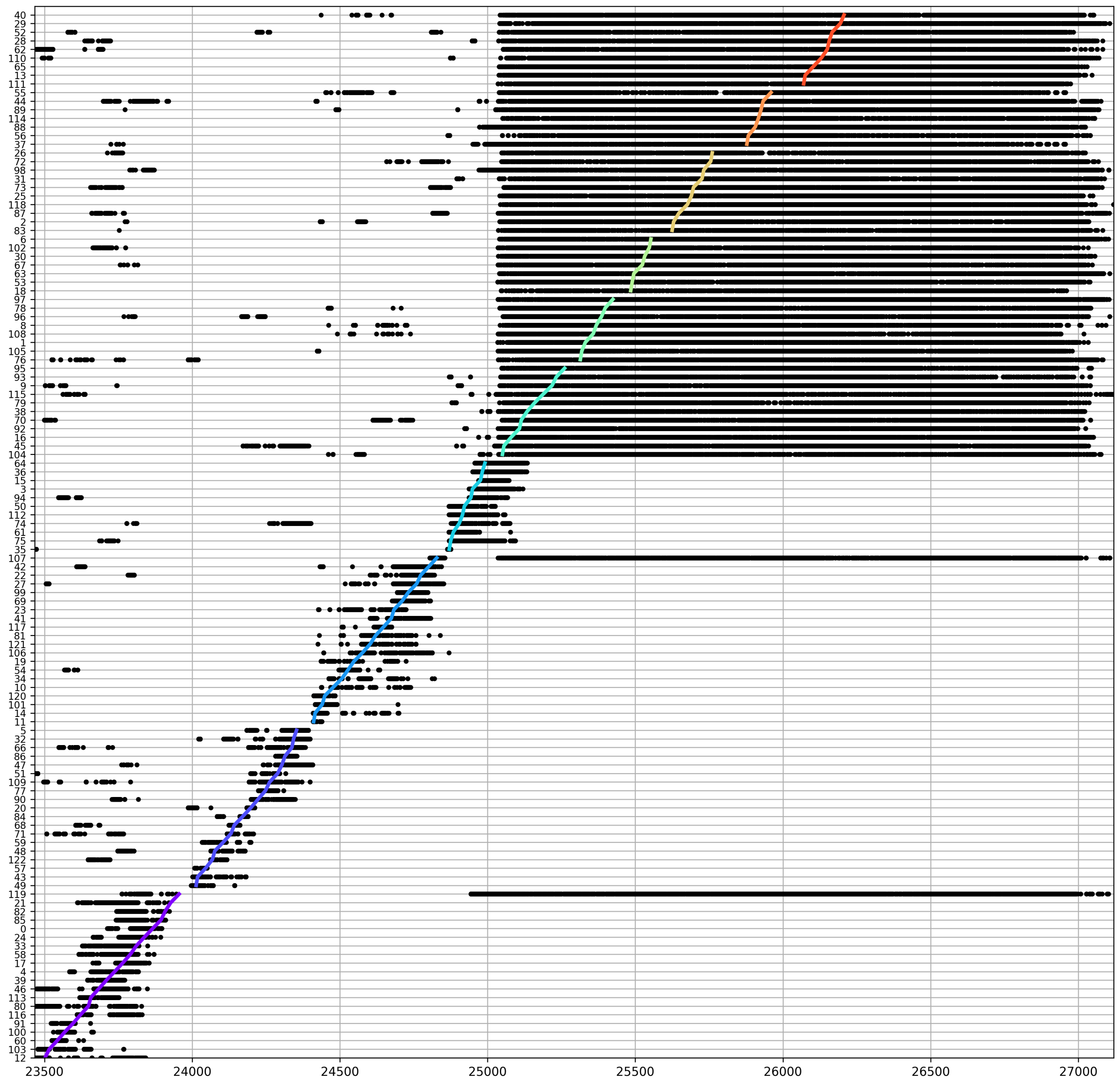


FIGURE 7. *Visualization of the full solution found by the hybrid strategy together with all possible debris removals generated for its construction. Vertical axis: id of removed debris, ordered in sequence of removal by the found solution. Horizontal axis: epoch in MJD. Each black dot marks the epoch of a possible removal of the corresponding debris as returned in one the runs of BS. Each colored line corresponds to one mission of the full solution. The first four missions (lower left corner) are found by incremental BS, picking the missions with the longest removal sequences out of* 1000 *runs. The remaining* 6 *missions were selected by solving the set cover problem. The set of possible missions* $\mathcal{S}$ *was built considering only the remaining debris.*

"conservative" in our final choice, we discussed the possibility of introducing a game theoretical aspect in the competition with teams having the possibility to influence past and future decisions of other teams (think for example of set-ups where the debris population is shared and needs to be collectively cleaned, etc.), or to go against specific teams at the cost of losing resources, thus creating the possibility of team coalitions, etc. Since possible team interactions were somewhat hard to predict, we were afraid that these dynamics could negatively impact the experience of some participants. Moreover, making the objective function dependant on the decision of all participating teams rather than having it fixed for each team would make it hard to study the problem in isolation. As a consequence, we refrained from this idea, but allowed for teams to get an early submission bonus and access to a leader-board. The leader-board provided real-time information on the current ranking of all teams and, perhaps even more importantly, reliable information about possible values of the objective function and the number of submissions that were used to obtain it. It turned out that our decisions on c_m and c_M were not game-breaking but just enough to motivate the teams to keep their trajectories updated and visible to all, creating an engaging race throughout the whole month of the competition.

6 Concluding remarks

The 9th edition of the Global Trajectory Optimization Competition turned out to be a great success, mostly thanks to the many teams who registered and worked hard on the challenge we created. Designing such a problem and making sure that our on-line submission system would be prepared for it, was a challenge we faced with great enthusiasm. We hope that the lessons we learned as organizers will be of use to future similar endeavours.

Looking back at our initial objectives, we believe we managed to achieve most of them satisfactorily and to provide the community with a new, complex and relevant benchmark in the field of interplanetary trajectory optimization.

Acknowledgements

Several people were involved in a preliminary brainstorming about the problem and in the later testing of the Kelvins web site and submission workflow. They actively and passionately proposed ideas and modifications which helped greatly, some of which found their way in the final problem set-up. We thank Luís F. Simões, Diego Escorial Olmos, Chit Hong Yam, Stefano Campagnola, Alessio Mereta and Martin Nisser. Stefano Campagnola, now working at JPL, participated only in very preliminary discussions and later we agreed (with great sadness) to exclude him as to avoid the risk of unintentional leaks to possible participants.

A Dynamics

Equations of Motion

Each spacecraft dynamics is described, between two manoeuvres, by the following set of Ordinary Differential Equations (ODEs):

$$\begin{aligned}
\ddot{\mathbf{x}} &= -\frac{\mu x}{r^3}\left\{1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(1-5\frac{z^2}{r^2}\right)\right\}\\
\ddot{\mathbf{y}} &= -\frac{\mu y}{r^3}\left\{1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(1-5\frac{z^2}{r^2}\right)\right\}\\
\ddot{\mathbf{z}} &= -\frac{\mu z}{r^3}\left\{1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(3-5\frac{z^2}{r^2}\right)\right\}
\end{aligned}$$

that describe a Keplerian motion perturbed by main effects of an oblate Earth, i.e. J_2. Note that between an arrival and a departure event the spacecraft is co-orbiting with the debris piece and hence its position and velocity is not described by the above equations, but coincides with the debris' orbit.

Debris Ephemerides

Each debris orbit is defined by the values $t_0, a, e, i, \Omega_0, \omega_0, M_0$ as read from the file distributed on the competition starting day (all files can be downloaded from the official portal [2]). At any given epoch t the position and velocity vectors of each debris piece must be computed by updating its osculating Keplerian elements using the mean motion and the precession rates and then converting, as in the Keplerian case, the updated osculating elements to position and velocity. Note that by doing so we are neglecting the velocity component deriving from $\dot{\Omega}$ and $\dot{\omega}$, for the purpose of this competition this is deemed as appropriate as it removes complexity from the equations without introducing any significant change on the search space landscape.

The procedure detailed below (assumes consistent units everywhere) shows all necessary equations.

1 - Computation of the Osculating Keplerian Parameters

After having defined the mean motion $n = \sqrt{\frac{\mu}{a^3}}$, the semilatus rectum $p = a(1-e^2)$ and the precession rates:

$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{r_{eq}}{p}\right)^2 n \cos i$$

$$\dot{\omega} = \frac{3}{4} J_2 \left(\frac{r_{eq}}{p}\right)^2 n(5\cos^2 i - 1)$$

compute the right ascension of the ascending node Ω from:

$$\Omega - \Omega_0 = \dot{\Omega}(t - t_0),$$

the argument of perigee ω from:

$$\omega - \omega_0 = \dot{\omega}(t - t_0),$$

and the mean anomaly from:

$$M - M_0 = n(t - t_0)$$

2 - Computation of Position and Velocity as in the Keplerian Case

The Kepler's equation is used to compute the eccentric anomaly E from the mean anomaly:

$$E - e \sin E = M$$

while the true anomaly θ can be obtained from the relation:

$$\tan\frac{E}{2} = \sqrt{\frac{1-e}{1+e}} \tan\frac{\theta}{2},$$

compute the flight path angle γ from:

$$\tan\gamma = \frac{e\sin\theta}{1 + e\cos\theta},$$

the norm of the radius vector from:

$$r = \frac{a(1-e^2)}{1 + e\cos\theta},$$

and the velocity norm from:

$$v = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}}.$$

The Cartesian coordinates of the position vector $\mathbf{r}$ and velocity vector $\mathbf{v}$ can then be computed from:

$$x = r[\cos(\theta + \omega)\cos\Omega - \\ - \sin(\theta + \omega)\cos i \sin\Omega]$$

$$y = r[\cos(\theta + \omega)\sin\Omega + \\ + \sin(\theta + \omega)\cos i \cos\Omega]$$

$$z = r[\sin(\theta + \omega)\sin i]$$

$$v_x = v[-\sin(\theta + \omega - \gamma)\cos\Omega - \\ \cos(\theta + \omega - \gamma)\cos i \sin\Omega]$$

$$v_y = v[-\sin(\theta + \omega - \gamma)\sin\Omega + \\ \cos(\theta + \omega - \gamma)\cos i \cos\Omega]$$

$$v_z = v[\cos(\theta + \omega - \gamma)\sin i]$$

B The Set Cover Problem

The set cover problem (sometimes denoted as *set covering problem*) was one of the original 21 $\mathcal{NP}$-complete problems proposed in the landmark paper[4] of Richard Karp [18]. It is a decision problem which asks, whether a set of elements can be "covered" by selecting k subsets of those elements out of given collection of those subsets. Formally, let the universe $U = \{1, 2, \ldots, n\}$ be a set of n elements, k be an integer and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ a collection of subsets $S_i \subseteq U$ for $i = 1, \ldots, m$, $m > k$. The set cover problem is to decide, whether there is a sub-collection $\mathcal{C} \subseteq \mathcal{S}$ of size k such that the universe U can be written as the union of all sets in $\mathcal{C}$:

$$U = \bigcup_{S \in \mathcal{C}} S.$$

In its optimization variant, the set cover problem is asking for the *smallest* number k to cover U completely. Karp showed by polynomial-time reduction from the k-Clique problem that there cannot be a polynomial time algorithm for solving the set cover problem if $\mathcal{NP} \neq \mathcal{P}$,

[4]Richard Karp introduced the concept of polynomial reducibility into the newly developing field of computational complexity theory. Thanks to this methodology, $\mathcal{NP}$-completeness of thousands of problems have been proven. Richard Karp received the highest award in computer science, the Turing Award, 1985 for his contributions to the theory $\mathcal{NP}$-completeness.

an unproven conjecture of long history in mathematics [19]. Given the common belief that this conjecture holds, solving large instances of the set cover problem with the means of nowadays computational power can be regarded as extremely time-consuming up to the point of intractability. This does not rule out the possibility, that special cases of the problem may be solvable, though set cover showed itself as a particularly difficult problem among the $\mathcal{NP}$-complete problems we know about. Under the assumption that $\mathcal{NP} \neq \mathcal{P}$, there exists no constant factor approximation of k for the set cover problem. In fact, it has been shown that approximating set cover within a factor of $(1-\alpha)\ln n$ for arbitrarily small $\alpha > 0$ is already $\mathcal{NP}$-hard [20].

The best known (polynomial) approximation algorithm for the problem is a greedy algorithm, which iteratively selects the subset S_i that covers the maximum of still uncovered elements of U. Despite its simplicity, it can be shown that this algorithm achieves an "optimal" approximation with respect to the given inapproximability results [21].

Integer Linear Programming Formulation

The set cover problem can be reformulated as the following integer linear program (ILP):

$$\begin{aligned} &\min \sum_{i}^{m} x_i \\ &\text{subject to: } x_i \in \{0,1\} && \forall i = 1,\dots,m \\ &\sum_{j:v\in S_j} x_j \geq 1 && \forall v \in U \end{aligned}$$

Each variable x_i corresponds to the subset S_i being selected or not. The first constraint ensures x_i to be binary, while the second constraint ensures that each element from U is covered by *at least* one subset S_i. If we modify this constraint to

$$\sum_{j:v\in S_j} x_j = 1 \qquad \forall v \in U$$

we enforce each element of U to be covered exactly once. This variant is called the *disjoint set cover problem*, since it demands that the solution sets are mutually disjoint.

References

[1] Dario Izzo. 1st ACT global trajectory optimisation competition: Problem description and summary of the results. *Acta Astronautica*, 61(9):731–734, 2007.

[2] ESA. The GTOC portal. `https://sophia.estec.esa.int/gtoc_portal/`. Accessed: 2017-05-01.

[3] Anastassios E. Petropoulos. GTOC8: Problem description and summary of the results. *Paper AAS 16-501, presented at the 26th AAS/AIAA Space Flight Mechanics Meeting, Napa, CA*, 2016.

[4] Dario Izzo, Daniel Hennes, Marcus Märtens, Ingmar Getzner, Krzysztof Nowak, Anna Heffernan, Stefano Campagnola, Chit Hong Yam, Naoya Ozaki, and Yoshihide Sugimoto. GTOC8: Results and methods of ESA advanced concepts team and JAXA-ISAS. *AAS 16-275, presented at the 26th AAS/AIAA Space Flight Mechanics Meeting, Napa, CA, arXiv preprint arXiv:1602.00849*, 2016.

[5] Dario Izzo, Ingmar Getzner, Daniel Hennes, and Luís F. Simões. Evolving solutions to TSP variants for active space debris removal. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1207–1214. ACM, 2015.

[6] Vitali Braun, A Lüpken, S Flegel, J Gelhaus, M Möckel, C Kebschull, C Wiedemann, and P Vörsmann. Active debris removal of multiple priority targets. *Advances in Space Research*, 51(9):1638–1648, 2013.

[7] Brent W Barbee, Salvatore Alfano, Elfego Pinon, Kenn Gold, and David Gaylor. Design of spacecraft missions to remove multiple orbital debris objects. In *Aerospace Conference, 2011 IEEE*, pages 1–14. IEEE, 2011.

[8] Max Cerf. Multiple Space Debris Collecting Mission-Debris Selection and Trajectory Optimization. *Journal of Optimization Theory and Applications*, 156(3):761–796, 2013.

[9] JT Olympio and N Frouvelle. Space debris selection and optimal guidance for removal in the

SSO with low-thrust propulsion. *Acta Astronautica*, 99:263–275, 2014.

[10] Donald J Kessler and Burton G Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics (1978–2012)*, 83(A6):2637–2646, 1978.

[11] Star Wars. Episode IV - A New Hope. *Dir. George Lucas. Twentieth Century Fox Film Corporation*, 1977.

[12] D. Izzo. *Problem description for the 9th Global Trajectory Optimisation Competition*, doi: 10.5281/zenodo.570193, May 2017.

[13] Roberto Bisiani. Beam search. *Encyclopedia of Artificial Intelligence*, 2(5), 1987.

[14] Christopher Makoto Wilt, Jordan Tyler Thayer, and Wheeler Ruml. A comparison of greedy search algorithms. In *third annual symposium on combinatorial search*, pages 129–136, 2010.

[15] Dario Izzo, Daniel Hennes, Luís F. Simões, and Marcus Märtens. Designing complex interplanetary trajectories for the global trajectory optimization competitions. In *Space Engineering*, pages 151–176. Springer, 2016.

[16] Gerald Gamrath, Tobias Fischer, Tristan Gally, Ambros M. Gleixner, Gregor Hendel, Thorsten Koch, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Sebastian Schenker, Robert Schwarz, Felipe Serrano, Yuji Shinano, Stefan Vigerske, Dieter Weninger, Michael Winkler, Jonas T. Witt, and Jakob Witzig. The SCIP optimization suite 3.2. Technical Report 15-60, ZIB, Takustr.7, 14195 Berlin, 2016.

[17] ESA. Kelvins - ESA's advanced concepts competition website. `https://kelvins.esa.int`. Accessed: 2017-05-01.

[18] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[19] Scott Aaronson. P=?NP. In *Open Problems in Mathematics*, pages 1–122. Springer, 2016.

[20] Dana Moshkovitz. The projection games conjecture and the NP-Hardness of ln n-approximating set-cover. *Theory of Computing*, 11(7):221–235, 2015.

[21] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

Acta Futura 11 (2018) 25-35

DOI: 10.5281/zenodo.1139152

Acta Futura

GTOC9: Results from the Jet Propulsion Laboratory (team JPL)

Anastassios Petropoulos*, Daniel Grebow, Drew Jones, Gregory Lantoine, Austin Nicholas, Javier Roa, Juan Senent, Jeffrey Stuart, Nitin Arora, Thomas Pavlak, Try Lam, Timothy McElrath, Ralph Roncoli, David Garza, Nicholas Bradley, Damon Landau, Zahi Tarzi, Frank Laipert, Eugene Bonfiglio, Mark Wallace, Jon Sims

Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Dr., Pasadena CA 91109, USA

Abstract. The removal of 123 pieces of debris from the Sun-synchronous LEO environment is accomplished by a 10-spacecraft campaign wherein the spacecraft, flying in succession over an 8-yr period, rendezvous with a series of the debris objects, delivering a de-orbit package at each one before moving on to the next object by means of impulsive manoeuvres. This was the GTOC9 problem, as posed by the European Space Agency. The methods used by the Jet Propulsion Laboratory team are described, along with the winning solution found by the team. Methods include branch-and-bound searches that exploit the natural nodal drift to compute long chains of rendezvous with debris objects, beam searches for synthesising campaigns, ant colony optimisation, and a genetic algorithm. Databases of transfers between all bodies on a fine time grid are made, containing an easy-to-compute yet accurate estimate of the transfer ΔV. Lastly, a final non-linear programming optimisation is performed to ensure the trajectories meet all the constraints and are locally optimal in initial mass.

*Corresponding author, Anastassios.E.Petropoulos@jpl.nasa.gov. Copyright 2017 California Institute of Technology. U.S. Government sponsorship acknowledged

1 Introduction

The 9th Global Trajectory Optimisation Competition (GTOC9) considered the problem of debris removal from the Sun-synchronous, Low-Earth-Orbit environment [1]. In this paper we describe the methods developed at the Jet Propulsion Laboratory to tackle the problem in the short, one-month competition timeframe. We also present the results obtained, both the submitted winning solution which removed all 123 debris objects in a campaign of ten missions, and results obtained shortly after the close of the competition.

An initial rough sizing of the problem was performed to understand the dynamics, estimate the ΔV sensitivities, and characterise the effect of number of missions on the cost function. The range of inclinations of the debris object orbits (about 96° – 101°) and semi-major axes (about 600 – 900 km larger than the Earth's radius) were sufficient to result in considerable variation in the drift rate of the ascending nodes (0.75°/day – 1.3°/day); the eccentricity, ranging from about 0.02 down to almost zero provided only a second order effect on the drift rate. The drift rate must of course be exploited in the transfers, because the ascending nodes are spread over the full circle and the cost of large plane changes is prohibitive (about 1.3 km/s per ten degrees). Drift-rate changes of about 0.1°/day can be achieved for about 100 m/s ΔV; for larger drift-rate changes, above about 0.2°/day, changing inclination is increasingly more effective (per unit ΔV) than raising apoapsis in affecting the node rate, and also permits an increase in the node rate, not just a decrease. The ΔV cost of matching the phasing and argument of periapsis of the debris was deemed of secondary importance.

Initial estimates of the number of missions that would be required were difficult to make, a fact reflected in the initial range of estimates that were made — from about 5 to 20. The base cost of a mission was comparable to the mass-dependent cost of a fully fueled spacecraft. Thus the trade-off between few missions with large ΔV requirements versus many missions with low ΔV requirements had to remain in play. This tradeoff was captured graphically a few days into the competition as shown in the Results section.

It was also realised early on that using the debris dynamics to propagate the spacecraft trajectory, rather than the full, unaveraged, J_2 dynamics, would be sufficiently accurate for good preliminary solutions. A variety of methods were then used to develop databases of body-to-body transfers, chains of bodies comprising a single mission, and complete campaigns of missions. The remaining sections of the paper describe these main facets of our solution methods, followed by a section on our results.

2 Propagation

A variety of spacecraft propagation methods were used, depending on the specifics of the broad-search method and the accuracy required. In the initial broad search, the debris dynamics, which correspond to the averaged J_2 dynamics, were used as an approximation of the spacecraft dynamics. In later searches, a correction to the mean motion, n, was incorporated [2]:

$$\dot{M} = n\left[1 + \frac{3}{2}J_2\left(\frac{r_{eq}}{a}\right)^2 \frac{1-\frac{3}{2}\sin^2 i}{(1-e^2)^{\frac{3}{2}}}\right]$$

When phase is relevant, it is important both to use the correction term to the mean motion and to propagate from initial values for the elements that are obtained from computing the mean orbital elements rather than from the osculating element values at a particular epoch. Doing so results in relatively small errors of about 5 to 50 km in position after ten days. An asymptotic solution to the main problem was also implemented but not extensively used. For full accuracy, the spacecraft equations of motion were directly integrated using the Gragg-Bulirsch-Stoer extrapolation, either in their given Cartesian form or in equinoctial elements. Since some broad searches used the accurate integrations, some effort was spent to optimise the integration speed.

For final optimisation (see Final Optmisation), a simple, 8-th order Runge-Kutta integrator was used for propagating the dynamics (via Cartesian states). Other dynamical models such as equinoctial elements or the averaged equations were considered but deemed not as convenient, accurate, or robust.

3 Body-to-Body Transfers

A number of body-to-body transfer techniques and databases were developed to approximate chains so that they could be more easily computed, or so that bodies could be easily added to existing chains using simple database lookups. These databases grew in accuracy and size as different methods were developed throughout the competition.

Transfer techniques

One method for estimating the body-to-body ΔV was coded in a subroutine called AF2. The goal of AF2 was to find debris-to-debris candidate transfer opportunities below a user defined ΔV threshold, for a range of departure and transfer times. The algorithm estimated total transfer ΔV by selectively adding or taking the root-sum-square of individual ΔVs required for matching node, inclination, periapsis, apoapsis, argument of periapsis and approximate phase change. Actual propagations in full J_2 dynamics were done (for node matching) to capture the effect of varying transfer time on ΔV. After refinements, the ΔV estimates from AF2 were found to be within 5% of the actual optimised transfer cost for most of the cases.

Another technique was based on the debris dynamics and provided an analytic estimate of the ΔV needed to match the semi-major axis, node angle, and inclination of the target debris in a specified transfer time. The ΔV was split between an initial ΔV to change the drift rate and a final ΔV to match the semi-major axis, node and inclination. The ΔV was split optimally between the two manoeuvres such that a linear combination of ΔVs needed to change each of those three elements individually was optimised.

A final technique involved making a quadratic fit of the plane-change ΔV as a function of time of the manoeuvre, which means that this simplified solution space needs little information to describe it. The actual departure and arrival time can be optimised later with relatively small adjustments to fix phase.

Databases

Throughout the competition, the team created many databases for quick lookups for estimating the ΔV for transfers between bodies. One of the first databases provided estimates of the transfer ΔV between all body pairs at one day intervals assuming debris dynamics and transfer durations of approximately one day. This estimate did not exploit changing the node rate, but intended to capture transfers between bodies whose orbit planes naturally drifted close to each other. The estimation included primarily a plane change at the relative node and one or two further impulses to change the eccentricity vector and semi-major axis. About 340,000 such transfers were found below 400 m/s.

Another database was also developed early in the competition that provided ΔV-optimal, full-phase transfers based on two-impulses using the full J_2-dynamics. At discrete times and for a set of discrete flight times, multi-revolution Lambert arcs were used as initial guesses to seed optimisation with Matlab's *fmincon*. (A similar database of optimal single-impulse transfers was also computed based on JPL optimisation software.)

The final database, named the GIGABASE, was perhaps the most reliable database created by the team during the competition. The GIGABASE was created roughly halfway through the competition and further developed in the final weeks. Transfers in the GIGABASE exploited the possibility of changing the node rate using ΔV and also matched phase, overall providing a good estimate of the optimal ΔV.

To quickly estimate the ΔV and flight time needed to transfer between debris objects, the GIGABASE assumed the spacecraft followed the dynamics of the debris. First derivatives (with respect to inclination and semi-major axis) of the nodal drift rate and the argument of latitude ($= \omega + \theta$) rate were used to approximate the required changes in inclination (δi) and semi-major axis (δa) for transfers with a given transfer time, a given integer number of revolutions, and with a given propulsive change in the right ascension of the ascending node, Ω. These simple equations require only solving a two-dimensional linear system of equations.

These initial δi and δa values are then differentially corrected to satisfy the full dynamics. The ΔVs to effect the differentially corrected initial δi and δa as well as the given change in Ω are computed assuming two-impulses. Subsequently, after the given transfer duration, another pair of impulses matches a, i and Ω of the debris (also θ has at this point drifted into alignment). Within the next orbit, the final two impulses match eccentricity and argument of periapsis of the debris.

Since the method is simple and fast, it is possible to loop over all possible number of integral revolutions and choose the lowest ΔV for each transfer. Furthermore, the simplicity of the computations allows looping over transfer times discretised in a fine grid, target bodies, departure times and departure bodies. Thus, a large database (the GIGABASE) was made with the following spacings: Debris, from every object to every object (123^2 = 15,129 options); departure epoch, 2-day spacing (1477 options); transfer time, 1 to 25 days, steps of 2 days (13 options). The resulting database contained about 290 million rows of data.

In order to facilitate the conversion of broad-search to detailed solution, a script called the decoderRing was created to reproduce more accurately the actual manoeuvre times and ΔV vectors approximated in the GIGABASE as a (good) initial guess into the final optimisation process. The inputs to this function were the bodies, departure epoch, transfer time, intermediate a, i and Ω values computed using the approximate method and stored in the GIGABASE. These were sufficient to allow analytic computation of all six manoeuvres, and a final one-dimensional corrections scheme added the missing secular term to the spacecraft J_2 dynamics. The decoderRing was only used when transitioning from campaign-level search to local optimisation and therefore did not need to be particularly fast.

4 Chain Building

Branch-and-bound

To construct low-ΔV chains (and sets of compatible chains), early in the competition a heuristic was developed that only considered plane change manoeuvres. Starting from an initial debris object at time t_0 a set of unvisited near-co-planar (to a tolerance) debris was identified. The minimum node difference was computed numerically, and a transfer of 1 – 2 days was initiated at the minimum. The following three cases were considered for when the minimum occurs: (i) within the feasible time interval, ΔV done to match inclination; (ii) before the feasible time interval (moving apart), immediate inclination change to alter the drift rate to match node in two days, and a second manoeuvre to match inclination; and (iii) after the feasible time interval (moving together), long stay at the initial debris object, apply drift rate change manoeuvre, and manoeuvre to match inclination two days later.

A branch-and-bound algorithm was used to build good chains of varying length with the heuristic ΔV. A grid of starting epochs and the set of unvisited debris initialised chain construction. A chain was deemed complete when there were no further targets, or when it ran up against the maximum time constraint. Filtering (bounding) was applied to chain length (minimum), total fuel, and average ΔV. The branch-and-bound algorithm was parallelized to quickly yield a database of low-ΔV chains over all epochs, with chain lengths from 3 – 22. The longest were used as backbones to seed building of campaigns.

Ant Colony Optimisation

A previous investigation applied Ant Colony Optimisation (ACO) to the removal of debris objects [3]; modifications were made to this to accommodate J_2 drift and the constraint that missions cannot overlap in epoch. Briefly, the ACO algorithm seeds "agents" at starting debris objects, where these agents then build chains of encounters by interspersing random steps and the directed following of "pheromone" trails laid by previous generations. When an agent reaches the defined propulsive limits for an individual mission, it resets by "launching" to a new debris object and begins again, repeating this behavior until a complete mission set is generated. At the end of each generation, the mission sets are evaluated using the GTOC9 cost function, with the best performing agents chosen to lay pheromones along the routes they followed. In order to differentiate among the multiple possible transfers between any debris object pair, the GIGABASE solution giving

$$\min \Delta t^{\xi} \Delta V$$

was selected, where ξ is a tuning parameter and Δt is the time interval between arrivals at the respective debris objects (prior to the GIGABASE, an earlier database was utilized in a similar manner). Time-varying aspects of the problem were addressed by seeding objects and initial chain epochs based on debris clustering information that created groups of common orbital elements at differing epochs across the available mission window. By the end of the competition, two main variants of ACO were employed: i) a "subset" routine that searched among a limited set of remaining debris objects and available launch windows in order to complete an existing set of missions, and ii) a "full" search that began at the end of the mission window and worked forward to build mission sets eliminating as many debris objects as possible. Throughout the competition, the "subset" approach reliably discovered the minimal set of additional launches to remove all remaining objects, with the complete sets then fed into the genetic algorithm and other refinement methods; by the end of the competition, the "full" ACO was reliably generating mission sets removing nearly all 123 objects for roughly 950 MEUR using 10-13 launches.

ACM

Built upon the ΔV estimation capability of AF2, ACM's primary task was to rapidly build chains using an algorithm which preserves diversity while preventing exponential increase in the number of solutions. This was achieved by using a hash-function based chain identification algorithm which kept equal proportions of best-in-time and best-in-ΔV solutions during the chain building process. Randomized additions were also done with a 1% chance of being accepted to the next length level. After tuning, ACM was able to generate hundreds of thousands of chains of length greater than 10 and up to 23 in a matter of few seconds.

5 Campaign Building

Campaign Beam Search

Starting with a backbone (or two), typically from the branch-and-bound method, partial campaigns were built using a Beam Search variant [4] with probabilistic mixing. Every generation (or depth) adds a new mission (chain), and the starting epochs for chain building are selected randomly from the currently valid time interval sets. After each highly-parallel generation evaluation, a subset of solutions are maintained for the next generation based on minimizing the heuristic campaign cost:

$$h = J + h_{\alpha} \frac{J}{D} (123 - D)$$

where J is the running GTOC cost function, h_α is a constant weight (scaling cost to go), and D is the number of de-orbited debris. Various knobs are available to ensure diversity and to prevent an overly greedy algorithm.

The chains and partial campaigns were advantageous in seeding combinatorial algorithms (e.g. ACO and GA), since they represent quality populations with an inherent reduction in the degrees-of-freedom (relative to the initial problem).

Chain Recombination

Given a database of body-to-body transfers or a database of chains, a directed graph was created where the nodes were defined by the debris object ID and time, and the edges indicated a transfer to a new object. Edges contained information like transfer time and ΔV. Chains were created by traversing the graph following different criteria: minimum ΔV, maximum chain length, mission time interval, whether or not the mission should contain a particular debris object, etc.

Given a database of chains, an undirected graph was created where the nodes contained a mission (*i.e.*, chain) and the edge indicated that two missions were compatible. A standard algorithm was used to search for cliques (a subset of nodes that are totally connected, that is, a subset of chains that are compatible). Only the cliques with the maximum number of debris were reported.

Chains visiting the same debris object at similar times would be connected in the directed graph, allowing "mixing-up". In this way, the graph can be traversed following an existing chain and at the common node it can transfer to another existing chain, creating in the process a new chain. "Re-jiggering" was often used to find better alternatives to a given mission: A directed graph was created with only the given chain and then populated with compatible transfers from the GIGABASE. The graph becomes very dense, allowing several permutations of debris objects and transfer times. Once the directed graph is created, we can traverse it with the above criteria.

Manual Completion of Campaigns

Two dedicated tools were implemented to insert missing bodies automatically using a ΔV database (normally the GIGABASE or the single-impulse database), or AF2. The tool looks for long sitting times in a chain

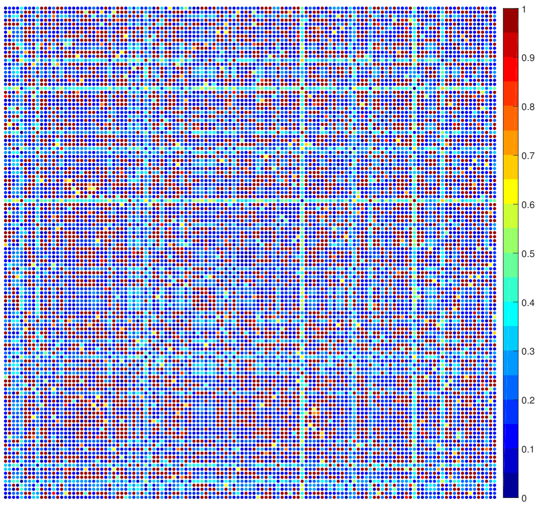


FIGURE 1. *Minimum body-to-body transfer ΔV.*

(> 7 d) and time gaps between missions (> 36 d) and computes the ΔV-optimal insertion point for each debris.

Anchor Bodies

Late in the competition the idea of anchor bodies was developed. These bodies are ones that will likely not appear in the same chain with each other due to having generally unfavourable relative geometry. For example, bodies 74, 102, 109 have not only very high ΔV to transfer between any pair of them, but also comparatively high ΔV to transfer to all of the other bodies. This is clearly visible in Fig. 1 which shows the minimum ΔV, over the entire 8.1-yr launch period, needed to transfer from a body to any other body in 25 days or less, computed by scanning the GIGABASE. Each column corresponds to a different departure body, IDs 0 through 122 left-to-right, while each row corresponds to an arrival body, IDs 0 through 122 bottom-to-top. The plot is nearly symmetric. The three bright stripes in each direction correspond to bodies 74, 102, 109, whose inclinations and node rates of 100.98°,101.07°, 96.24°, and 1.28°/day, 1.30°day, 0.75°/day, make them outliers by at least 0.4° and 0.06°/day (body 74), 0.01°/day (body 109).

Converse to the idea of anchor bodies, but relevant in that chains would have to be built around them, are the following two observations, facilitated by the GIGABASE. First, the minimum ΔV over all possible transfers from a departure body over the whole launch pe-

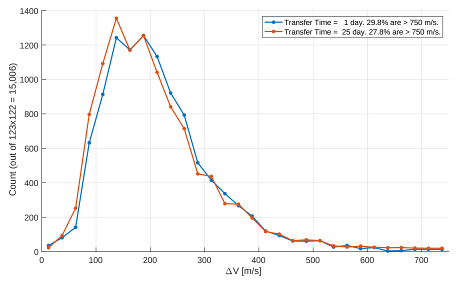


FIGURE 2. *Histogram of minimum transfer ΔVs.*

riod is less than 90 m/s for all departure bodies, and about 40 m/s when averaged over all departure bodies. Second, the "mode" of the distribution of the minimum body-to-body ΔVs shown in the matrix plot of Fig. 1 is about 125 – 225 m/s. This mode is seen in Fig. 2 which provides a histogram of the ΔVs, binned in 25 m/s bins.

In an attempt to build chains and complete campaigns around these anchor bodies at a suitable epoch, a tool called AMM was created based on the tools and algorithms developed in ACM and AF2. The main motivation behind AMM was to force the chain building process to maintain long, campaign-compatible chains with near equal lengths. The tool was developed late in the competition and therefore did not have sufficient time to mature to produce results that could be used. The utility of the anchor bodies would thus have been to reduce the dimension of the search space, allowing a fuller search to be conducted in the same time.

Rare bodies

A variation on the Anchor Body concept was also considered late in the competition but not fully explored due to time constraints. Using the quadratic-fit ΔV estimates, the minimum ΔV for a given sequence of debris objects is readily found as a quadratic programming problem (almost always convex with inequality constraints on time between manoeuvres). An attempt at building a campaign (i.e., a complete set of missions) begins by tallying how many times each debris occurs in the entire pool of debris sequences. We pull out the subset of missions that include the "rarest" debris and permute that set with the set of missions containing the object with the second fewest occurrences. This way the most difficult objects are built into the mission sets early in the design process. After each permutation, the set is filtered by cost per unique object and number of "difficult" objects deorbited.

6 Campaign Re-Adjustment

Genetic Algorithm

Early in the contest, campaigns were assembled by piecing together locally-optimised missions spanning different bodies and different epochs. About halfway through, a need for campaign-level, global optimisation was identified as critical to improving our score. One approach, and the one which was ultimately used for the remainder of the contest, was to pose the problem as a single-traveling-salesman formulation of the campaign. A customised genetic algorithm (GA), named GIGA, was written in Matlab, based on the generic GA of Kirk [5]. The problem was represented as a list of nodes, visited sequentially and separated by a manoeuvre time. The key breakthrough in formulating the problem was the inclusion of both the debris objects and launch offsets (time from end of prior mission to start of next mission) as nodes, as opposed to keeping separate lists of debris and absolute epochs for each launch. This problem is similar to the Time-Dependent Travelling Salesman Problem (edge costs depend on order), but with an added dimension of an associated choice variable (manoeuvre time) at each edge.

The problem was encoded into a genome which took discrete values. Each genome was an ordered list of debris and launch offsets, each of which had an associated time code which mapped into a time-between-nodes that considered different values for debris and launch offsets (Fig. 3). In constructing the problem this way, only complete and time-feasible campaigns were modelled and produced. This eliminated the need to have any constraints enforced numerically (the 5000 kg propellant limit manifested itself strongly enough in the cost function that an explicit constraint was not required), other than the total campaign time constraint which was enforced with a barrier function of a 10% cost increase per day above the maximum time.

The fitness of each genome was a near-instantaneous, full evaluation of the campaign cost function using ΔVs from the GIGABASE. Being able to directly and rapidly optimise the campaign cost function was essential to this approach. The selection of which genomes advanced to the next generation was accomplished using a deterministic tournament of size 4 and tournament players were selected randomly. Other sizes were explored, but 4 was found to be a good balance between

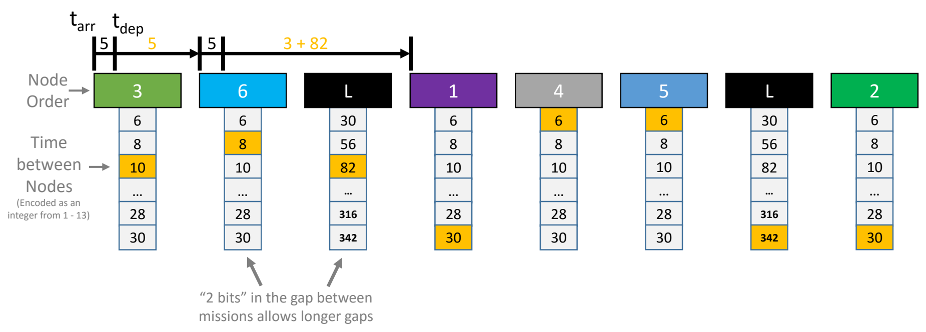

t_{arr}
t_{dep}
5
5
5
3 + 82
Node Order
Time between Nodes
(Encoded as an integer from 1 - 13)
3
6
L
1
4
5
L
2
"2 bits" in the gap between missions allows longer gaps

FIGURE 3. *Encoding of the Campaign for the Genetic Algorithm*

flexibility and selectiveness. The winner of the deterministic tournament advanced without modification to the next generation and was selected for 3 replications.

Because the node list had to be unique (non-repeating), a simple crossover was not implemented due to the possibility of generating infeasible genomes. Thus mutation is the only mechanism for improving a genome. Each mutation is constructed by first selecting two genes, say I and K, from each genome. A few factors were used for selecting the genes: Random; propellant mass fraction of that node (average of transfers to and from); propellant mass fraction compared to full mission (launch fraction of the whole mission); distance between I and K. By the end of the contest, there were 13 different mutations of roughly six types: (i) Swap, switches the list position and/or time code of node I and node K, (ii) Slide, puts node K next to node I and slides all the nodes and/or times between I and K over by 1, (iii) Flip, reverses the order of all nodes and/or times between I and K, (iv) Time Mutate, randomises the time code of nodes I and K, (v) Net-Zero Time Mutate, randomises the time code of node I, then changes K by the inverse amount, and (vi) Insertion (Net-zero Time Slide), puts node K between node I and $I+1$ and adjusts the times of I, $I+1$, K, and $K-1$ so as to minimize the perturbations of the epochs of the rest of the campaign. The net-zero operators were invented because, although a certain slide or mutation would help locally, it would perturb the epochs of many other bodies, ultimately causing the overall cost to increase. Near the end of the contest, 15 heuristic combinations of weighting values for gene selection and mutation types were implemented, and one was selected randomly in each run. For instance, one combination focused on inserting high-mass-fraction nodes into low-mass-fraction missions while others focused on optimising the time only (without reordering nodes).

Random initialisation of GIGA did yield feasible campaigns, but they were not cost-competitive with human-generated campaigns. However, when seeded with a human-generated campaign, GIGA was able to significantly improve that campaign's cost. Therefore, initialisation was accomplished by encoding a previous campaign, then mutating it 10–30 times to introduce sufficient randomness in the initial population without totally destroying the good seed campaign. One problem with this implementation was its propensity to lose diversity. To partially correct this, if no improved campaigns were found within a certain number of iterations, an additional 10 mutation steps were introduced without selection.

A single-threaded GA could run through tens of thousands of iterations with a population of a few hundred genomes in under an hour on a typical PC. In the last days of the contest, GIGA was parallelized and running on 12 nodes of a cluster (total of 144 threads). Each call of GIGA would search stored solutions for the globally best solution for initialisation and attempt to improve it using a random heuristic setting. By the end of the contest, approximately 6,400 full GIGA runs had been completed, based on the initial inputs of about 30 seeds, which means approximately 10^{11} campaigns were evaluated, or approximately 10^{13} body-to-body transfers.

Human-Guided Adjustments

If or when GIGA gets stuck in a local optimum, it can be advantageous to provide new, slightly varied initial solutions as a "kick" in hopes that subsequent runs may find new, lower-cost solutions. To construct new initial guesses, the cost-contour plot (Fig. 6) was used extensively to identify ΔV-infeasible missions and the worst-performing missions in terms of mission cost. Debris involved in high-ΔV transfers were typ-

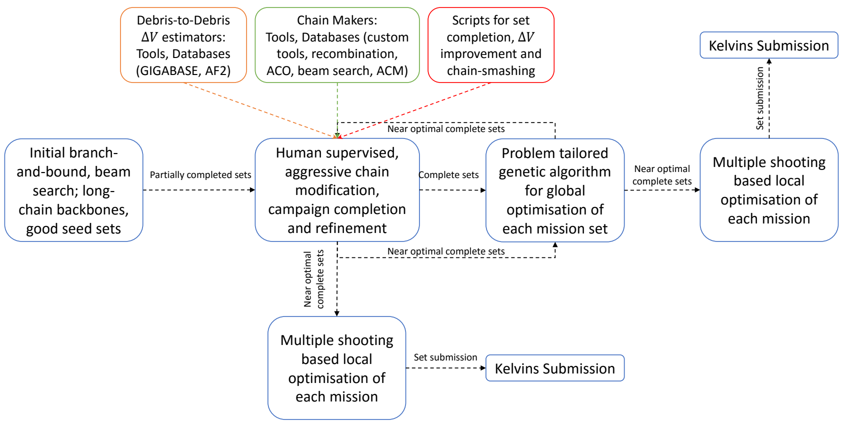

Debris-to-Debris ΔV estimators: Tools, Databases (GIGABASE, AF2)
Chain Makers: Tools, Databases (custom tools, recombination, ACO, beam search, ACM)
Scripts for set completion, ΔV improvement and chain-smashing
Kelvins Submission
Set submission
Near optimal complete sets
Initial branch-and-bound, beam search; long-chain backbones, good seed sets
Partially completed sets
Human supervised, aggressive chain modification, campaign completion and refinement
Complete sets
Problem tailored genetic algorithm for global optimisation of each mission set
Near optimal complete sets
Multiple shooting based local optimisation of each mission
Near optimal complete sets
Near optimal complete sets
Multiple shooting based local optimisation of each mission
Set submission
Kelvins Submission

FIGURE 4. *Workflow process.*

ically removed from their respective chains and added to shorter, low-scoring missions, either by prepending, appending, or inserting. The GA could often improve considerably on the new initial guesses even if one or two of the new transfers had high ΔV.

These same techniques were also leveraged to reduce the number of missions in a given campaign. For example, the initial 10-mission and 9-mission solutions provided to the GA were created by starting with 11- and 10-mission campaigns, respectively, disbanding the shortest mission entirely, and distributing those bodies across the remaining missions.

7 Final Optimisation

The local optimiser used for individual missions was based on the OPTIFOR framework [6]. The complete trajectory is decomposed into different legs, which facilitates the modeling of rendezvous constraints and makes the process less sensitive to the control variables (multiple shooting formulation). A forward-backward strategy is implemented to reduce sensitivity with respect to the initial guess. Additionally, each leg is discretized into multiple segments, where a segment corresponds to an impulsive ΔV followed by a propagation of the full J_2-spacecraft dynamics. If no manoeuvre is needed at the beginning of a segment, the optimiser drives the corresponding ΔV to zero. The optimisation of the number of impulses as well as their respective locations is therefore automatically resolved. A total of 10 manoeuvres per revolution were considered to allow sufficient variety in the manoeuvre locations. Initial mass was minimized, while the final mass was constrained to be equal to the dry mass. Initial guesses were ballistic during the first part of the competition, then initial guesses from the decoderRing (see Databases) were used when available. The resulting discrete problem is solved using SNOPT [7]. Smooth convergence after 2,000 iterations was observed for most missions (the number of iterations can probably be decreased by changing some step-size parameters and better scaling of variables and constraints). The HDDP solver [8] was also tested on long debris-to-debris transfers, but it was found to be generally slower to converge.

8 Putting it all together

As described in this paper, the team had a variety of methods for estimating body-body ΔV and creating databases of body-body transfers and chains. Thus the human-in-the-loop was an essential part of the workflow and eventual finding of the winning solution (see Fig. 4). The most important contributions to attaining competitive solutions was the use of the branch-and-bound search, the associated beam search and campaign-completion strategies (manual and ACO) to feed GIGA, which in turn fed promising solutions to the final optimisation step. After the creation of GIGA was completed in the final week of the competition, GIGA became the primary mechanism for global optimisation at the campaign level. GIGA was very good at making large numbers of significant changes to existing solutions, but had difficulty in finding truly new global optima due to the extremely strong local optima

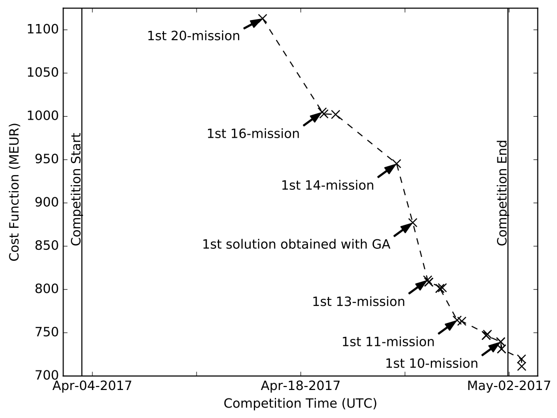


FIGURE 5. *Evolution of score during competition.*

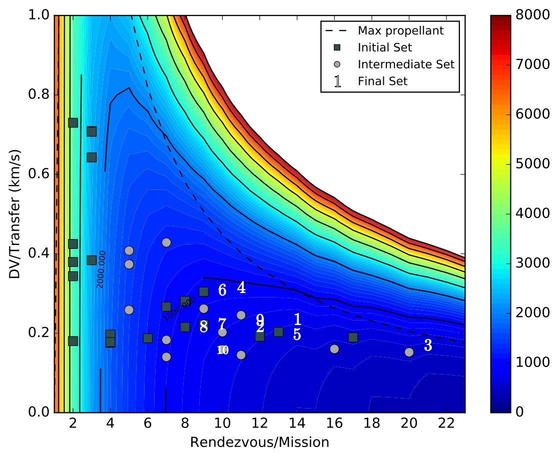


FIGURE 6. *Cost-contour plot with final solution, numbered by mission; base cost: 55 MEUR.*

in this problem, and sometimes missed "obvious" single changes. So, human analysts focused on: creating qualitatively different input seeds for GIGA (including reducing the number of launches), and could even include launches which grossly violated the propellant mass limit (by 10,000+ kg); manually modifying GIGA outputs to make an obvious swap or insertion, or to inject some desirable features, such as trying to even out the number of debris in each launch or "smash" smaller chains together. Results coming out of GIGA were either directly optimised and submitted to Kelvins, or were further refined by human adjustment and then optmised and submitted, as showing in Fig. 4.

9 Results

Figure 5 provides a summary of all the complete missions sets submitted by JPL during the competition, including the team's winning final submission of 731 MEUR. In general, cost decreases as the number of missions decreases, and flattens near 10 missions. Also depicted on this graph are 10-mission solutions found by JPL, completed shortly after the competition ended, costing 720 MEUR and 711 MEUR. Although a feasible 9-mission campaign was found, the best cost, at 750 MEUR, was worse than the best 10-mission campaigns.

Figure 6 shows cost contours for various ΔVs per transfer and rendezvous per mission. Overlayed on the contour plot is our initial 20-mission solution, the solution we submitted on April 27 when JPL occupied the top of the leaderboard for the first time (labeled 'Intermediate Set'), and our final submitted solution (numbered in white by mission number). As our solutions improved they moved from the top-left to the bottom-right side of the contour plot. For the final submitted solution, the smallest chain has nine rendezvous and the largest has 21.

Complete summaries of the final submitted solution and the 711-MEUR solution, obtained shortly after the competition, are provided in Tables 1- 4. Every mission ends with $m_{dry} = 2000$ kg. With respect to the submitted solution, the 711-MEUR solution shares missions 1 and 2; reshuffles some debris across missions 4, 5, and 9 (mission 4 grows in number of bodies by one, and mission 5 shrinks by one); and makes minor timing and mass adjustments to the remaining missions. The base cost for the revised missions is assumed to be the maximal base cost of 55 MEUR, while for the unchanged missions the base cost is kept at the value obtained in the submitted solution.

10 Conclusions

The debris rendezvous problem posed for this edition of the GTOC series was a challenging problem of interdependent and time-dependent combinatorics. The insights into the problem dynamics, which readily yielded a plethora of low-ΔV chains of transfers between debris objects, coupled with a tuned beam search to build near-complete multi-spacecraft campaigns, fed into grid-search-based and ant-colony-optimisation-based design phases to complete the campaigns. Complete campaigns then benefited greatly from a judiciously genomed genetic algorithm, as indicated by the drop in cost annotated in Fig. 5. However

TABLE 1. *Campaign Overview of Final Submitted Solution, UTC submission times on 01 May 2017 indicated.*

Mission	Start MJD2000	End MJD2000	Launch Mass, kg	Number of objects	Debris ID	UTC
1	23557.18	23821.03	5665.38	14	23,55,79,113,25,20,27,117,121,50,95,102,38,97	20:17
2	23851.08	24024.53	4666.15	12	19,115,41,26,45,82,47,85,7,2,11,77	20:17
3	24057.47	24561.49	6589.58	21	72,107,61,10,28,3,64,66,31,90,73,87,57,35,69,65,8,43,71,4,29	21:42
4	24637.26	24916.44	5679.10	11	108,24,104,119,22,75,63,112,37,32,114	20:18
5	24946.47	25232.94	4906.59	14	84,59,98,1,40,51,36,67,62,99,54,122,76,15	20:18
6	25262.95	25455.15	5062.74	10	101,48,53,5,12,39,58,13,60,74	20:18
7	25485.20	25682.33	4082.33	10	49,9,70,93,105,46,88,118,18,91	20:18
8	25712.38	25915.53	3725.73	9	86,34,100,30,92,6,110,96,81	20:19
9	25946.06	26237.29	4897.35	12	33,68,116,106,14,52,120,80,16,94,83,89	20:19
10	26267.80	26416.00	3438.62	10	44,111,56,78,0,17,109,103,42,21	20:19

TABLE 2. *Campaign Overview of 711-MEUR Solution, Obtained Shortly after the Competition.*

Mission	Start MJD2000	End MJD2000	Launch Mass, kg	Number of objects	Debris ID
1	23557.18	23821.03	5665.38	14	23,55,79,113,25,20,27,117,121,50,95,102,38,97
2	23851.08	24024.53	4666.15	12	19,115,41,26,45,82,47,85,7,2,11,77
3	24057.47	24561.49	6483.75	21	72,107,61,10,28,3,64,66,31,90,73,87,57,35,69,65,8,43,71,4,29
4	24635.53	24968.66	4482.13	12	119,24,108,22,83,75,63,112,37,104,32,114
5	24998.68	25235.88	4478.11	13	84,67,51,40,36,1,62,99,54,122,98,76,15
6	25266.35	25455.14	5122.95	10	101,48,53,5,12,39,58,13,60,74
7	25485.20	25681.08	4090.80	10	49,9,70,93,105,46,88,118,18,91
8	25711.12	25916.47	3728.67	9	86,34,100,30,92,6,110,96,81
9	25947.96	26239.81	4888.26	12	33,68,116,106,14,52,120,59,16,80,94,89
10	26269.94	26413.16	3497.42	10	44,111,56,78,0,17,109,103,42,21

it must be stressed that the genetic algorithm required very good initial-seed sets to manipulate, as well as iterations with human-guided searches, synergies which were key to yielding the winning solution, and the reason behind the centrality in Fig. 4 of the "Human supervised, aggressive chain modification" box.

The JPL team thanks the Advanced Concepts Team of the European Space Agency, in particular the team lead Dario Izzo, for posing this fascinating and relevant problem, for making the logistics of problem dissemination and solution verification almost trivial, and for introducing the excitement of real-time solution ranking.

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] D. Izzo. *Problem description for the 9th Global Trajectory Optimisation Competition*, doi: 10.5281/zenodo.570193, May 2017.

[2] P. Gurfil. Analysis of J2-perturbed motion using mean non-osculating orbital elements. *Celest. Mech. Dyn. Astron.*, 90(3–4):289–306, 2004.

[3] J. R. Stuart, K. C. Howell, and R. S Wilson. Application of multi-agent coordination methods to the design of space debris mitigation tours. *Advances in Space Research*, 57(8):1680–1697, 2016.

[4] Luís F Simões, Dario Izzo, Evert Haasdijk, and AE Eiben. Multi-rendezvous Spacecraft Trajectory Optimization with Beam P-ACO. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 141–156. Springer, 2017.

[5] J. Kirk. *Travelling Salesman Problem GA, Matlab code*, https://goo.gl/R99UHm, April 2017.

[6] G. Lantoine. *A Methodology for Robust Optimization of Low-Thrust Trajectories in Multi-Body Environments*. PhD thesis, Georgia Institute of Technology, 2010.

[7] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[8] G. Lantoine and R. P. Russell. A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems, Part 1: Theory. *Journal of Optimization Theory and Applications*, 154(2):382–417, 2012.

TABLE 3. *Mission Characteristics of Final Submitted Solution.*

Mission	*Rendezvous Duration, days*
1	5.00,5.00,5.04,5.01,5.01,5.03,5.00,5.00,5.00,5.03,5.03,5.04,5.04,5.00
2	5.00,5.02,5.02,5.00,5.04,5.00,5.05,5.02,5.07,5.03,5.02,5.00
3	5.00,5.06,5.01,5.02,5.07,5.02,5.04,5.02,5.01,5.02,5.01,5.07,5.06,5.02,5.01,5.01,5.06,5.01,5.02,5.04,5.00
4	5.00,6.01,6.01,6.03,6.05,6.05,6.04,6.01,6.06,6.04,5.00
5	5.00,5.02,5.07,5.04,5.01,5.01,5.02,5.06,5.06,5.02,5.06,5.01,5.07,5.00
6	5.00,5.02,5.01,5.04,5.07,5.02,5.01,5.02,5.02,5.00
7	5.00,5.00,5.06,5.06,5.04,5.06,5.04,5.06,5.03,5.00
8	5.00,5.01,5.03,5.00,5.01,5.04,5.07,5.02,5.00
9	5.00,5.51,5.53,5.53,5.53,5.55,5.54,5.53,5.54,5.55,5.52,5.00
10	5.00,5.54,5.50,5.50,5.52,5.52,5.54,5.53,5.52,5.00
Mission	*Transfer Duration, days*
1	24.86,24.98,22.42,24.99,0.29,10.63,25.00,2.70,1.51,1.41,24.67,24.31,5.86
2	24.93,0.28,0.73,0.39,17.07,1.61,22.42,2.39,15.88,24.97,2.49
3	14.16,24.94,2.87,8.10,9.00,23.13,23.09,23.09,22.83,24.98,24.98,24.93,24.94,9.10,13.44,24.99,24.94,24.99,24.98,24.96
4	23.96,6.48,16.72,23.97,23.95,23.95,23.96,23.99,23.94,23.96
5	0.45,3.17,24.93,10.34,12.53,7.11,13.44,24.94,24.94,24.98,22.19,24.99,22.01
6	24.91,0.30,18.39,3.08,20.24,24.96,24.85,24.97,0.28
7	15.69,0.50,9.83,24.94,24.90,24.48,20.87,24.91,0.66
8	10.03,24.00,2.83,24.99,24.99,24.96,21.19,24.98
9	22.69,4.24,24.47,24.46,24.47,24.44,24.46,24.46,24.46,18.54,9.22
10	0.81,11.59,7.66,1.11,17.46,6.47,20.47,24.47,3.99
Mission	ΔV, *m/s*
1	161.8,139.2,65.8,208.2,115.2,300.1,564.9,78.3,105.0,233.3,453.5,340.4,300.8
2	659.0,301.1,252.1,143.8,146.8,68.6,40.6,84.2,105.3,448.5,148.0
3	219.1,80.8,105.2,55.2,140.2,85.5,95.0,237.6,205.9,149.9,245.2,71.6,197.3,160.4,132.2,240.0,161.2,364.3,230.4,232.5
4	86.1,103.1,62.6,222.9,709.1,553.9,219.9,233.9,739.0,232.6
5	129.6,45.2,172.9,52.6,160.7,280.8,221.1,163.5,98.2,115.7,164.8,674.8,291.1
6	156.0,198.0,305.8,71.2,194.4,920.5,314.1,353.0,272.8
7	400.6,173.6,211.3,374.4,109.6,171.2,145.1,194.3,233.0
8	287.9,111.9,112.2,144.5,540.0,260.1,198.8,82.7
9	83.3,148.1,495.9,464.9,405.2,285.9,254.8,62.3,156.6,36.5,174.9
10	189.4,112.9,110.0,121.3,117.9,280.1,300.4,120.6,70.2

TABLE 4. *Mission Characteristics of 711-MEUR Solution, Obtained Shortly after the Competition.*

Mission	*Rendezvous Duration, days*
1	5.00,5.00,5.04,5.01,5.01,5.03,5.00,5.00,5.00,5.03,5.03,5.04,5.04,5.00
2	5.00,5.02,5.02,5.00,5.04,5.00,5.05,5.02,5.07,5.03,5.02,5.00
3	5.00,5.06,5.01,5.02,5.07,5.01,5.04,5.01,5.00,5.01,5.01,5.07,5.06,5.02,5.02,5.01,5.06,5.01,5.03,5.04,5.00
4	5.00,5.03,5.03,5.06,5.00,5.01,5.02,5.01,5.06,5.04,5.00,5.00
5	5.00,5.07,5.07,5.05,5.01,5.03,5.03,5.04,5.03,5.04,5.01,5.04,5.00
6	5.00,5.03,5.03,5.02,5.03,5.04,5.02,5.03,5.02,5.00
7	5.00,5.03,5.04,5.04,5.07,5.07,5.06,5.04,5.02,5.00
8	5.00,5.02,5.03,5.01,5.06,5.05,5.04,5.04,5.00
9	5.00,5.00,5.02,5.07,5.04,5.02,5.06,5.02,5.04,5.05,5.04,5.00
10	5.00,5.01,5.00,5.05,5.02,5.06,5.05,5.04,5.03,5.00
Mission	*Transfer Duration, days*
1	24.86,24.98,22.42,24.99,0.29,10.63,25.00,2.70,1.51,1.41,24.67,24.31,5.86
2	24.93,0.28,0.73,0.39,17.07,1.61,22.42,2.39,15.88,24.97,2.49
3	14.16,24.94,2.86,8.10,9.01,23.13,23.10,23.10,22.84,24.99,24.97,24.93,24.94,9.09,13.44,24.99,24.94,24.99,24.97,24.96
4	24.94,24.97,24.96,24.94,25.00,24.99,24.98,24.99,24.94,24.96,23.21
5	0.48,5.15,10.86,14.74,3.25,24.96,20.93,24.89,24.96,24.96,0.33,16.26
6	21.64,0.27,18.90,3.32,19.25,24.96,24.98,24.97,0.28
7	15.80,0.27,10.30,24.96,24.93,24.35,19.67,24.96,0.28
8	11.17,24.60,2.09,24.98,24.94,24.95,22.41,24.95
9	24.44,0.29,24.98,24.93,24.96,24.98,24.93,22.55,21.03,13.50,24.92
10	1.34,11.52,0.35,5.26,16.55,9.20,19.23,24.95,4.58
Mission	ΔV, *m/s*
1	161.8,139.2,65.8,208.2,115.2,300.1,564.9,78.3,105.0,233.3,453.5,340.4,300.8
2	659.0,301.1,252.1,143.8,146.8,68.6,40.6,84.2,105.3,448.5,148.0
3	19.4,69.2,104.4,54.8,137.3,84.2,91.6,235.4,203.0,146.6,237.4,70.8,194.6,159.1,131.4,235.6,162.4,363.0,227.3,224.6
4	108.8,60.4,113.1,169.7,240.9,605.6,320.8,102.1,295.2,120.9,149.3
5	245.4,100.2,160.7,121.3,110.5,182.4,87.7,108.5,159.2,434.4,162.4,393.2
6	174.1,211.6,302.7,76.1,211.1,908.3,314.4,352.5,275.0
7	398.1,171.9,208.1,373.6,110.8,161.6,146.7,208.7,241.9
8	290.0,117.5,114.2,141.2,558.2,261.8,190.7,66.2
9	108.8,151.2,491.8,438.5,397.0,297.6,207.3,105.0,64.3,160.8,140.2
10	203.1,108.8,126.4,79.4,125.9,260.8,300.0,169.4,111.4

Acta Futura 11 (2018) 37-47

DOI: 10.5281/zenodo.1139226

Acta Futura

GTOC9: Results from the National University of Defense Technology (team NUDT)

Ya-Zhong Luo[*], Yue-He Zhu, Hai Zhu, Zhen Yang, Zhen-Jiang Sun, Jin Zhang

College of Aerospace Science and Engineering, National University of Defense Technology
Changsha 410073, China

Abstract. The ninth edition of the Global Trajectory Optimization Competition (GTOC) series was successfully organized in April 2017, wherein the competitors were called to design a series of missions able to remove a set of 123 orbiting debris pieces while minimizing the overall cumulative cost. A three-level optimization framework of the NUDT Team is presented and an improved Ant colony Optimization Algorithm, a hybrid-encoding Genetic Algorithm and an improved Differential Evolution algorithm are applied to solve the complex problem, which combines the dynamic TSP, mixed-integer sequence optimization and perturbed trajectory rendezvous optimization. The result obtained during the competition ranked second in the eventual leaderboard.

1 Introduction

The design of space trajectories can be profitably approached as a global optimization problem. The optimal trajectory, which is significant for practical space mission design, is usually very difficult to be obtained. The Global Trajectory Optimization Competition (GTOC) series [1], was born with the objective of fostering research in this area by letting the best aerospace engineers and mathematicians worldwide challenge themselves to solve one, difficult, well-defined, problem of spacecraft trajectory design.

Since the launch of the first satellite, Sputnik, in 1957, mankind has placed countless spacecraft in orbit around the Earth. Today, less than 10% of the trackable objects orbiting the Earth are operational satellites. The remainder is simply junk and the space debris is becoming an increasingly serious problem. Following the unprecedented explosion of a Sun-synchronous satellite, the Kessler effect triggered further impacts and the Sun-synchronous orbits environment was severely compromised [2]. Scientists from all main space agencies and private space companies isolated a set of 123 orbiting debris pieces that, if removed, would restore the possibility to operate in that precious orbital environment and prevent the Kessler effect to permanently compromise it.

For calling to protect the environment of earth orbits, the background of GTOC9 is to clean the debris to avoid the Kessler effect. It is the first time that GTOC focuses on the near-earth space problem. The competitors are called to design a series of missions to remove a set of 123 orbiting debris pieces while minimizing the overall cumulative cost.

To find the optimal solution of such a complex problem, three sub-problems need to be extracted and

[*]Corresponding author. E-mail: luoyz@nudt.edu.cn

solved. First, the set of 123 debris pieces needs to be divided into several groups. Each group of debris is removed by one mission. Optimization is performed to minimize the overall cumulative cost. This can be approached as a dynamic TSP and solved by evolutionary algorithms [3]. Second, given a group of the debris, one mission is designed by mixed-integer optimization to remove them while costing minimal velocity increment [4]. Finally, given the current and next debris as well as the rendezvous duration, the impulsive maneuver strategy is designed to produce the optimal flight trajectory [5].

This paper presents the solving methods and results from the National University of Defense Technology (NUDT) for GTOC9. The remainder of the paper is organized as follows. Section 2 makes a short description of the problem and analyzes the main challenges of this problem. Section 3 gives the optimization framework of the NUDT Team. The detailed solving approach and procedure are presented in Section 4-6. Conclusions are drawn in Section 7.

2 Problem Description and Analysis

2.1 Problem Description

The problem of GTOC9 is to design n missions to cumulatively remove all the 123 orbiting debris while minimizing the overall cumulative cost of such an endeavor. The cost function is expressed as

$$\begin{array}{l} J = \sum\limits_{i=1}^{n} C_i = \sum\limits_{i=1}^{n} \left[c_i + \alpha (m_{0_i} - m_{dry})^2 \right] \\ c_i = c_m + \frac{t_{submission} - t_{start}}{t_{end} - t_{start}} (c_M - c_m) \end{array} \tag{1}$$

where C_i is the cost charged by the contracted launcher supplier for the i^{th} mission. At the beginning of the i^{th} mission, m_{0_i} is the spacecraft mass and m_{dry} its dry mass. Each spacecraft initial mass m_0 is the sum of its dry mass, the weights of the $N \geq 1$ de-orbit packages to be used and the propellant mass: $m_0 = m_{dry} + Nm_{de} + m_p$. α is a parameter set to be $\alpha = 2.0 \times 10^{-6}$ $\left[\text{MEUR/Kg}^2\right]$. $t_{submission}$ is the epoch at which the i^{th} mission is validated, and t_{end} and t_{start} are the end and the beginning epochs of the GTOC9 competition. The minimal basic cost c_m is 45 MEUR and the maximum cost c_M is 55 MEUR. Other definition and constraints can be found in [2].

During each transfer between two successive debris, the spacecraft dynamics is described by a Keplerian motion perturbed by main effects of an oblate Earth, i.e. J_2.

$$\begin{cases} \dot{x} = v_x, \quad \dot{y} = v_y, \quad \dot{z} = v_z \\ \dot{v}_x = -\frac{\mu x}{r^3} + \frac{3\mu J_2 R_E^2}{2r^5} \left(\frac{5z^2}{r^2} - 1 \right) x + \Gamma_x \\ \dot{v}_y = -\frac{\mu y}{r^3} + \frac{3\mu J_2 R_E^2}{2r^5} \left(\frac{5z^2}{r^2} - 1 \right) y + \Gamma_y \\ \dot{v}_z = -\frac{\mu z}{r^3} + \frac{3\mu J_2 R_E^2}{2r^5} \left(\frac{5z^2}{r^2} - 3 \right) z + \Gamma_z \end{cases} \tag{2}$$

where $\boldsymbol{r} = [x, y, z]^T$ and $\boldsymbol{v} = [v_x, v_y, v_z]^T$ are the spacecraft's position and velocity vector described in the mean equator inertial coordinate system of the center body, $r = ||\boldsymbol{r}||$, $|| \cdot ||$ denotes the Euclidean norm of a vector, μ, R_E and J_2 are the gravitational constant, mean equator radius and J_2-perturbation coefficient of the central body respectively. $\boldsymbol{\Gamma}$ is the thrust acceleration.

The only maneuvers allowed to control the spacecraft trajectory are instantaneous changes of the spacecraft velocity (its magnitude being denoted by ΔV. After each such maneuver, the spacecraft mass is to be updated using Tsiolkovsky equation:

$$m_f = m_i \exp\left(-\frac{\Delta V}{v_e} \right) \tag{3}$$

where $v_e = I_{sp} g_0$. A maximum of 5 impulsive velocity changes is allowed during each transfer between two successive debris. These do not include the departure and arrival impulse.

2.2 Analysis

The goal of this problem is to design a minimal mass vehicle compliant of a series of suc-cessive removal missions. For optimization this problem, three following sub-problems must be addressed:

1) How to plan the successive removal missions?

2) How to minimize the cost of a single mission?

3) How to minimize the trajectory between each two debris?

The first problem is a large-scale multi-sequence combinatorial optimization problem, which is similar to the combination of the classic TSP (Travelling Salesman Problem) and BPP (Bin Packing Problem). The TSP is to find a minimal distance closed path visiting all the nodes once and the BPP is to find a minimal bin-packing scheme placing all the items without omission.

However, compared with TSP and BPP, the following differences make this sub-problems much more difficult to solve.

In the BPP, only the weight constraint need to be satisfied. The placing sequence of each item can be ignore. While in this problem not only the total fuel cost constraint should be satisfied, but also the sequence of the visited debris must be considered. This makes the solution space of this problem much larger than BPP and increase the optimization difficulty.

In the TSP, each node has to be visited once and once only and the path is closed. While removing the debris can be divided into several missions in this problem, where the number of missions are not fixed and the removal paths are opened. This makes the solution space of this problem much larger than TSP and also increase the optimization difficulty.

In the TSP, all the nodes to be visited are fixed in the plane and the cost of going from one node to another can be easily calculated according to the Cartesian distance in the plane. While in this problem the cost of going from a debris to the next one depends on the starting date and arrival date. This makes the problem time-dependent and further increase the optimization difficulty.

The second problem is a mixed-integer nonlinear-programming (MINLP) problem. Not only the sequence of the debris (integer variables) but also the transfer times between each debris (real variables) need to be considered as the Design Variables, which are typically much more difficult to solve than both mixed-integer linear-programming (MILP) and nonlinear-programming (NLP) problems.

The third problem is an orbital transfer problem. It is very difficult to find the optimal solution for the long-duration (t_f is up to 30 days) perturbed rendezvous problem. A fuel-optimal orbital rendezvous problem is to find a maneuvering plan for the spacecraft to minimize the total velocity increment and simultaneously satisfy specific constraints. While the J_2-perturbation is taken in account, the well-known orbital targeting algorithms such as the Lambert algorithm will be failed in obtaining the feasible solutions, and the constrained optimization methods which can directly corporate final state constraints, such as SQP, will also encounter convergence problems for long-duration rendezvous. From the scope of orbital dynamics, at least two impulses are needed to target the final position and velocity vectors. However, the total velocity increment of the 2-impulse maneuvers will be very large for a rendezvous mission, especially for the long-duration, large non-coplanar rendezvous. Therefore, a rendezvous mission usually uses more than two impulses. Due to the long-duration, multi-impulse characteristics, the design variables (e.g. the maneuver time) will have large search space, and many sub-optimal solutions may exist, thus it is difficult to find the global optimal solution for this problem even though the state-of-art optimization algorithm is used. In addition, numerical integration of the J_2-perturbed trajectory is required in the optimization process, which makes the optimization time-consuming.

3 Optimization framework

Based on the analysis of the problem and the optimization tools we have accumulated, our optimization framework is divided into three levels, which are illustrated in Fig. 1.

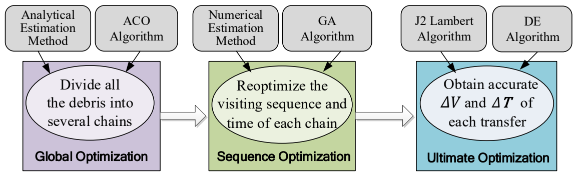

Analytical Estimation Method
ACO Algorithm
Divide all the debris into several chains
Global Optimization
Numerical Estimation Method
GA Algorithm
Reoptimize the visiting sequence and time of each chain
Sequence Optimization
J2 Lambert Algorithm
DE Algorithm
Obtain accurate ΔV and ΔT of each transfer
Ultimate Optimization

FIGURE 1. *Optimization framework*

The task of the global optimization is to appropriately divide the debris into several chains. It is a combinatorial optimization problem with huge search space that is similar to the TSP. For such NP-hard problem, no algorithm can guarantee to the global optimum. As an efficient optimization tool, ACO performs well on the classic TSP and many other TSP variants. Following the characteristic of this problem, we improve an ACO based on the one for the extravehicular missions packing programming (EMPP) [6] and apply it to solve the first-level problem. In addition, compared with the calculating of the distance between any two cities in the TSP, the calculating of the ΔV from a debris to the next one is much more time-consuming. Thus, an analytical estimation method of the transfer ΔV and ΔT between any two debris is employed in the global optimization.

With the completion of the global optimization for the whole mission, the number of the chains and the debris in each chain are determined and will not be changed. However, since the optimal transfer ΔV and ΔT between each debris are estimated by an analytical model with high error (up to 30% in some conditions), a numerical estimation method of the transfer ΔV and ΔT between any two debris is developed and the mixed integer reoptimization for both the visiting sequence of the chain and the transfer time between each debris is necessary. An im-proved hybrid-encoding Genetic Agorithm (HEGA) [7, 8] is applied to solve the second-level problem.

Once the visiting sequence of each chain is determined, the optimization of the accurate transfer ΔV and ΔT between each two debris is required. The orbital transfer from a debris to another is a multi-impulse, perturbed rendezvous problem. A feasible solution cannot be directly obtained by the orbital targeting algorithms based on two-body dynamics unless some differential corrections or simple iterations are used. In order to efficiently obtain a near-optimal solution for the given long-duration (up to 30 days) rendezvous problem, a feasible iteration optimization model is employed, in which the homotopic perturbed Lambert algorithm [9, 10] is used as the orbital targeting algorithm. An improved differential evolution (DE) algorithm [11] is applied to solve the third-level problem.

4 Global Optimization for the Whole Mission

4.1 Analytical estimation method of the transfer ΔV and ΔT

The analytical estimation method for evaluating the objective function of each transfer and the overall cumulative cost are based on the Gauss form of variational equations [12],

$$\left\{\begin{array}{l}\Delta a=\frac{2}{n\sqrt{1-e^2}}\left[e\sin f\cdot\Delta v_r+(1+e\cos f)\,\Delta v_t\right]\\ \Delta e=\frac{\sqrt{1-e^2}}{na}\left[\sin f\cdot\Delta v_r+(\cos f+\cos E)\,\Delta v_t\right]\\ \Delta i=\frac{r\cos u}{na^2\sqrt{1-e^2}}\Delta v_h\\ \Delta\Omega=\frac{r\sin u}{na^2\sqrt{1-e^2}\sin i}\Delta v_h\\ \Delta\omega=\frac{\sqrt{1-e^2}}{nae}\left[-\cos f\cdot\Delta v_r+\left(1+\frac{r}{p}\right)\sin f\cdot\Delta v_t\right]-\cos i\cdot\Delta\Omega\\ \Delta M=n-\frac{1-e^2}{nae}\left[\left(2e\frac{r}{p}-\cos f\right)\Delta v_r+\left(1+\frac{r}{p}\right)\sin f\cdot\Delta v_t\right]\end{array}\right. \tag{4}$$

where the mean motion $n=\sqrt{\frac{\mu}{a^3}}$ and the semilatus rectum $p=a\left(1-e^2\right)$. Detailed procedure is described as follows.

1) Adjustment of the RAAN difference

The RAAN of an orbital object drifts due to the J_2 perturbation. The drift velocity is formulated as follow,

$$\dot{\Omega}=-\frac{3}{2}J_2\left(\frac{r_{eq}}{p}\right)^2 n\cos i \tag{5}$$

where r_{eq} is the mean radius of the earth.

As the adjustment of orbital plane costs a large velocity increment, the difference of the RAAN drift velocity between the spacecraft and the debris should be fully used. If the RAAN difference cannot be remedied naturally during the maximum rendezvous duration, an impulse perpendicular to the orbital plane can be implemented at the north or south vertex of the orbit.

$$\Delta v_h=\frac{na^2\sqrt{1-e^2}\sin i}{r}\left|\Delta\Omega\right| \tag{6}$$

2) Adjustment of the inclination difference

As the J_2 perturbation does not change the orbital inclination, the inclination difference must be remedied by maneuvers. An impulse perpendicular to the orbital plane can be implemented at the ascending node or descending node.

$$\Delta v=2\frac{h}{r}\sin\frac{|\Delta i|}{2} \tag{7}$$

where $h=r^2\dot{\theta}$, θ is the argument of latitude.

3) Adjustment of the semimajor axis and eccentricity

After the spacecraft transfers to the same orbital plane with the debris, the semimajor axis and eccentricity are adjusted by two tangential impulses. For a near circular orbit, omitting the high order terms of e^2, the impulses are formulated as follows.

If $\Delta a\Delta e\geq 0$, the first tangential impulse Δv_{t1} is implemented at the perigee, where the true anomaly $f=0$, and the second tangential impulse Δv_{t2} is implemented at the apogee of $f=\pi$.

$$\begin{cases} \Delta v_{t1} = n\frac{\Delta a + a(1-e)\Delta e}{4} \\ \Delta v_{t2} = n\frac{\Delta a - a(1+e)\Delta e}{4} \end{cases} \tag{8}$$

If $\Delta a \Delta e < 0$, the first tangential impulse Δv_{t1} is implemented at the apogee, where the true anomaly $f = \pi$, and the second tangential impulse Δv_{t2} is implemented at the perigee of $f = 0$.

$$\begin{cases} \Delta v_{t1} = n\frac{\Delta a - a(1+e)\Delta e}{4} \\ \Delta v_{t2} = n\frac{\Delta a + a(1-e)\Delta e}{4} \end{cases} \tag{9}$$

4) Estimation of rendezvous duration

The rendezvous duration should mainly come from the adjustment of the RAAN difference so as to make full use of the natural RAAN drift due to J_2 perturbation. The other adjustments do not need too much time. To be conservative, the rendezvous duration is roughly estimated as the duration for RAAN adjustment plus one day.

4.2 ACO for Debris Grouping and Bunching

ACO algorithm was originally inspired by the ability of biological ants to find the shortest path between their nest and a food source [13]. The fundamental working procedure of the ACO for debris Grouping and bunching (ACO_DGB) is similar to the classic ACO, which is shown in Algorithm 1. The most important feature of an ACO is the design of the heuristic, which is eventually combined with the pheromone information to build solutions. In this part, we mainly present the heuristic and solution construction method of the DCB_ACO.

Algorithm 1 Ant System

```
step 1:  Pheromone trail initialization;
         **while** termination criteria not met
         **do**
step 2:  Solution construction;
step 3:  Pheromone update;
         **end while**
```

The procedure of bunching a debris chain is illustrated in Fig. 2. After setting the start time and select a debris as the head of the chain, the estimation of the optimal transfer ΔV and ΔT between the last debris of the current chain and all remaining ones and the selection from the candidate pool are followed and cycled to bunch the chain one by one. When the candidate pool becomes empty, which means none of the remaining debris can be added to the tail of the chain, the procedure will be stopped and a chain will be obtained.

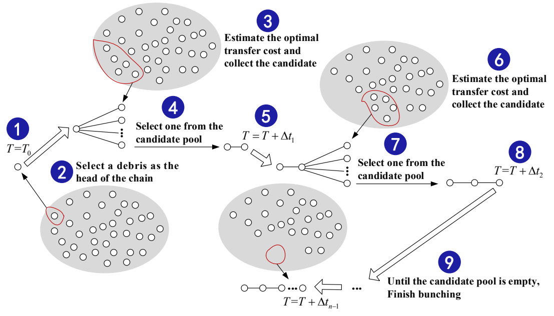

3
Estimate the optimal transfer cost and collect the candidate
6
Estimate the optimal transfer cost and collect the candidate
1
$T=T_0$
4
Select one from the candidate pool
5
$T=T+\Delta t_1$
2
Select a debris as the head of the chain
7
Select one from the candidate pool
8
$T=T+\Delta t_2$
9
Until the candidate pool is empty, Finish bunching
$T=T+\Delta t_{n-1}$

FIGURE 2. *Bunching procedure of the debris chain*

Three remarks should be noted for this procedure:

1) In steps 3 and 6, the candidate refers to all of the debris that satisfy the total fuel constraints for one mission after being added to the tail of the chain.

2) In steps 4 and 7, the probability that an ant k will choose a debris j as the next debris for the current chain b in the partial solution s is given by

$$p_{bj}^{k}(s) = \begin{cases} \frac{\tau_{bj} \cdot \eta_j{}^{\beta}}{\sum\limits_{g \in U^k(s,b)} \tau_{bg} \cdot \eta_g{}^{\beta}}, & j \in U^k(s,b) \\ 0 & otherwise \end{cases} \tag{10}$$

where $U^k(s,b)$ is the candidate pool and $\eta_j = \Delta V_{lj}$ is the heuristic value. The parameter α in the classic ACO is fixed to 1 here because using the parameter β is sufficient to reflect the weight between the pheromone information and heuristic information. τ_{bj} is the pheromone from debris l to debris j, where debris l is the last debris in the current chain b.

3) In step 5 and 8, δt is the estimated optimal transfer time between the last debris in the current chain and the selected debris.

Based on the chain bunching method, building a solution for each ant should take the following procedure, which is presented as Algorithm 2.

Algorithm 2 Solution Construction Procedure of the ACO_DGB

```
step 1:  Determine the start time $T_0$ (MJD)
step 2: Produce a debris chain based on the chain
        bunching method in Fig. 2
step 3: if None debris remains
            Go to step 4
        else
            Set the current time $T{=}T{+}\Delta T_M$ ($\Delta T_M \in rand[30day, 60day]$)
            Return to step 2
        end if
step 4: Collect all of the debris chains and obtain a solution
```

The evaporation parameter ρ is set as 0.05 and the increase of the pheromone $\Delta\tau_{ij}^{k}$ is limited to the maximum value of $0.1\tau_{ij}$ to avoid premature convergence. The pheromone update rule used in the ACO_DGB is the same as the one in the ACO for the EMPP [6].

4.3 Solving Strategy

To minimize the cost function that is expressed in Eq. (1), not only the launch times but also the total propellant cost in each launch should be reduced. The time-related part c_i is set as the maximum cost (55 MEUR) in the optimization for the whole mission.

Due to the insufficient optimization performance of the ACO_DGB, we can hardly obtain the optimal solution or even a good solution if using the algorithm to optimize 123 debris all at once and taking the solution from the result directly. In order to make the original problem easier to be optimized and obtain better solutions, a chain-by-chain solving strategy is applied, which is illustrated in Fig. 3.

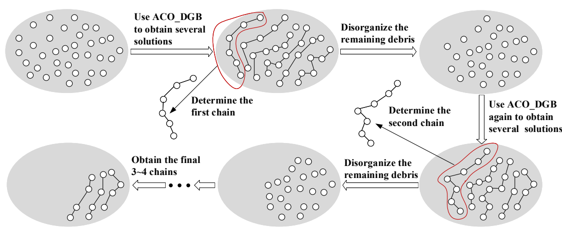

Use ACO_DGB to obtain several solutions
Disorganize the remaining debris
Determine the first chain
Determine the second chain
Use ACO_DGB again to obtain several solutions
Obtain the final 3-4 chains
Disorganize the remaining debris

FIGURE 3. *Procedure of the chain-by-chain solving strategy*

The main idea of this solving strategy is to determine the debris chains of the final solution step by step. 2000 runs will be implemented for the ACO_DGB to optimize the remaining debris each time and the first chain of the best partial solution will be selected and determined as the next chain of the final solution. Here the best partial solution refers to the one that owns the smallest objective function value excluding the determined chains. The final 3-4 chains are determined all at once because the search space is small enough and further disorganization and reoptimization for the remaining debris will not make the final solution better.

5 Sequence Optimization for the Debris Chain

5.1 Optimization Model

1) Design variables

The solution of a debris chain $\mathbf{Y}$ is made up of a group of serial integers $\mathbf{Y_1}$ and a set of real numbers $\mathbf{Y_2}$.

$$\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2) \tag{11}$$

where $\mathbf{Y}_1$ refers to the rendezvous sequence $(p_1, p_2, ...p_Q)$, and $\mathbf{Y}_2$ refers to the orbital transfer time $(dur_1, dur_2, ...dur_Q)$.

Through the sequence of its elements the serial integer vector $\mathbf{Y_1}$ represents a rendezvous order. The search space of $\mathbf{Y_1}$ is therefore discrete and its elements must be manipulated in combination.

2) Objective function

The objective is to minimize the propellant consumed by orbital maneuvers:

$$\min f_2 = (m_0 - m_{dry} - Qm_{de}) \tag{12}$$

where m_{dry} is the spacecraft's mass after the last removing mission and also denotes the spacecraft's dry mass(Generally, m_{dry} should include the propellant used by spacecraft to deorbit, otherwise, the spacecraft itself would be a debris now).

5.2 Numerical estimation method of the transfer ΔV and ΔT

The state of a spacecraft can be expressed as

$$\mathbf{E} = (a, u, \xi, \eta, i, \Omega)^T \tag{13}$$

where a is the semi-major axis, i is the orbital inclination, Ω is the right ascension of ascending node (RAAN), u is the argument of latitude, e is the eccentricity, ω is the argument of perigee, and $\xi = e\cos\omega$ and $\eta = e\sin\omega$ are the modified orbital elements suitable for de-scribing near-circular orbits.

The state variable used to express orbital differences between the spacecraft and a debris is

$$\mathbf{X} = (\Delta a/a_r, \Delta\theta, \Delta\xi, \Delta\eta, \Delta i, \Delta\Omega)^T \tag{14}$$

where the subscript r denotes the reference orbit, Δa is the difference in semi-major axis, $\Delta\theta$ is the difference in argument of latitude, Δi is the difference in orbital inclination, $\Delta\Omega$ is the difference in RAAN, and $\Delta\xi$ and $\Delta\eta$ give the differences in eccentricity vector.

Using the first order approximations, the state transitions of the orbital element differences under the J_2 perturbation are given by [10]

$$\begin{cases} \Delta a = \Delta a_0 \\ \Delta\theta = \Delta\theta_0 - \left[\frac{3}{2}n_r\frac{\Delta a_0}{a_r} + \frac{7}{2}\frac{\Delta a_0}{a_r}C(3-4\sin^2 i_r)\right]\Delta t - 4C\sin(2i_r)\Delta i_0\Delta t \\ \Delta\xi = \Delta\xi_0\cos(\dot{\omega}_{J_2}\Delta t) - \Delta\eta_0\sin(\dot{\omega}_{J_2}\Delta t) \\ \Delta\eta = \Delta\xi_0\sin(\dot{\omega}_{J_2}\Delta t) + \Delta\eta_0\cos(\dot{\omega}_{J_2}\Delta t) \\ \Delta i = \Delta i_0 \\ \Delta\Omega = \Delta\Omega_0 + \left(\frac{7}{2}\frac{\Delta a_0}{a_r}\cos i_r + \sin i_r\Delta i_0\right)C\Delta t \end{cases} \tag{15}$$

where the subscript 0 denotes the initial state, Δt is the orbital transfer time, μ is the geocentric gravitation constant, a_e is the mean equatorial radius of the Earth, $n_r = \sqrt{\frac{\mu}{a_r^3}}$ is the mean angular motion rate, $C = \frac{3J_2 {a_e}^2}{2}\sqrt{\mu}{a_r}^{-\frac{7}{2}}$, and $\dot{\omega}_{J_2} = C\left(2 - \frac{5}{2}\sin^2 i_r\right)$ is the drift rate of perigee.

Thus, the orbital transfer of the q^{th} rendezvous operation can be expressed as

$$\mathbf{X}(t_{qf}) = \mathbf{\Phi}(\Delta t_{q0})\mathbf{X}_0 + \sum_{j=1}^{2}\mathbf{\Phi}_v(\Delta t_{qj}, u_{qj})\Delta\mathbf{v}_{qj} \tag{16a}$$

$$\mathbf{\Phi}(\Delta t_{q0}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \left(\begin{matrix} -\frac{3}{2}n_r\Delta t_{q0} - \\ \frac{7}{2}C\left(\begin{matrix} 3- \\ 4\sin^2 i_r \end{matrix}\right)\Delta t_{q0} \end{matrix}\right) & 1 & 0 & 0 & -4C\sin(2i_r)\Delta t_{q0} & 0 \\ 0 & 0 & \cos\tau_{q0} & -\sin\tau_{q0} & 0 & 0 \\ 0 & 0 & \sin\tau_{q0} & \cos\tau_{q0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{7}{2}C\cos i_r\Delta t_{q0} & 0 & 0 & 0 & C\sin i_r\Delta t_{q0} & 1 \end{bmatrix} \tag{16b}$$

$$\mathbf{\Phi}_v(\Delta t_{qj}, u_{qj}) = \begin{bmatrix} 0 & 2 & 0 \\ 0 & \left[\begin{matrix} -3n_r- \\ 7C\left(\begin{matrix} 3- \\ 4\sin^2 i_r \end{matrix}\right) \end{matrix}\right]\Delta t_{qj} & -4C\sin(2i_r)\cos u_{qj}\Delta t_{qj} \\ \sin(u_{qj}+\tau_{qj}) & 2\cos(u_{qj}+\tau_{qj}) & 0 \\ -\cos(u_{qj}+\tau_{qj}) & 2\sin(u_{qj}+\tau_{qj}) & 0 \\ 0 & 0 & \cos u_{qj} \\ 0 & 7C\cos i_r\Delta t_{qj} & \left(\begin{matrix} \frac{\sin u_{qj}}{\sin i_r} + \\ C\sin i_r\cos u_{qj}\Delta t_{qj} \end{matrix}\right) \end{bmatrix} \tag{16c}$$

where $\tau_{qj} = \dot{\omega}_{J_2}\Delta t_{qj}$, $\Delta t_{q0} = t_{qf} - t_{q0} = dur_q$ is orbital transfer time, $\Delta t_{qj} = t_{qf} - t_{qj}$, u_{qj} is the argument of latitude of the j^{th} maneuver, and $\Delta \mathbf{v}_{qj} = (\Delta v_{qjx}, \Delta v_{qjy}, \Delta v_{qjz})^T$ is the impulse vector. The orbital coordinate system used to describe the impulse is given as follows: x is along the orbital radial direction, y is along the in-track direction and z is along the orbital normal direction and completes the rand-handed system. The last maneuver is executed at t_{qf}, i.e. $t_{qf} = t_{q2}$.

Eq. (16) is a linear relative dynamic equation under the J_2 perturbation. Only two maneuvers are considered for each orbital transfer that six unknown impulse components correspond to six equations, and then the solution to Eq. (16) can be easily obtained using Gaussian elimination. The details of this linear dynamics model can be found in the references [14, 15].

Long-duration rendezvous problems under the J_2 perturbation have multiple local minima both in the duration of one orbital period and in the duration of multiple orbital period [15]. In order to overcome the property of multiple local minima in one orbital period, the burn time of the first maneuver t_{q1} is enumerated from t_{q0} to $t_{q0} + T_r$ with a step of T_r/N_{enum}, where T_r is the reference orbital period and N_{enum} is the number of enumerations. For each value of t_{q1}, a group of values for $\Delta \mathbf{v}_{q1}$ and $\Delta \mathbf{v}_{q2}$ can be obtained, and is referred to as $\Delta \mathbf{v}_{q1}(t_{q1})$ and $\Delta \mathbf{v}_{q2}(t_{q1})$. The $N_{enum} + 1$ groups of $\Delta \mathbf{v}_{q1}(t_{q1})$ and $\Delta \mathbf{v}_{q2}(t_{q1})$ in total are calculated and then are compared with each other to find the group with the local minimum value of $\|\Delta \mathbf{v}_{q1}(t_{q1})\| + \|\Delta \mathbf{v}_{q2}(t_{q1})\|$, and the values of the $\Delta \mathbf{v}_{q1}$ and $\Delta \mathbf{v}_{q2}$ in this group are used as the impulses for the orbital transfer of the q^{th} rendezvous.

Based on the method provided above, the maneuver impulses of each rendezvous orbital transfer are only functions of the initial state, the required ending state and the orbital transfer time, and then the propellant cost can be evaluated with small computation cost.

6 Optimization for the Debris-to-debris Transfer

6.1 Optimization Model

1) Design variables

$4n$ design variables are contained in an n-impulse maneuver plan:

$$\boldsymbol{D} = [t_i, \Delta v_{ix}, \Delta v_{iy}, \Delta v_{iz}], \quad i = 1, 2, ..., K \tag{17}$$

where K is the total number of the maneuvers, t_i is the i^{th} maneuver time and $\Delta v_i = [\Delta v_{ix}, \Delta v_{iy}, \Delta v_{iz}]^T$ is the i^{th} maneuver impulse vector. Herein, 4-impulse maneuver plan is adopted.

2) Objective function

The objective is to minimize the total velocity increment:

$$\min J = \Delta v = \sum_{i=1}^{K} \|\Delta v_i\| \tag{18}$$

3) Constraints

The duration between two adjacent maneuvers should be larger than a given value, i.e.,

$$\begin{cases} t_i - t_{i-1} \geq \Delta T_i, \\ t_i \in [t_0, t_f], \quad i = 1, 2, ..., K \end{cases} \tag{19}$$

where $t_0 = 0$, $t_f = 30$ days, $\Delta T_1 = 5$ days, ΔT_i can be set as zeros for $i = 2, ..., K$. In addition, at the final time, the deviation between the spacecraft's state vector $x_f = [r_f, v_f]^T$ and the state vector x_{next} of the next debris should be smaller than the given tolerant error, i.e.,

$$\begin{cases} \|\boldsymbol{r}_f - \boldsymbol{r}_{\text{next}}\| \leq 100 \text{ m}, \\ \|\boldsymbol{v}_f - \boldsymbol{v}_{\text{next}}\| \leq 1 \text{ m/s} \end{cases} \tag{20}$$

6.2 Feasible Solution Iteration Optimization Approach

Based on the impulsive maneuver assumption, a feasible solution iteration approach is used to solve this optimization problem, which can be divided into the following two parts.

1) Dealing with the Linear Constraints

A group of proportionality coefficients $\eta_1, \cdots, \eta_K \in [0, 1]$ is used to substitute the maneuver times $t_1, \cdots, t_K$ as optimization variables. Then, the maneuver times can be calculated as

$$\begin{aligned} &t_i = t_{i-1} + \eta_i(t_f - t_{i-1}) + \Delta T_i \\ &\begin{cases} \Delta T_1 = 5 \text{ days}, \\ \Delta T_i = 0, i = 2, ..., K \end{cases} \end{aligned} \tag{21}$$

2) Dealing with the Nonlinear Constraints

The last two impulses $\Delta \boldsymbol{v}_{K-1}$ and $\Delta \boldsymbol{v}_K$ are chosen to satisfy the nonlinear equality constraints, and that they are obtained by solving a perturbed two-point boundary value problem. Otherwise, the number of design variables in Eq. (17) is reduced to $4K-6$, and the equation can be expressed as

$$\boldsymbol{X} = [\eta_1, \cdots, \eta_K, \Delta\boldsymbol{v}_1, \cdots, \Delta\boldsymbol{v}_{K-2}] \tag{22}$$

After the proportionality coefficients and the first $K-2$ nominal impulses $\Delta\boldsymbol{v}_i\,(i = 1, 2, \cdots, K-2)$ are provided by the optimization algorithm, the maneuver time can be computed using Eq. (21). Then the spacecraft's trajectory is propagated to t_{K-1} by substituting $\Delta\boldsymbol{v}_i\,(i = 1, 2, \cdots, K-2)$ into the dynamics of Eq. (2), and the spacecraft's state $\boldsymbol{x}(t_{K-1})$ can be obtained. Following this, the last two nominal impulses are computed by solving a two-point boundary value problem so that the final rendezvous conditions of Eq. (20) can be automatically satisfied. Here the homotopic perturbed Lambert algorithm proposed by Yang et al. [10] is used to calculate these two impulses as described by:

$$(\Delta v_{K-1}, \Delta v_K) = Lambert\_p\left((t_{K-1}), \boldsymbol{x}(t_K), t_K - t_{K-1}\right) \tag{23}$$

where $\boldsymbol{x}(t_K) = \boldsymbol{x}_{\text{next}}$, and the position and velocity error tolerances for the perturbed Lambert algorithm are respectively set as 100 m and 1 m/s. This perturbed Lambert algorithm allowed the perturbed solutions that included the successful computation of the gravitational potential terms J_2 through a homotopic targeting technique in which the two-body Lambert solution is used as an initial value and the Runge-Kutta integration is used as a perturbed trajectory propagator. A set of middle target points along the position offset vector (i.e., the offset between the initial and the final perturbed trajectories) is chosen to approach the final target point iteratively so that the iteration from two-body Lambert solution can converge for this long-duration, multi-revolution Lambert problem.

7 Results

Table 1 presents the best solution we obtained during the competition, in which the start and end epoch as well as the sequence and the start mass of each mission are listed. It can be found that the number of the removal debris in each mission are mainly distributed from seven to twelve except for the first mission.

The total velocity increments for rendezvous of each mission are presented in Fig. 4. It can be seen that the total velocity increments of most missions are between 1500 m/s and 2500 m/s while only that of the fifth mission is beyond 3000 m/s. However, it should be noticed that the first mission has also removed the most debris. Consequently, the average velocity increments of each mission are better indexes to evaluate the performance of each mission, which are shown in Fig. 5. We find that the average velocity increments of the first four missions are below 250 m/s while for most of other missions the average velocity increments are near 300 m/s. It indicated that the performances of the first four missions are better than others. It is clear that the average velocity increment of the eighth mission is the largest with a number of near 400 m/s, which indicates that the mission is not optimal. The minimum and maximum velocity increments of each mission are illustrated in Fig. 6. The smallest velocity increment of all 12 missions is 38.6 m/s while the largest one is 798.3 m/s. It can be seen that the range of velocity increments for a single rendezvous process of each mission is very wide.

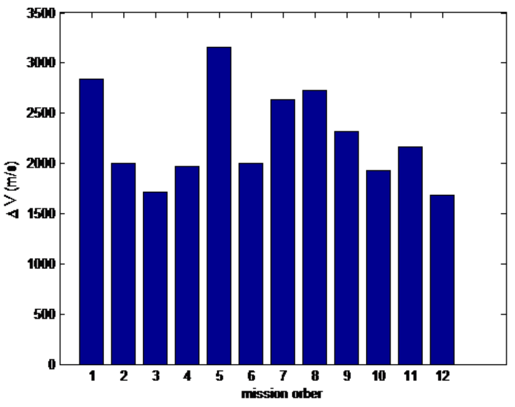


FIGURE 4. *Total ΔV of each mission*

The histories of the RAAN of the active spacecraft

TABLE 1. *Details of the best solution from NUDT*

Mission Order	Start Epoch (MJD)	End Epoch (MJD)	Debris Number	Debris Removal Sequence	Start Mass (kg)
1	23517.00	23811.52	17	0, 115, 12, 67, 19, 48, 122, 7, 63, 61, 82, 107, 41, 11, 45, 85, 47	5478.12
2	23893.80	24092.29	11	58, 28, 90, 51, 72, 69, 10, 66, 73, 64, 52	4106.88
3	24122.30	24427.74	12	84, 86, 103, 16, 121, 92, 49, 23, 20, 54, 27, 36	3809.97
4	24461.50	24660.15	10	8, 43, 9, 55, 95, 14, 102, 39, 113, 110	4081.09
5	24785.00	24975.41	12	83, 75, 22, 35, 119, 24, 108, 37, 112, 104, 32, 114	5782.68
6	25006.00	25198.32	9	118, 65, 74, 50, 94, 21, 97, 79, 120	4024.43
7	25281.60	25454.87	10	62, 1, 40, 76, 89, 99, 15, 59, 98, 116	4877.61
8	25555.40	25669.64	8	117, 91, 93, 70, 18, 105, 88, 46	4909.98
9	25702.40	25860.22	9	5, 53, 33, 68, 71, 80, 57, 60, 106	4419.99
10	25912.74	26055.85	8	2, 81, 96, 6, 100, 30, 34, 26	3902.24
11	26087.53	26262.18	10	87, 29, 101, 31, 38, 25, 4, 77, 13, 3	4267.35
12	26292.26	26381.58	7	44, 111, 56, 78, 17, 109, 42	3584.37

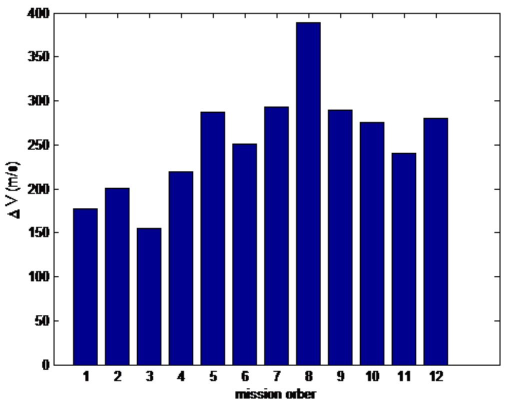


FIGURE 5. *Average* ΔV *of each mission*

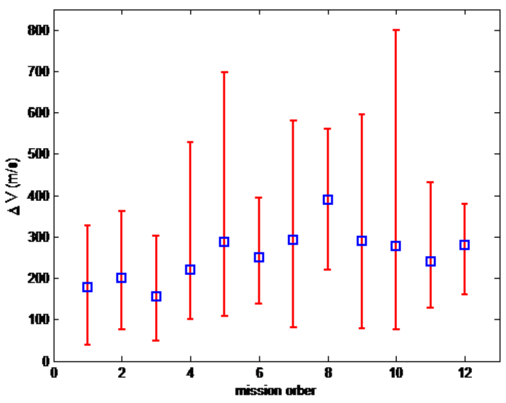


FIGURE 6. *Minimum and maximum* ΔV *of each mission*

and corresponding debris removed in the last mission are shown in Fig. 7, where the red line with circles indicated the history of RAAN of the spacecraft. We can find that the RAAN of the spacecraft increases gradually as it rendezvouses the debris one by one. The RAAN of debris #42, as shown in the figure, is not close to others in this sequence. However, there is an intersection of the RAAN between debris #42 and debris #109 around 26377 MJD. Thus, the spacecraft can wait to transfer from debris #42 to debris #109 until to that epoch so as to reduce the velocity increments for maneuvers caused by a large initial RAAN error.

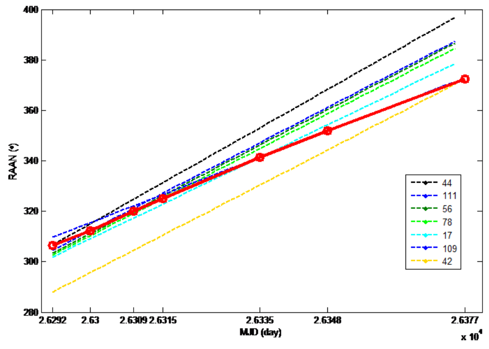


FIGURE 7. *History of the RAAN of the active spacecraft and corresponding debris for mission #12*

8 Conclusions

A three-level optimization framework is presented to solve the problem of GTOC9, wherein the competitors are called to design a series of missions able to remove a set of 123 orbiting debris pieces while minimizing the overall cumulative cost. The top level is similar to a dynamic TSP, wherein the debris pieces are divided into several groups and each group of debris is removed by one mission. The middle level is a mixed-integer optimization problem, wherein the impulses and durations of each rendezvous in one mission are designed. And the bottom level is the precise and detailed optimization of the flight trajectory in one rendezvous. The result of GTOC9 obtained by this framework is then illustrated. The result indicates that the three-level optimization framework is efficient and can obtain good solutions in considerable time.

Acknowledgement

The NUDT team thanks the Advanced Concepts Team of the European Space Agency for organizing such a wonderful GTOC. The innovation of the real-time leaderboard makes GTOC9 more exciting than the past edition and the interesting problem brings us many ideas for further research.

References

[1] GTOC Portal. `https://sophia.estec.esa.int/gtoc_portal/`.

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

[3] D. Izzo, I. Getzner, D. Hennes, and L. F. Simes. Evolving solutions to TSP variants for active space debris removal. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO*, pages 1207–1214. ACM Press, 2015.

[4] J. Zhang, G. T. Parks, Y. Z. Luo, and G. J. Tang. Multispacecraft refueling optimization considering the J2 perturbation and Window Constraints. *Journal of Guidance, Control, and Dynamics*, 37(1):111–122, 2014.

[5] Y. Z. Luo, G. J. Tang, Y. J. Lei, and H. Y. Li. Optimization of Multiple-Impulse, Multiple-Revolution, Rendezvous-Phasing Maneuvers. *Journal of Guidance, Control, and Dynamics*, 30(4):946–952, 2007.

[6] Y. H. Zhu, Y. Z. Luo, K. C. Tan, and X. Qiu. An Intelligent Packing Programming for Space Station Extravehicular Missions. *IEEE Computational Intelligence Magazine*, 12(4):38–47, 2017.

[7] S. Chatterjee, C. Carrera, and L. A. Lynch. Genetic algorithms and traveling salesman problems. *European Journal of Operational Research*, 93(3):490 – 510, 1996.

[8] K. Deb and A. Pratap and S. Agarwal and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[9] Z. Yang, Y. Z. Luo, and J. Zhang. Robust Planning of Nonlinear Rendezvous with Uncertainty. *Journal of Guidance, Control, and Dynamics*, 40(8):1954–1967, 2017.

[10] Z. Yang, Y. Z. Luo, J. Zhang, and G. J. Tang. Homotopic Perturbed Lambert Algorithm for Long-Duration Rendezvous Optimization. *Journal of Guidance, Control, and Dynamics*, 38(11):2215–2223, 2015.

[11] Y. H. Zhu, H. Wang, and J. Zhang. Spacecraft Multiple-Impulse Trajectory Optimization Using Differential Evolution Algorithm with Combined Mutation Strategies and Boundary-Handling Schemes. *Mathematical Problems in Engineering*, 2015:1–13, 2015.

[12] Y. Z. Luo, H. Y. Li, and G. J. Tang. Hybrid Approach to Optimize a Rendezvous Phasing Strategy. *Journal of Guidance, Control, and Dynamics*, 30(1):185–191, 2007.

[13] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[14] P. Labourdette and A. A. Baranov. Strategies for on-orbit rendezvous circling Mars. *Advances in the Astronautical Sciences*, 109:1351–1368, 2002.

[15] J. Zhang, Y. Z. Luo, and G. J. Tang. Hybrid planning for LEO long-duration multi-spacecraft rendezvous mission. *Science China Technological Sciences*, 55(1):233–243, 2012.

Acta Futura 11 (2018) 49-55

DOI: 10.5281/zenodo.1139240

Acta Futura

GTOC9: Results from the Xi'an Satellite Control Center (team XSCC)

HONG-XIN SHEN[*], TIAN-JIAO ZHANG, AN-YI HUANG, ZHAO LI

STATE KEY LABORATORY OF ASTRONAUTIC DYNAMICS, XI'AN SATELLITE CONTROL CENTER, XI'AN, CHINA

Abstract. This paper describes methods used by the team from the Xi'an Satellite Control Center (XSCC) for solving the 9th Global Trajectory Optimization Competition (GTOC9) problem. The removal of all 123 pieces of debris is accomplished using 12 launches in about 8 years time span, and the performance index finally ranked third in the competition. We refined our results after the competition, and the improved solutions are also presented.

1 Introduction

The 9th global trajectory optimization competition (GTOC9) problem concerned the multiple debris removal in low Earth orbit in order to relieve the Kessler effect [1]. This was accomplished by multiple missions, where each mission is a multiple-rendezvous spacecraft trajectory where a subset of size N of the 123 orbiting debris is removed by the delivery and activation of N de-orbit packages. The goal was to rendezvous as many debris objects as possible. A performance index, which depended on the number of launches used penalized by an added quadratic cost that including the sum of propellant mass and de-orbiting kits, must be minimized subject to a variety of constraints. Earlier submissions of single missions were rewarded through a smaller base cost. The only manoeuvres allowed to control the spacecraft trajectory are instantaneous changes of the spacecraft velocity (specific impulse was 340s); the tour should last less than 2947 days. Details can be found in Reference [1]. In this paper, we describe the main design model and optimization methods for this problem.

The complexities of this problem are to manage multiple-missions and select suitable sequence to rendezvous all debris objects from a given set of 123. Since the overall sequence of missions / debris removed is too large to be explored exhaustively, a global optimization procedure based on Ant Colony Optimization (ACO) is employed to automatically generate a near-optimal solution, as a replacement of the global search [2].

2 Leg cost estimation and optimization

Optimal phasing is assumed to obtain a preliminary estimate of the transfer velocity increment ΔV. The Hohmann transfer cost for a small change of the semimajor axis Δa reduces to $\Delta V/V = 0.5\Delta a/a$; similarly, for a small eccentricity change, the relation Δe, $\Delta V/V = 0.5\Delta e$ holds while $\Delta V/V = 2\sin(0.5\Delta i)$ applies for inclination changes. An empirical relation is thus introduced to consider simultaneously the change of semimajor axis Δa, to account for the additional eccentricity change Δe as well as for the change of incli-

[*]Corresponding author. E-mail: hongxin.shen@gmail.com

nation Δi [3]:

$$\Delta V'/V = 0.5\sqrt{(\Delta a/a)^2 + \Delta e^2} + 2\sin(0.5\Delta i) \quad (1)$$

where $\Delta V'$ is one part of the total velocity increment ΔV for a body-to-body transfer. The smallest value of semimajor axis a among the two objects involved in the leg, and the corresponding circular velocity V are used.

Suitable object sequences are found by estimating the rendezvous times and cost of each leg in terms of velocity change ΔV. The estimation procedure assumes that favorable opportunities occur only when the required plane change is the smallest, and therefore when the RAAN of the chaser is close to that of the target. To this purpose, the mission can take advantage of the J2 perturbation, which changes the RAAN of bodies orbiting the Earth with a rate that depends on semimajor axis and eccentricity. Objects with different orbits will therefore have different rates of change of Ω. The cost required for a small RAAN change $\Delta\Omega$ is

$$\Delta V''/V = 2\sin(0.5\Delta\Omega) \quad (2)$$

where $\Delta V''$ is the other part composing the total velocity increment ΔV for a body-to-body transfer. Therefore, one obtains the total cost of a leg as $\Delta V = \Delta V' + \Delta V''$.

The search starts from finding suitable times for the transfer between any debris pair (objects j and k). Each leg starts in rendezvous conditions with object j. It is necessary to wait for the RAAN difference $\Delta\Omega$ between objects j and k to become minimal in a prescribed time range in order to perform the transfer with a small amount of propellant consumption. Assuming t^j as the leg starting epoch, the rendezvous time for the next object k is denoted with t^{jk}. According to the GTOC9 rules the transfer time for each leg ranges between 5days and 30days, taking servicing time into account. Thus, $t^j + 5.5 = t^{jk}_{min} \le t^{jk} \le t^{jk}_{max} = t^j + 30$, where we have reserved a buffer of 0.5days for the orbit transfer time. Considering all possible debris pairs, the times when this favorable condition occurs are computed as:

$$t^{jk} = \begin{cases} \text{solve } \Delta\Omega(t) = 0, \text{if } \Delta\Omega(t^{jk}_{min})\Delta\Omega(t^{jk}_{max}) < 0, \text{else} \\ t^{jk}_{min}, \text{if } |\Delta\Omega(t^{jk}_{min})| < |\Delta\Omega(t^{jk}_{max})| \\ t^{jk}_{max}, \text{otherwise} \end{cases} \quad (3)$$

These times represent the theoretical rendezvous times for legs connecting the two objects; they are valid both for transfer from object j to object k and vice versa. Multiple-leg missions can be built based on the theoretical rendezvous times that have been determined with the procedure described above. All the permutations of m targets selected among the n possible objects should be considered. Mission starts at t_1 from target 1; rendezvous with target $i+1$ at the end of the i-th leg ($i = 1, 2, \ldots, m-1$) is assumed to happen at the best orbit alignment, that occurs after t_i, between the orbit planes of targets i and $i+1$. Because the number of targets and the dimension of the target set are relatively large, all the sequences cannot be evaluated in reasonable times. Pruning technique or an global optimization strategy is required to find the best sequences when the number of possible solutions becomes too large. Instead of pruning technique, an ant colony optimization approach is used in this paper to obtain an near-optimal solution, by solving a variant of the well-known travelling salesman problem.

The actual missions are designed acounting for the full dynamics complexity and constraints by means of an ant colony optimization approach in continuous domain [4], which is used to obtain the optimal solution, i.e., minimum ΔV, for each single transfer leg.

Assume the i-th leg starts at time $\tau_1 = t_i$, which is known from the solution of the previous leg, taking servicing time into account. Impulses are applied at times $\tau_2 < \tau_3 < \tau_4 = t_{i+1}$. The rendezvous/last impulse time $\tau_4 = t_{i+1}$ is unknown and only an estimation is available. Each arc is described by a limited set of 6 variables. Three variables define impulse times: $p_1 = \tau_4 - \tau_1$, $p_2 = (\tau_2 - \tau_1)/(\tau_4 - \tau_1)$, $p_3 = (\tau_3 - \tau_2)/(\tau_4 - \tau_2)$; the time of flight p_1, p_2 and p_3 vary between 0 and 1. p_4 is the impulse velocity change ΔV_1, which varies between 0 and 800 m/s. p_5 and p_6 define the direction of the first impulse.

Given the set of variables it is possible to evaluate the arc ΔV. Initial time, position and velocity are known. Once the velocity after the impulse has been evaluated from the optimization variables, Kepler's problem is solved taking J2 perturbation into account, to evaluate position and velocity just before the following impulse (2-); again, the optimization variables provide the velocity components after the impulse (2+), and position and velocity at 3- are determined solving a perturbed Kepler's problem. The arc from point 3 to point 4 (the target position at t_4 is also evaluated by solving a perturbed Kepler's problem) is first solved as an unperturbed Lambert's problem to obtain the velocity components after the second impulse that would allow to intercept the target in the absence of perturbations. J2 effect would not allow the intercept with these values, so they are corrected with an iterative scheme to nullify the error between the (perturbed) positions at t_4 of chaser and target. The scheme is based on Newton's method and employs the numerical derivatives of final position with respect to initial velocity components [5]. For the legs with relatively long transfer time, four impulses may be required, and the procedure can fit this situation easily with minor changes.

Each arc is solved in sequence, starting with the values obtained at the end of the previous one. It is important to note that the optimal strategy for favorable opportunities is often to wait on the initial orbit for a relatively long time, until the orbit planes have become sufficiently close. Estimations and results of the leg optimization have been compared for a large number of debris pairs. Differences in terms of rendezvous times are typically limited at 1 or 2 days. However, optimization of ΔV usually are remarkably smaller than the corresponding estimations, particularly for high-ΔV legs. Thus, we discount the estimation typically by $\Delta V = 0.7(\Delta V' + \Delta V'')$, where $\Delta V'$ and $\Delta V''$ are given by Eq. (1) and Eq. (2).

3 Formulation of a variant of TSP

We consider an Active Debris Removal(ADR) task with n missions which are responsible for the removal of a given set $S = \{s_0, s_1, ..., s_N\}, N = 122$ of 123 orbiting debris over a time horizon $[0, 2947]$ with reference time 23467[MJD2000]. The ADR task can be represented by a graph $G =< S, E >$ where the nodes are defined by the debris objects. The directed edge $(s_i, s_j) \in S \times S, s_i \neq s_j$ corresponds to the orbital

Algorithm 1 *ACO for the ADR problem*

```
 1: Set parameters, initialize pheromone trails
 2: while The stopping criterion is not met do
 3:   for all ants $k = 1, \dots, K$ do
 4:     set mission index: $m = 1$
 5:     while there are unselected debris do
 6:       $Can(\Pi^{p,k})$ is the set of all unselected debris objects
 7:       choose a debris $s_i$ randomly from $Can(\Pi^{p,k})$ as the start one in the $m$-$th$ mission
 8:       remove the forbidden debris according to both the time constraints and tour length limitation
 9:       while $Can(\Pi^{p,k}) \neq \emptyset$ do
10:         choose a debris $s_j \in Can(\Pi^{p,k})$ with probability $p_{ij}^k$
11:         add the chosen debris to the $m$-$th$ sub-path $\Pi_m^{p,k}$
12:       end while
13:       $m = m + 1$
14:     end while
15:     employ the 2-$Opt$, $insertion$ and $swap$ operators to improve the combined sequence $\Pi^k$
16:     divide the combined sequence into sub-paths $\{\Pi_1^k, \Pi_2^k, \dots, \Pi_n^k\}$ according to either time constraints or length limitation
17:   end for
18:   for all ants $k = 1, \dots, K$ do
19:     evaluate the solution $f(\Pi_1^k, \Pi_2^k, \dots, \Pi_n^k)$
20:     update the best-so-far solution $\Pi_1^*, \Pi_2^*, \dots, \Pi_n^*$
21:   end for
22:   update pheromones on the best-so-far sub-paths
23: end while
24: return $\Pi_1^*, \Pi_2^*, \dots, \Pi_n^*$
```

transfer made by a spacecraft from object s_i to object s_j. Moreover, each edge has an associated transfer cost Δv_{ij} in terms of velocity change as well as Δm_{ij} in terms of fuel usage, it is characterized by a number of factors: $\Delta v_{ij} \equiv \Delta v_{ij}(s_i, s_j, t_i, t_j)$, where t_i and t_j, respectively, are the departure and the arrival date associated with the transfer. Thus, a single mission can be viewed as an open sub-path in the graph. To be feasible, a sub-path must pass through each selected debris object only once, the servicing time between two successive debris rendezvous within the same mission should be greater than 5.5 days and not exceed 30 days. Moreover, the time gap of at least 30 days must be accounted for any two missions, and all the mission events must be finished within the allowed time horizon. The goal is to design a series of sub-paths able to cover all the 123 nodes, while minimizing the sum of overall mission costs and the penalty term related to the possible incomplete removals. The ADR problem is analogous to the Traveling Salesman Problem(TSP) [6], but there are two main differences between them:

- Instead of a single closed path visiting all the nodes in TSP, the debris are covered by several sub-paths(mission). The length of sub-path cannot exceed the upper limit Δv_{max}. Therefore, the overall path length is measured as the sum of all sub-path lengths.
- Unlike the constant distances between node pairs in the TSP, the debris are moving with different precession rates. Accordingly, the time varying cost of going from one debris object to another makes the ADR problem time-dependent.

For the sake of clarity, we present an integer programming formulation for the ADR problem. It is based on two sets of binary decision variables, y_{ik}, designating the removal of debris s_i to the k-$th(1 \le k \le n)$ mission by the value 1 (and 0, otherwise), and $x_{ij}, i, j \in \{0, 1, \dots, N\}$, determining the debris disposal sequence in a single mission. x_{ij} takes the value 1 when the vehicle proceeds from debris s_i to debris s_j. Clearly, the search space defines over two finite sets of discrete decision variables $\{x_{ij}, i, j \in \{0, 1, \dots, N\}\}, \{y_{ik}, i \in \{0, 1, \dots, N\}, k \in \{1, 2, \dots, n\}\}$. A solution in the ADR problem can be represented through a set of N variables, each of which is associated with a debris object. Here, solution components are debris, and they are to be visited in the order appearing in the solution. The decision variables x_{ij}, y_{ik} indicate the debris to be removed in what order and in which mission. Hence, the ADR problem can be stated as follows:

$$\begin{aligned}
\text{min} \quad & cn + \alpha \sum_{k=1}^{n} (\sum_{\forall s_i \in S} \sum_{\forall s_j \in S} x_{ij} y_{ik} \Delta m_{ij})^2 && (4) \\
\text{s.t.} \quad & \sum_{\forall s_i \in S} x_{ij} \le 1, \forall s_j \in S && (5) \\
& \sum_{\forall s_j \in S} x_{ij} \le 1, \forall s_i \in S && (6) \\
& \sum_{\forall s_i \in S} \sum_{\forall s_j \in S} x_{ij} y_{ik} = \sum_{\forall s_i \in S} y_{ik} - 1, \forall k \le n && (7) \\
& \sum_{\forall s_i \in S} \sum_{\forall s_j \in S} x_{ij} y_{ik} \Delta v_{ij} \le \Delta v_{max}, \forall k \le n && (8) \\
& 5.5 \le t_j - t_i \le 30, \text{if } \exists k \le n, \forall s_i, s_j \in S, x_{ij} y_{ik} y_{jk} = 1 && (9) \\
& t_j - t_i \ge 35, \text{if } \exists k \le n, \forall s_i, s_j \in S, y_{ik} + y_{jk} = 1 && (10) \\
& t_j \le 26414, \text{if } \exists k \le n, \forall s_j \in S, y_{jk} = 1 && (11) \\
& t_j \ge 23467, \text{if } \exists k \le n, \forall s_j \in S, y_{jk} = 1 && (12) \\
& x_{ij}, y_{ik} \in \{0, 1\}, \forall s_i, s_j \in S, \forall k \le n && (13)
\end{aligned}$$

where c is the base cost of each mission ranging from 45 to 55 (increasing linearly during the competition time frame),n is the number of missions, α is set to be 2.0×10^{-6} according to the GTOC9 rules. Eq. (4) states the objective function, where the fuel usage Δm_{ij} is computed from impulsive Δv_{ij} using the Rocket equation. Eqs. (5) and (6) ensure that a selected debris cannot be removed more than once. Furthermore, Eq. (7) signifies that the adjacent chosen debris in one mission must be arrived by the spacecraft consecutively. From the graph theory point of view, Eqs. (5)-(7) make the selected edges compose a series of paths without loops. Eq. (8) is the length limit of velocity change for a single sub-path. Eqs. (9)-(12) define all the time constraints. Eventually, Eq. (13) imposes the restriction on the decision variables.

4 Ant colony optimization

Ant colony optimization, denoted as ACO, is a metaheuristic framework for solving static combinatorial optimization problem. It takes inspiration from the following behavior of ant species: ants are able to find the shortest path from their home to a food source over a period of time [7].

ACO solves an optimization problem by a construction graph and uses K artificial ants to walk on the graph where K is the size of ant colony. Each ant constructs a solution iteratively and its behavior is guided by pheromone and heuristic information. The construction graph for the sequence optimization problem is the static ADR mission graph, in which the nodes are debris and each edge represents a transfer between two debris objects. The ACO metaheuristic is shown in Algorithm 1. Its main iterative procedure consists of three steps. At first, each ant acts in the same manner: starting from an empty solution $\Pi^p = \emptyset$, the partial solution Π^p is extended incrementally by adding a debris object as a solution component from available candidates $Can(\Pi^p) \subset S$ according to a biased probabilistic mechanism. In particular, if a debris s_j has not been previously removed, it can be selected with the probability:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in Can(\Pi^{p,k})} \tau_{il}^\alpha \cdot \eta_{il}^\beta}, & \text{if } l \in Can(\Pi^{p,k}) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $Can(\Pi^{p,k})$ is defined as the set of debris objects that can be added to a sub-path by the k-th ant without violating either the time constraints or the limitation of the sub-path length. τ_{ij} is the pheromone of edge (i, j) which corresponds to the transfer from the debris s_i to s_j. The heuristic information is chosen as $\eta_{ij} = \frac{\lambda_1}{\Delta v_{ij}} + \frac{\lambda_2}{\Delta t_{ij}}$, where λ_1 and λ_2 are nonnegative real parameters that control the relative importance between transfer cost and time. Furthermore, α and β are positive real parameters that control the relative importance between the pheromone and the heuristic information. Subsequently, a local search procedure is employed to improve every constructed solution. At the last step, pheromone is updated as follows: all pheromone trails are decreased uniformly through pheromone evaporation as to allow ants to forget bad solutions. Then, the pheromone deposited, the mount of which is proportional to the quality of the solution, guides subsequent ants to search in the promising regions of the search space. In this way, edges, associated with components in promising solutions, are reinforced with additional pheromone gradually. The iterative process terminates when a stopping condition is satisfied. In order to improve exploitation capacity of ACO, we use a modified version of the MAX-MIN ant system with respect to the classic one. In our algorithm, we propose an enhanced local search strategy by employing the 2-Opt, $insertion$ and $swap$ operators in turn. Once all these operators cannot find a better solution, local search stops. Otherwise, it continues until the stopping condition is met. The illustration of these operators is shown in Figure 1.

5 Results

A summary of the final solution submitted during the competition timeframe and achieving an objective function of 821MEUR, is provided in Table 1. Each mission ends with a dry mass $m_{dry} = 2000kg$. Shortly after the competition ended, an improved solution reaching an objective function value of 766MEUR was obtained refining all the body-to-body transfers, as given in Table 2. Under the competition pressure and stimulated by the ideas provided by the winner solution that made use of a reduced number of launches, we experimented with an increased weight of launches in Eq.(4). By doing so, our algorithms returned got several 10-mission solutions, among which the best oneachieving an objective function of 698MEUR is summarized in Table

Table 1. *Submitted solution*

Mission	Number of objects	Debris ID	Start MJD2000	End MJD2000	ΔV, m/s	Cost*
1	16	7,67,12,48,122,63,61,19,107,41,11,82,115,45,85,47	62.09	405.03	2433.44	71.49
2	10	58, 90, 51, 72, 69, 10, 66, 28, 52, 64	476.4	618.81	1844.81	62.12
3	12	86, 84, 103, 16, 121, 92, 49, 20, 27, 54, 23, 36	668.95	960.88	1758.58	61.95
4	11	8, 43, 9, 55, 95, 73, 14, 102, 39, 113, 110	991.04	1195.71	1957.82	63.31
5	11	83, 75, 35, 119, 24, 108, 37, 112, 104, 32, 114	1322.00	1506.76	3176.94	83.41
6	9	74, 50, 94, 21, 97, 79, 120, 109, 77	1603.00	1763.17	2390.13	67.34
7	12	62, 1, 40, 76, 89, 99, 0, 15, 87, 59, 98, 116	1793.35	2012.06	3148.97	82.79
8	9	93, 70, 31, 105, 46, 88, 118, 18, 117	2079.42	2227.17	2667.12	71.07
9	9	5, 106, 53, 33, 17, 60, 68, 80, 71	2257.17	2409.36	2371.04	67.33
10	10	6, 2, 65, 81, 96, 100, 30, 4, 34, 26	2439.36	2588.5	2608.18	70.97
11	6	3, 42, 44, 56, 78, 111	2619.67	2701.14	1264.68	57.63
12	8	101, 22, 29, 38, 25, 91, 13, 57	2731.98	2833.56	2186.65	64.66

*Base cost: 55 MEUR.

Table 2. *Improved submitted solution*

Mission	Number of objects	Debris ID	Start MJD2000	End MJD2000	ΔV, m/s	Cost
1	16	67,12,48,122,7,63,61,19,107,41, 11,82,115,45,85,47	73.06	393.40	1803.65	65.78
2	10	90,58,51,72,69,10,66,28,52,64	431.67	642.00	1147.32	59.16
3	12	86,84,103,16,121,92,49,23,20,54,27,36	672.00	969.09	1278.77	59.31
4	11	8,43,9,55,95,73,14,102,39,113,110	1000.00	1195.89	1885.13	64.75
5	11	83,75,35,119,24,108,37,112,104,114,32	1302.45	1530.06	2368.30	71.02
6	9	94,74,21,79,50,120,97,109,77	1580.41	1764.24	1883.52	64.37
7	12	1,62,40,76,89,99,0,15,87,59,98,116	1794.24	2017.40	2128.08	66.77
8	9	93,70,31,105,46,88,118,18,117	2047.40	2227.40	2015.48	64.61
9	9	5,53,106,17,60,80,33,68,71	2257.40	2405.00	1877.61	65.50
10	10	4,2,65,81,96,6,100,30,34,26	2436.58	2590.00	2186.32	68.22
11	6	3,42,44,56,78,111	2620.00	2697.00	927.96	57.23
12	8	101,22,91,13,57,25,38,29	2733.19	2937.13	1331.01	59.46

Table 3. *10-mission solution*

Mission	Number of objects	Debris ID	Start MJD2000	End MJD2000	ΔV, m/s	Cost
1	14	38,103,95,57,16,118,50,23,117,55,113,20,79,27	0	281.73	2713.2	74.99
2	17	88,77,31,104,48,39,91,21,11,70,63,47,8,82,45,7,41	311.73	705.50	2727.3	77.29
3	13	94,75,0,18,2,6,108,24,44,120,26,67,119	735.50	955.00	2195.6	66.61
4	9	13,114,96,74,46,32,105,83,89	985.00	1090.28	2374.0	67.16
5	10	64,30,66,28,69,14,93,90,19,9	1127.19	1274.32	2241.2	66.31
6	12	107,61,5,4,78,92,53,25,111,109,56,42	1304.32	1500.00	2528.3	70.81
7	17	84,51,36,1,40,62,54,99,122,35,76,85,98,59,15,121,112	1530.00	1899.74	2473.9	72.79
8	11	97,115,22,102,86,110,65,10,100,34,73	2067.92	2285.00	1967.0	63.53
9	16	3,17,43,60,80,106,12,52,71,116,68,33,58,72,37,49	2315.00	2662.36	2909.5	80.76
10	4	29,101,87,81	2692.36	2787.02	1501.5	58.33

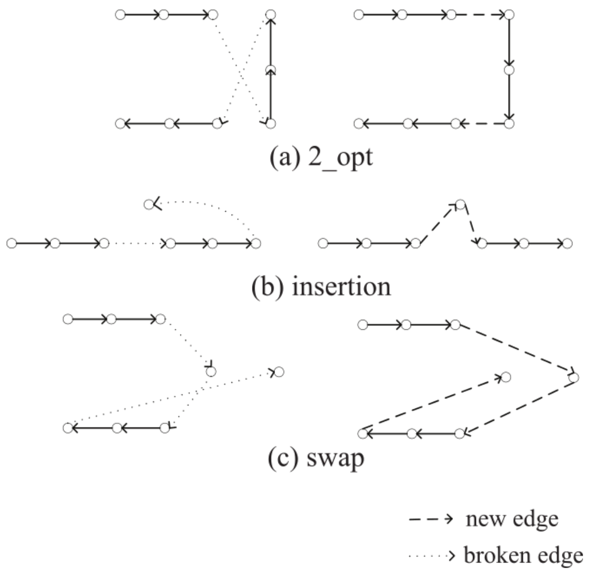


FIGURE 1. *The illustration of the local search operators, each of which transforms the left into the right. (a) 2-opt operator, (b) insertion operator, (c)swap operator.*

3. All the mission times in these tables refer to the permitted mission starting time i.e., 23467[MJD2000]. For fairness in the comparison, the maximum base cost, i.e., 55MEUR, is used for the above solutions.

6 Conclusions

The multiple debris rendezvous problem posed for the 9th edition of the GTOC is, essentially, a time-dependent combinatorial problem, and thus it can be casted into a dynamic variant to the standard Travelling Salesman Problem. Fortunately, multiple-revolutions transfers together with the time constraints considered in this particular competition allows for an accurate estimation of the single transfer cost and time, therefore the resulting dynamic TSP degenerates into a simpler version. As ant colony optimization ACO is one of the most acclaimed algorithms able tackle TSP problems [8], it was also implemented by the XSCC-ADL team to find near-optimal rendezvous sequences. An efficient local optimizer is also important to improve the overall search strategy adopted, as can be seen from the fact that we refined our submitted solution and got about 60MEUR performance index improvement only through local optimization. Finally, when we focused on the number of launches by adjusting the weight in the performance index, 10-mission solutions with remarkable performance improvement also can be found.

References

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

[2] Hong-Xin Shen, Tian-Jiao Zhang, Lorenzo Casalino, and Dario Pastrone. Optimization of Active Debris Removal Missions with Multiple Targets. *Journal of Spacecraft and Rockets*, pages 1–9, 2017.

[3] T. N. Edelbaum. Propulsion Requirements for Controllable Satellites. *ARS Journal*, 31(8):1079–1089, 1961.

[4] T. Liao, T. Stutzle, M. A. Montes de Oca, and M. Dorigo. A unified ant colony optimization algorithm for continuous optimization. *European Journal of Operational Research*, 234(3):597–609, 2014.

[5] L. Casalino and D. Pastrone. Active Debris Removal Missions with Multiple Targets. In *AIAA Paper 2014-4226*, 2014.

[6] D. Izzo, I. Getzner, D. Hennes, and L.F. Simoes. Evolving Solutions to TSP Variants for Active Space Debris Removal. In *Proceedings of the 17th annual conference on Genetic and evolutionary computation (GECCO 2015)*, 2015.

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

[8] L.F. Simoes, D. Izzo, E. Haasdijk, and A.E. Eiben. Multi-rendezvous Spacecraft Trajectory Optimization with Beam P-ACO. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, 2017.

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	N_L	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

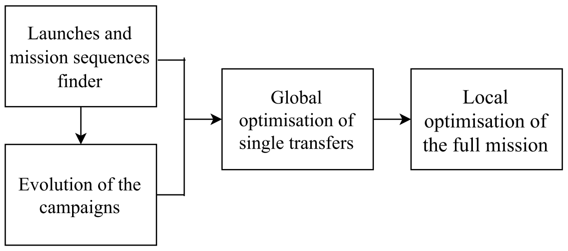

Launches and mission sequences finder
Evolution of the campaigns
Global optimisation of single transfers
Local optimisation of the full mission

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:

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

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:

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

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-

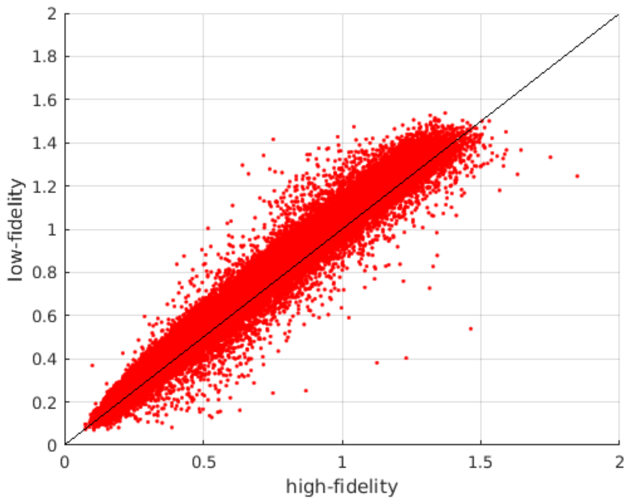


FIGURE 2. *Cost comparison (in meters per second) on 44683 cases between guess from Keplerian model and value from J_2 dynamics.*

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

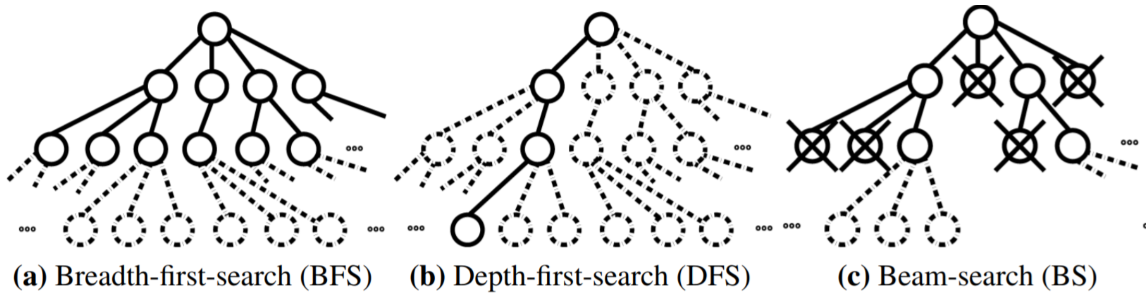


FIGURE 3. *Different tree search strategies in comparison. Dotted nodes are yet to be explored. Crossed out nodes are pruned and will not be branched [7].*

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 $\{(D_j, t_j)\ ,\ j \leq n)\}$, 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.

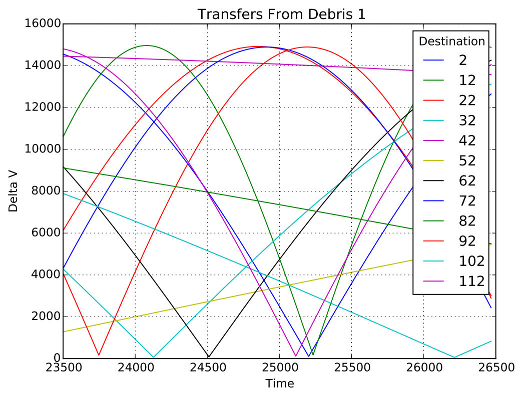


FIGURE 4. *Velocity required to reach a destination debris from a reference debris at a point in time.*

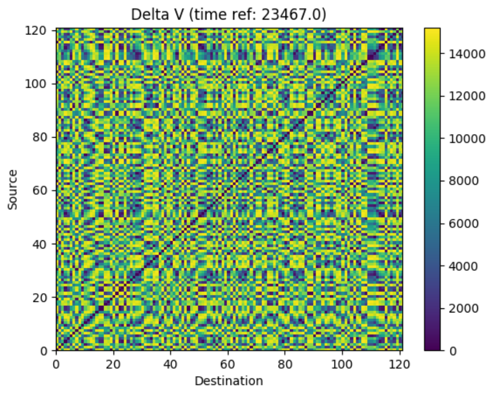


FIGURE 5. *Velocity required to transfer between debris at a reference time point.*

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:

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

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

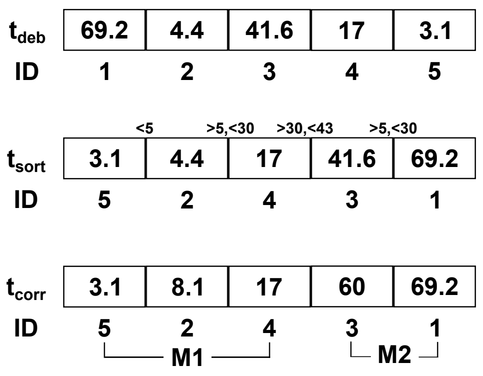


FIGURE 6. *Scheme of the Time Shuffler encoding and a possible time feasible solution*

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.

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).*

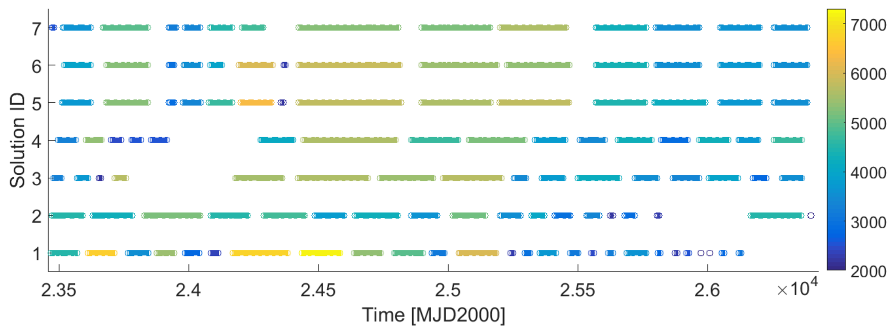


FIGURE 7. *Launches and debris removal epochs of the solutions in Table 2. Colour relates to initial mass of each of the launches.*

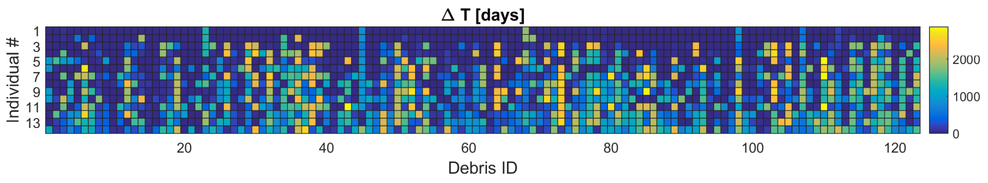


FIGURE 8. *Difference in time at debris between individuals in the initial population and final solution obtained after evolution. Most similar individuals on top.*

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

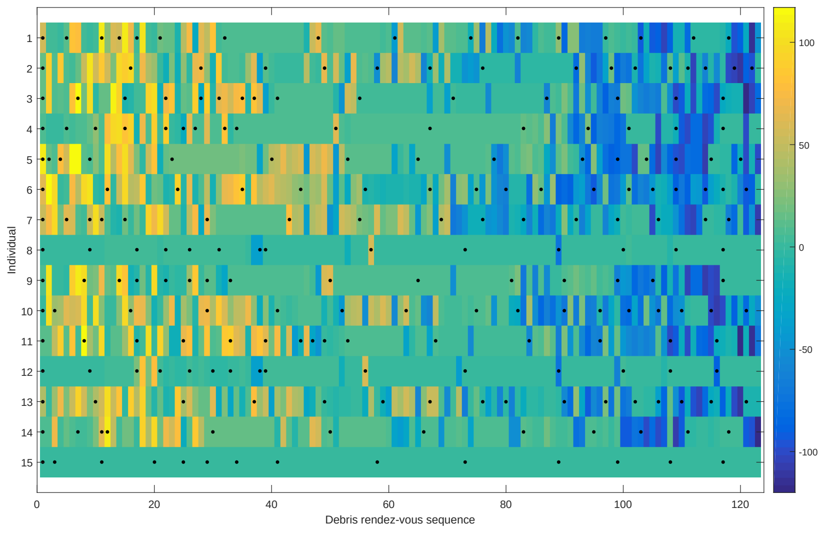


FIGURE 9. *Individual sequence similarity. Each row represents an individual's debris rendez-vous sequence, dots represent new mission launch. Colour details how many positions the rendez-vous with that debris ID is shifted in the sequence of the final submission. Individual 15 is the final submission.*

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-

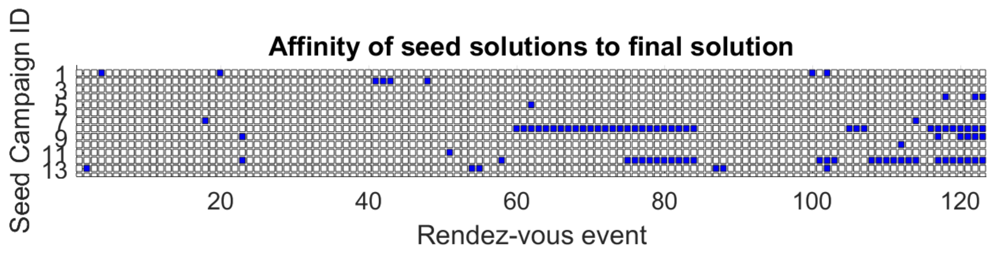


FIGURE 10. *Debris order affinity between seed solutions and final solution. A dot represents when the ID of the debris i-th rendezvous in chronological order of a seed campaign matches with the final one.*

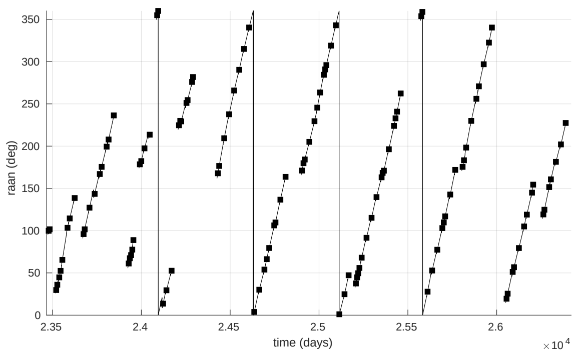


FIGURE 11. *Evolution in time of the RAAN of the final submitted solution.*

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.

Acta Futura 11 (2018) 71-77

DOI: 10.5281/zenodo.1139250

Acta Futura

GTOC 9: results from the German Aerospace Center (team DLR)

Marcus Hallmann[*], Markus Schlotterer, Ansgar Heidecker, Marco Sagliano
Federico Fumenti, Volker Maiwald, René Schwarz

Institute of Space Systems, DLR, Robert-Hooke-Str. 7, 28359 Bremen (Germany)

Abstract. This paper discusses the methods used by the team from the German Aerospace Center (DLR) for solving the 9th Global Trajectory Optimization Competition (GTOC) problem. The GTOC is an event taking place every year lasting roughly one month during which the best aerospace engineers and mathematicians world wide challenge themselves to solve a nearly-impossible problem of interplenatery trajectory design.

1 Introduction

The paper is organized as follows: section 2 summarizes briefly the problem statement; section 3 points out how the overall strategy was developed; section 4 focuses on the combinatorial part of the problem and 5 on the transfer between two debris. Finally, in sections 6 and 7 we discuss the results and draw some conclusions.

2 Problem Statement

The task was to design a scenario with n missions which collect a given set of 123 space debris on Sun-synchronous Low Earth Orbits. The following cost function has to be minimized:

$$J = \sum_{i=1}^{n} c_i + \alpha(m_{0_i} - m_{dry})^2 \tag{1}$$

where c_i is the base cost (increasing linearly during the competition time frame from 45 MEUR to 55 MEUR). Each spacecraft initial mass m_0 is the sum of dry mass, propellant mass and N times the deorbit package mass: $m_0 = m_{dry} + m_p + Nm_{de}$, with $m_{de} = 30$ kg. The α parameter is set to be $2.0 \cdot 10^{-2}$ MEUR/kg^2.

In order to control the spacecraft, five impulsive manoeuvres are allowed during the debris to debris transfer in addition to an impulsive manoeuvre at departure and at arrival. The overall time between two successive debris rendezvous, within the same mission, must not exceed 30 days. The deorbit package deployment takes 5 days. That results in a maximum transfer time of 25 days. The time between two missions must be at least 30 days. And the mission must take place between 23467 MJD2000 and 26419 MJD2000. The radius of pericenter r_p is constrained to be smaller than $r_m = 6600$ km.

The spacecraft dynamics is described by the follow-

[*]Corresponding author. E-mail: Marcus.Hallmann@dlr.de

ing set of Ordinary Differential Equations (ODE):

$$\begin{aligned}\ddot{x} &= -\tfrac{\mu x}{r^3}\left(1+\tfrac{3}{2}J_2\left(\tfrac{r_{eq}}{r}\right)^2\left(1-5\left(\tfrac{z}{r}\right)^2\right)\right)\\ \ddot{y} &= -\tfrac{\mu y}{r^3}\left(1+\tfrac{3}{2}J_2\left(\tfrac{r_{eq}}{r}\right)^2\left(1-5\left(\tfrac{z}{r}\right)^2\right)\right)\\ \ddot{z} &= -\tfrac{\mu z}{r^3}\left(1+\tfrac{3}{2}J_2\left(\tfrac{r_{eq}}{r}\right)^2\left(3-5\left(\tfrac{z}{r}\right)^2\right)\right)\end{aligned} \quad (2)$$

which describes a Keplerian motion perturbed by an oblate Earth. The orbital elements of the space debris are given for a certain epoch and are propagated via a more simplified model than equation (2):

$$\begin{aligned}\dot{\Omega} &= -\tfrac{3}{2}J_2\left(\tfrac{r_{eq}}{p}\right)^2 n\cos i\\ \dot{\omega} &= \tfrac{3}{4}J_2\left(\tfrac{r_{eq}}{p}\right)^2 n\left(5\cos^2 i-1\right)\end{aligned} \quad (3)$$

It can be seen that the ascending node Ω is the orbital element which encounters the most variations caused by J_2. That will have an impact on the overall strategy. For more details on the problem statement refer to the GTOC 9 problem statement [1] or visit https://kelvins.esa.int/gtoc9-kessler-run/.

3 Overall Strategy

The problem to be solved can be classified as Time Dependent Traveling Salesman Problem, with a nested optimal control problem for each transfer. One way to solve the combinatorial part would be to explicitly evaluate all possible combinations. That approach is only applicable for small dimensions. In this case there are 123 debris to be sorted for the best sequence (which gives 123! permutations). In addition they have to be chopped into n missions, which increases the dimension of the problem even more. Even with the most powerful computers and the smartest approach to calculate the ΔV for transfer from one debris to the next one it would take years to determine the entire tree. Section 4 deals about the solution of the combinatorial part of the problem.

Before looking into the transfer it makes sense to analyze the design space to get some reasonable boundaries for design variables like transfer time, delta v range, needed number of missions and so on. The first important question to answer is how are the debris pieces spread out regarding inclination i, eccentricity e and semi major axis a. Figure 1 shows the range of the orbital elements for the debris pieces. It can be seen that the orbits are nearly circular, inclination ranges from 96 deg to 102 deg and the orbital height (or $a - r_{eq}$) goes from 600 km to 900 km. These elements do not change over time for the dynamic model which is used for the debris.

If we only consider the change in inclination and semimajor axis during a transfer, the problem can be treated as a simple Traveling Salesman Problem (TSP). The ΔV which is needed to travel from debris A to debris B is the sum of the inclination change ΔV_{inc} plus the ΔV_{sma} for the Hohmann transfer:

$$\begin{aligned}\Delta V_{inc} &= 2V\sin((i_A-i_B)/2)\\ \Delta V_{sma} &= \sqrt{\mu/r_1}(\sqrt{2k/(1+k)}-1)+...\\ &\quad \sqrt{\mu/(r_1 k)}(1-\sqrt{2/(1+k)})\\ \Delta V_{AB} &= \Delta V_{inc}+\Delta V_{sma},\end{aligned}$$

where k is the ratio between a_A and a_B, assuming circular orbits. With that equations it is possible to set up a cost matrix showing the cost for the transfer from one debris to the next debris, ignoring the phasing in true anomaly and right ascension. With these assumptions it is possible to apply a genetic algorithm implemented in *Matlab* to find the optimal route. Figure 2 shows the result of one run with 500 populations and $1 \cdot 10^5$ iterations. The total distance is around 2654 m/s, so an average ΔV of 21.7 m/s is needed for one transfer. In theory and with no constraints on the mission time, this result would equal to $J < 100\,\text{MEUR}$. In practice the 8 years mission time constraint and the 25 day transfer time constraint has to be fulfilled.

So some reasonable ΔV has to be invested for changing the right ascension. That can be done in two ways:

- a direct plane change,
- an indirect plane change via a change in semi major axis.

Assuming that all other elements besides Ω are the same, one can compare the ΔV for both cases. With this assumption the equation for the direct plane change is similar to the one used for the inclination change.

$$\Delta V_{\Omega} = 2V\sin((\Omega_A-\Omega_B)/2) \quad (4)$$

Like the inclination change it is quite a cost intensive maneuver. If there is enough time available, an indirect transfer is cheaper. An important figure to look at is the

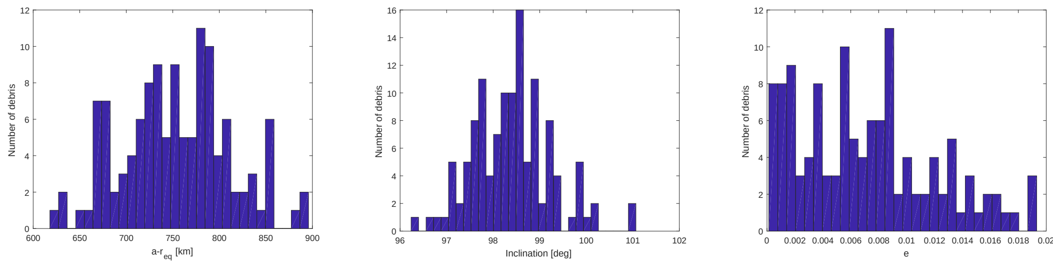


FIGURE 1. *Debris orbital elements*

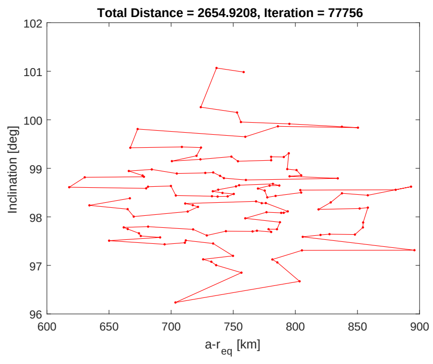


FIGURE 2. *Optimal Path in the i and a TSP*

change in right ascension for the debris which is plotted in Figure 3 over height (with an inclination equal to 98 deg). Out of that one can see that the orbits are

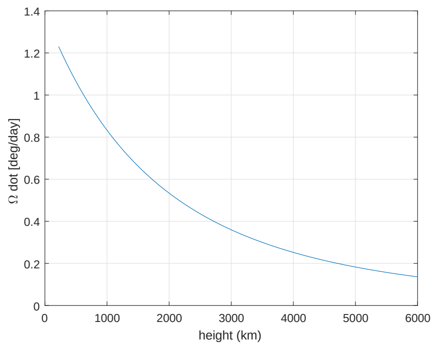


FIGURE 3. $\dot{\Omega}$ *over height*

drifting between 1.2 deg/day and 0.2 deg/day. Depending on the initial height the differential drift is around 0.5 deg/day. Assuming a 20 day transfer time it is possible to overcome a delta of 10 deg in Ω. For the indirect change two Hohmann like transfers are needed. The first one to reach the desired drift orbit, the second one to get the semi major axis of the arrival debris.

Assuming the same orbital elements for both debris as: $a = 600\,\text{km} + r_{eq}$, $e = 0$, $i = 98\,\text{deg}$, only a change in Ω needs to be considered. On Figure 4 one can see

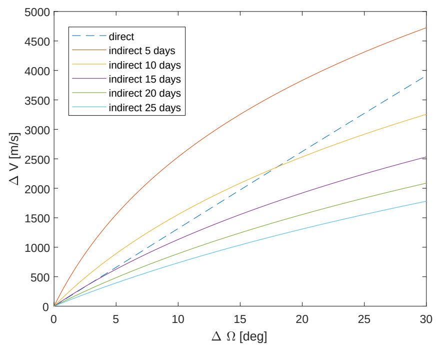


FIGURE 4. ΔV *over* $\Delta\Omega$

that the ΔV needed for a direct change increases nearly linear with $\Delta\Omega$ (dashed line). The other curves in the figure represent an indirect transfer for different transfer times (5:5:25 days). One can see that for transfer times larger than 15 days it is always better to make an indirect transfer. And that does even not take into account that it is possible to save some ΔV because of different semi major axis and inclination of the departing and arrival debris. The inclination change can either be performed before or after the drift change maneuvers. This choice also has an impact on the required ΔV, as it is a function of the inclination (see equation 3). With that thoughts one has a good set up for the combinatorial problem, which will be discussed later.

The next interesting question to look at is how many missions one may need (does it make sense to stack one launcher as full as possible) and what does the cost function look like. Assuming a range of different av-

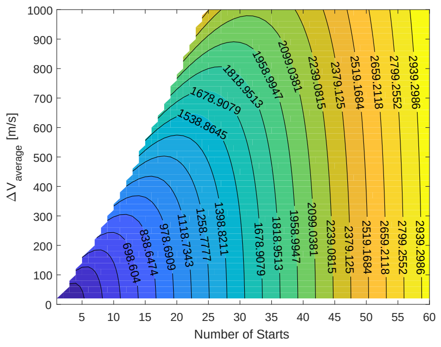


FIGURE 5. *J over n and $\Delta V_{average}$*

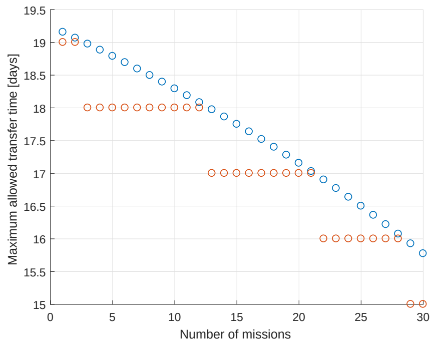


FIGURE 6. *Maximum transfer time over n*

erage transfer $\Delta V s$ and number of missions (n) one gets Figure 5, which shows the J function in MEUR. It can be seen that it is not advisable to use the maximum propulsion available. For an average ΔV of 300 m/s, J is 904.1 MEUR for 9 starts, while for 12 starts it is 827.6 MEUR and for a larger number of starts J increases again. Although one would have to add 12 times the base cost for the launcher instead of only 9 times, the used propellant mass goes in quadratic. So using the total allowed 5000 kg is not optimal. It is better to reduce the total allowed fuel or ΔV per mission by 10% to 20%, depending on how many missions are needed. The issue is that this number is not known before hand.

4 Combinatorial Problem

In section 3 basic figures have been derived to narrow down the problem. With these figures it is possible to estimate the cost or ΔV to perform a debris to debris transfer. The issue is, the problem is time variant, because the cost matrix depends on the epoch of the transfer. Or using the TSP syntax it is not a city to city routing problem, it is a boat to boat one, where the boats are sailing around.

Before going into detail of the graph implementation a deeper look into the handling of the transfer time is needed. As derived in section 3, for the drift change maneuver, it would be better to have a large transfer time available. At a first guess it might make sense to use the maximum allowed 25 days. But using that for all of the transfers, the total mission time would be larger than the allowed 8 years. And that even depends on the number of needed missions. Figure 6 shows the maximum allowed transfer times over the number of needed missions, which is again not known before solving the graph problem. In fact it is only possible to use average transfer times between 15 and 19 days. In that case the next question is, if it would make sense to have the transfer time as a design variable in the combinatorial problem or to keep it fixed to an average value. To not further increase the permutation space it was decided to keep it fixed. That approach also allows to use a look up table for all possible transfers with a 1 day grid size. This table is precalculated and loaded into the graph algorithm. In the cost matrix it is also possible to handle the radius of periapsis constraint, by setting the $\Delta V = \infty$ for all transfers where $r_p < r_{p_m}$.

For solving the routing of the time dependent problem the same genetic algorithm has been used, which already delivered good results for the inclination-sma routing problem. But it didn't brought any good results. Instead a graph algorithm which uses a certain beam width has been developed.

Each mission can be represented as a graph or tree (see Figure 7). For the first mission there are 123 nodes or debris as an option to start from. Keeping in mind that the cost function doesn't give any penalty at which debris the mission starts, it's a free design parameter. So the initial beam width would be 123. Using that one can calculate 122 possible debris transfers for each of that 123 debris. Because each debris should be only visited ones. This results in 123 times 122 possible options. There are many methods to explore such kind of trees or graphs:

- Depth First Search
- Breadth First Search
- Beam Search

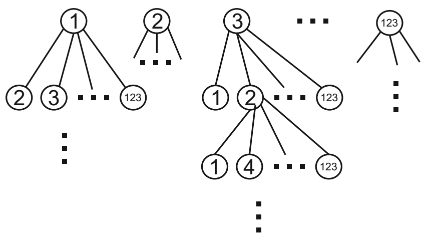


FIGURE 7. *Mission Graph*

- Greedy Search

The Depth First Algorithm travels along the left or right side of the tree. In this case it would just visit the debris either in the sequence 1:1:123 or 123:-1:1. It would be possible to add some backtracking if the algorithms gets stuck somewhere or to apply some heuristic, like the average ΔV is getting too large along the current path.

Breadth First in the opposite explores the tree first horizontal and than continues to the next level with all possible permutations. In this case like mentioned already a full breadth first search would not be possible cause the permutation would be to large.

Greedy search is like depth first, but it makes a decision on some heuristic which path to take. The easiest implementation is to make the decision on the next shortest (lowest ΔV) path. That was the first method used to find a proper sequence and it brought results for J around 2500. The Greedy search has the typical tree drawback that the best cookies are eaten first and the bad ones are left over in the end, and one still has to eat them, cause the entire set has to be collected and not only a subset. And that's something that has been observed when running a greedy search on the tree. The last transfers had a quite high ΔV and that caused a high number of missions and a resulting high J value.

In order to overcome that issue a beam search has been applied. Instead of only travelling along one path, k best options are selected. Depending on the beam width k the computational time of course grows in that case. For this problem an initial beam width of 123 has been selected for the first mission. On the next level the maximum beam width is already $123 \cdot 122 = 15006$ and so on. Limiting the beam width for the first runs to 2000 already gave good results for J around 1000. The entire idea behind that method is, that it is possible to look into the future by taking also some bad paths hopping they turn into golden paths in the end.

At each level each possible solution has been checked for uniqueness. That can also be explained when looking again at Figure 7. The sequence 1-3-2 is equal to 3-1-2, because for the next level the start node (in that case node 2) as well as the left over debris-set is the same. So the algorithm only takes the best sequence out of that two, because the beam width is limited and many different permutations are needed to find the golden path.

The graph algorithm has been implemented in *Matlab* and took roughly 1 hr computation time on a Intel Xeon CPU E3 3.50GHz, with a beam width set to 20000. Running the tree after the first mission the left nodes are getting less and the computation time goes down.

With the 5000 kg propellant a maximum ΔV of 5000 m/s can be achieved. But with the results from the qualitative J-function analysis the maximum has been set to 4500 m/s and also runs with lower values have been performed. For the first mission it was possible to perform 23 transfers. But the beam width at that point was only around 10 to 20. So in that case there are not enough permutations for the next missions. Instead of taking the maximum transfer solution one with a higher beam width has been taken. The algorithm has been set up in a way that the number of transfers is the same for all beams. That may not be the optimal choice and some further investigation may be performed to see if a free number of transfers brings a significant improvement.

The final sequence will be discussed in the Results section 6

5 Transfer Problem

Before solving the debris to debris final transfer the sequence coming out of the beam search algorithm needs to be re-optimized. As already discussed in section 4, the transfer time for all transfers has been kept to a fixed value. That has been re-optimized using the local optimizer *fmincon* in *Matlab*. The cost function in that case is the sum of the ΔVs for all transfers in that particular mission. The design variables are the transfer times. The upper bound has been set to 25 days, constrained by the problem statement. The lower bound was set to 1 day, cause some time for the final phasing of the true anomaly may be needed, which has been ignored completely so far. In the combinatorial part a fixed grid size for the transfer time has been used (e.g. always 17 days

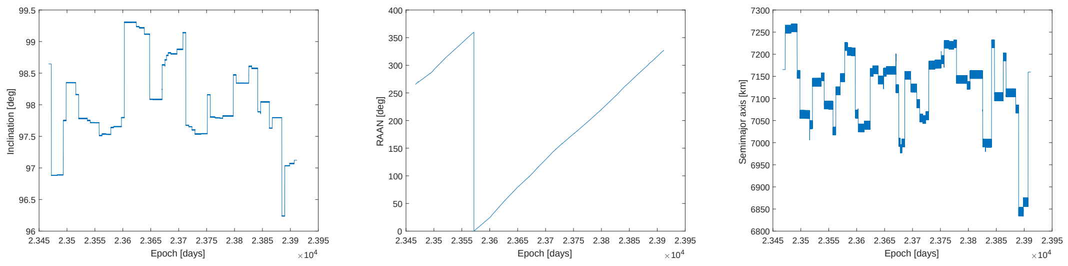


FIGURE 8. *Evolution of orbital elements for mission 1*

for the first 6 missions and than 19 or 20 for the remaining, depending how many debris is left to collect). An inequality constraint had to be introduced that the sum of the transfer times is not larger than the old sum of the fixed transfer times (That means one can only shift some transfer time from one transfer to the other). For the re-optimizer the look up table for the ΔV was not used. Instead it was calculated during the *fmincon* call. That has the advantage of getting a real value for the transfer time. With that approach between 10% and 25% ΔV per mission has been saved.

For each transfer the following information is known:

- departure epoch
- arrival epoch
- transfer time
- estimated ΔV

To solve that problem again *Matlab fmincon* with an interior point method has been used. The control parameters are the times between maneuvers and the thrust of 5 maneuvers itself in cartesian form. The cost function is quite easy in that case, it's just the sum of all 5 ΔVs we applied. There are 3 deep space maneuvers in addition to one at departure and one at arrival. The more demanding part is the constraint function passed to *fmincon*. Here the equations of motion are integrated between the maneuvers until we reach our final state. Than the final state should equal the arrival debris state at that time. There is a global parameter in order to activate or deactivate the constraints, and it is possible to choose between the cartesian state vector, Keplerian elements, or a mixture, or a subset. Another inequality constraint had to be introduced, taking care that the sum off all transfer times between maneuvers is not larger than the transfer time from the tree, otherwise the following transfers are messed up.

When using an ODE-solver in *fmincon*, the integration errors from the ODE solver may disturb the Jacobian or Hessian. That was the case for the first runs. An investigation has been performed which ODE solver brings reasonable results. The conclusion was that a fixed step size is more stable than a variable step size solver. In the end a RK8 has been used, implemented in C++ with a step size of 50 s (the RK4 needed 1 s step size), cause the *Matlab* implementation was to slow. An interesting observation was also that when using the debris dynamic model first and rerunning the optimizer with the spacecraft ODE, faster and better results have been achieved.

For the initial guess the first and last two maneuvers have been set to the Hohmann like maneuvers coming out of the drift strategy. The maneuver in the middle is set to 0. The inclination and phasing change has been solved by the optimizer. The first and last transfer times where set to a half orbital period. And the remaining two transfer times where chopped up equally (kind of mid course maneuver).

One result is that scaling the state vector x, the constraints and the cost function all close to 1 is crucial for success. Although one would assume that this techniques should be handled by optimizers automatically, that seems not to be the case.

With the RK8, the algorithm took roughly 5 minutes on a Intel Xeon CPU E3 3.50GHz. And all transfers converged proper. The achieved ΔV was even lower than the estimated one out of the tree search, cause in the tree search the ΔV for the inclination change and the drift maneuver has been added separately. In practice they can be combined and the optimizer seems to have taken care of that.

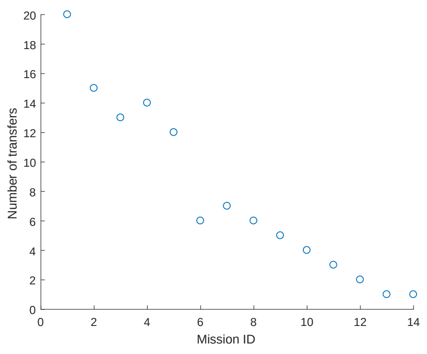


FIGURE 9. *Mission Graph*

6 Results

The solution submitted by the DLR team had a total of $n = 14$ missions and a performance index $J = 949.85$ MEUR. Figure 9 shows the number of transfers per mission. For the first mission the evolution of Ω, a and i are plotted over time (see Figure 8). It can be seen that mainly the inclination and semi major axis was changed by the ΔV and the right ascension just drifts along to the next target.

7 Conclusion

For the combinatorial part genetic algorithms are suitable when the problem is time invariant. But for time variant problems graph algorithms seem to be the better choice. The Beam search algorithm brought reasonable results, but still suffers a bit from the greedy effect: there are good sequences in the beginning but bad ones in the end. One option to improve that may be to select some feasible continuation beams randomly. In the transfer problem one may use a multiple shooting method to get rid off the ODE-integration issue.

References

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

[2] M. Cerf. Multiple Space Debris Collecting Mission - Debris Selection and Trajectory Optimization. *Journal of Optimization Theory and Applications*, 156:761–796, 2013.

[3] Andrew John Drake. *Approaches for solving some scheduling and routing problems*. PhD thesis, University of Southampton, February 2009.

[4] R. C. Engels and J. L. Junkins. The gravity-perturbed Lambert problem: a KS variation of parameters approach. *Celestial Mechanics*, 24:3–21, May 1981.

[5] Hernán Abeledo, Ricardo Fukasawa, Artur Pessoa, and Eduardo Uchoa. The time dependent traveling salesman problem: polyhedra and algorithm. *Mathematical Programming Computation*, 5(1):27–55, March 2013.

[6] D. Izzo, I. Getzner, D. Hennes, and L. F. Simões. Evolving solutions to TSP variants for active space debris removal. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1207–1214, July 2015.

[7] D. Izzo, D. Hennes, L. F. Simões, and M. Märtens. Designing complex interplanetary trajectories for the global trajectory optimization competitions. *Space Engineering*, pages 151–176, 2016.

[8] D. Hennes and D. Izzo. Interplanetary Trajectory Planning with Monte Carlo Tree Search. In *International Joint Conference on Artificial Intelligence*, pages 769–775, 2015.

[9] L. F. Simões, D. Izzo, E. Haasdijk, and A. E. Eiben. Multi-rendezvous Spacecraft Trajectory Optimization with Beam P-ACO. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 141–156, 2017.

Acta Futura 11 (2018) 79-90

DOI: 10.5281/zenodo.1139258

Acta Futura

GTOC9: Results from University of Trento (team ELFMAN)

ENRICO BERTOLAZZI [*], FRANCESCO BIRAL, MATTEO RAGNI

DEPARTMENT OF INDUSTRIAL ENGINEERING, UNIVERSITY OF TRENTO

Abstract. The GTOC9 competition requires the design of a sequence of missions to remove debris from the LEO orbit. A mission is a sequence of transfer of the spacecraft from one debris to another. Both missions and transfer must fulfill a set of constraints. The work presents the procedures to develop a solution for the GTOC9 problem (i.e the mission sequence) that does not violates constraints.

The solution is obtained through an evolutionary algorithm that combines pre-computed basic missions stored in a database. The main objective of the algorithm is to minimize the overall cost of the solution, in order to maximize the competition score.

The database of pre-computed missions is derived by connecting trasfers stored in a database of transfers, through a combinatorial approach that considers the problem constraints.

The database of transfer is formulated through the solution of a constrained minimization problem upon the control action (the magnitude of the overall impulsive velocity changes ΔV). Only a subset of all possible transfers (selected on the basis of *acceptable* ΔV), enters in the database.

1 Introduction

The 9th Global Optimization Competition (GTOC9) requires the design of a sequence of missions—i.e. the *solution*—in order to cumulative clean up the Sun-synchronous Low-Earth-Orbit from 123 debris that may trigger the Kessler effect, while minimizing an overall cost.

A single mission is characterized by a sequence of rendezvous spacecraft trajectories between debris. The spacecraft is controlled with impulsive changes in velocity. For each debris, the spacecraft activates a de-orbit package that removes the debris from the LEO orbit.

Each mission starts from a debris—i.e. it is not necessary to design the launch from the Earth—and continues for an arbitrary number of transfers, limited only by the fuel consumption. Each mission has to comply with some rules:

- the spacecraft has to wait 5 days to activate the de-orbit package;
- the time between two rendezvous must not exceed 30 days;
- a maximum of 5 velocity impulses are allowed;
- the spacecraft may never reach an orbital periapsis lower than $6600\,\mathrm{km}$.

At least 30 days must be accounted between two subsequent missions. The performance index is:

$$J = \sum_{i=1}^{N} \left(c_i + \alpha (m_{0,i} - m_{\mathrm{dry}})^2 \right) \tag{1}$$

[*]Corresponding author. E-mail: enrico.bertolazzi@unitn.it

where c_i is a submission cost that is proportional to submission time (favoring earlier submission), $m_{0,i}$ is the initial mass of the spacecraft at mission i, and m_{dry} is its dry mass (favoring lighter mission). α is a constant scaling factor.

The fuel consumption of a single change in velocity is calculated by the Tsiolkovsky equation:

$$\Delta m = \left(1 - \exp\left(-\frac{\Delta V}{I_{\text{sp}} g_0}\right)\right) m_i \tag{2}$$

During the transfer between two successive debris, the spacecraft trajectory is approximated with a Keplerian motion perturbed by the effect of an oblate Earth—i.e. J_2 factor—and it is modeled by the ODE:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= -\frac{\mu}{r^3}\mathbf{r} + \mathbf{J}(\mathbf{r}), \\ \mathbf{J}(\mathbf{r}) &= \frac{3}{2} J_2 \frac{r_{\text{eq}}^2 \mu}{r^5} \begin{pmatrix} x - 5\,x\,(z/r)^2 \\ y - 5\,y\,(z/r)^2 \\ 3z - 5\,z\,(z/r)^2 \end{pmatrix} \end{aligned} \tag{3}$$

where $\mathbf{r} = \begin{pmatrix} x & y & z \end{pmatrix}^T$ is the position vector, $r = ||\mathbf{r}||$ and $\mathbf{v}$ is the velocity vector. For more details, constants definitions and constraints, refer to problem description [1].

Sec. 2 explores the complex procedures to obtain a solution that fulfill problem statement constraints. The Section is divided in three major parts. Sec. 2.1 explores the formulation of the minimization problem to formulate a transfer between debris, leveraging the dynamical system of the spacecraft. The final result of the section is the *database of transfers*. Sec. 2.2 focuses on the exploration of a database of transfers in order to build chains of transfer that constitute a single mission, through combinatorial based approach, that creates the *database of missions*. Sec. 2.3 concatenates the missions in order to develop a final solution that becomes the actual submission to the authorities.

2 Implemented solution

At the core of the problem there is the transfer of the spacecraft from one debris to another. The transfer implies a loss in mass, due to the control action of the spacecraft ΔV, thus the evaluation of the control action is fundamental for the solution of the problem. The key of the proposed solution lays in the separation between mass loss evaluation and visited debris sequence in a single mission. The mass loss is treated as an independent problem with respect to the sequence.

The mission sequences result from a searching problem in a very large graph. The graph is explored by means of queries to a database of trajectories and *mean mass losses* for transfer from one debris to another.

The solving algorithm is divided in three logical steps:

- the first part identifies all maneuvers between two orbiting debris at a defined departure epoch, with an *acceptable* loss due to ΔV: only a subset of transfer is considered since the whole set of trajectory would be too large to handle, as detailed in Section 2.1. The trajectories are saved in a database for the following solution step.
- the second step connects maneuvers saved in the previous database to obtain a subset of all missions: also in this case there is a pruning policy to reduce the total size of the database that will contain all *acceptable* missions. This step is described in Section 2.2.
- as a third step, the missions are combined together, in such a way the total cost is minimized, as described in Section 2.3

2.1 Database of Transfers (DBT)

A database is filled with the transfer maneuvers that approximate the minimum fuel consumption. Virtually, the database contains an infinite number of transfers, thus, some heuristic criteria are used to group maneuvers into equivalence classes and store only one rendezvous maneuver for each class.

Let's consider d_i as the current debris, while d_j, with $i \neq j$, is the arrival debris. The starting and arrival epochs of each transfer belong to a prescribed domain.

The maneuvers may differ in terms of total transfer time and starting epoch. Heuristically, the longer the traveling time, the lower the fuel required to perform the transfer. To generate trajectories with both short and long traveling time, an optimization problem is performed on several time windows—e.g. time intervals of $[0, 1]$ days, $[0, 5]$ days and $[0, 15]$ days are used.

The resolution for starting epoch is one day. For maneuvers with starting and ending time that differs less than 1 day, only the most proficient ones are stored in the database.

The evaluation of the transfer that minimizes the mass loss is a result of a complex optimization problem that is formulated in order to reduce the computational resources required. The large number of trajectories to be generated, for the whole set of debris combination and starting epoch, makes the approach costly so that a number of speed-up strategy are used.

For a limited amount of time, the debris orbit can be well approximated as a Keplerian orbit. The theoretical minimum ΔV to transfer from one orbit to another can be computed semi-analytically (see [2]) with a very fast procedure based on the computation of the roots of an 8^{th} order real polynomial. Only the maneuvers with a theoretical ΔV below a reasonable threshold are considered for further computations. The details of the approach and its implementations are described in Section 2.1. The minimum ΔV is attained considering only a departing and an arriving point on the Keplerian orbits, disregarding the actual positions of the debris. The spacecraft transfer is approximated with a Keplerian trajectory obtained by a Lambert maneuver.

It is easy to find the starting epoch when the debris d_i is in the right departing position for the Lambert maneuver. The real challenge is the synchronization with the arrival debris d_j. An integer optimization problem is solved in order to minimize the distance between the arrival debris d_j and the spacecraft, which is approaching the target orbit—cfr. Section 2.1. The result of the integer optimization is the number of complete revolution that must be performed by the rocket on the Keplerian transfer orbit, the initial, and final time necessary to reach the debris d_j.

This transfer still approaches only approximatively the target debris d_j, thus, a new Lambert maneuver using initial and final time with initial and final position is computed. This second maneuver, in general, applies ΔV that is not too far from the theoretical minimum ΔV.

The rocket dynamics is not Keplerian, consequently the Lambert transfer is only an approximation of the required trajectory, that is the solution of a *two-burns* optimal variational problem (2B-OVP). The 2B-OVP is described in detail in Section 2.1 where the Lambert transfer is adopted as initial guess.

An high quality solution is evaluated only when a submission to the authorities is required—cfr. Section 2.3.

The trajectories are inserted in a database, where each record contains:

- starting and final epoch
- starting and arrival debris
- intermediate burn event time (if present)
- ΔV initial, final, intermediate (if present)

Those information are enough to reconstruct the transfer trajectory.

Minimum ΔV Transfer

The work of Zhang, Zou and Mortari [2] is the cornerstone for the minimum ΔV estimation. In this work, the authors derive a semi-analytical procedure that computes the travel time t_w which minimize the $\Delta V(t_w)$ required to transfer from one Keplerian orbit to another with initial and final position fixed.

The minimum is computed searching the points where derivatives of $\Delta V(t_w)$ is zeroed. This $\Delta V(t_w)$ is the sum of two norms $||\delta \mathbf{v}_0(t_w)||$ and $||\delta \mathbf{v}_1(t_w)||$ that are not differentiable near zero so that the derivative is computed formally as

$$\begin{aligned} 0 = \frac{\mathrm{d}}{\mathrm{d}t_w}\left(\Delta V(t_w)\right) &= \frac{\mathrm{d}}{\mathrm{d}t_w}\sum_{i=0,1}||\delta \mathbf{v}_i(t_w)|| \\ &= \sum_{i=0,1}\frac{\mathrm{d}\left(||\delta \mathbf{v}_i(t_w)||^2\right)/\mathrm{d}t_w}{||\delta \mathbf{v}_i(t_w)||} \end{aligned} \tag{4}$$

so that the relation

$$\frac{\mathrm{d}\left(||\delta \mathbf{v}_0(t_w)||^2\right)/\mathrm{d}t_w}{||\delta \mathbf{v}_0(t_w)||} = -\frac{\mathrm{d}\left(||\delta \mathbf{v}_1(t_w)||^2\right)/\mathrm{d}t_w}{||\delta \mathbf{v}_1(t_w)||} \tag{5}$$

is squared on both sides and an 8^{th} degree polynomial in t_w is obtained. The positive real roots are the candidates for t_w and the minima are discriminated by a simple procedure (see [2]). The roots are calculated through the fast Jenkins–Traub algorithm [3].

This procedure is extremely fast and is the core of the computation of the optimal debris transfer, and works as follows:

- The two orbits are sampled with e.g. nearly equally spaced points and all the combinations of pairs of starting and arrival points are evaluated for searching minimum ΔV. The minima are refined applying a re-sampling with points near the best candidates. This procedure is repeated a couple of time. At the end of this procedure the initial and final point $\mathbf{r}_0$ and $\mathbf{r}_1$ with the transfer time t_w are set.

- Let t_0 the time at which the initial debris reaches point $\mathbf{r}_0$—i.e. the initial point of Lambert trajectory—and t_1 the time at which the arrival debris intercepts point $\mathbf{r}_1$. Let T the period of the Lambert trajectory and T_1 the period of the target debris Keplerian orbit. A rendezvous satisfies the following equation

$$t_0 + \underbrace{t_w + n\,T}_{\text{travel time}} = \underbrace{t_1 + m\,T_1}_{\text{intercept time}} \tag{6}$$

where n is the number of revolutions the Lambert trajectory should complete, such that the target debris reaches the arrival point, at the same time, after m revolutions. Since there is no perfect match in practice, the previous equation is transformed in:

$$\underset{m,\,n}{\arg\min} \left|(t_0 + t_w + n\,T) - (t_1 + m\,T_1)\right| \tag{7}$$

where m and n belongs to a limited range, such that

$$0 \le \left\{ \begin{array}{c} t_w + n\,T \\ t_1 - t_0 + m\,T_1 \end{array} \right\} \le \Delta t_{\max} \tag{8}$$

where $\Delta t_{\max}$ is the maximum travel time considered.

- Once n is evaluated, the arrival time $t_f = t_0 + t_w + n\,T$ is used to compute the true position of the arrival debris. With this data—i.e. initial time and position, final time and position, and number of revolutions—a new Lambert problem is solved and used as initial guess for the 2B-OVP of Section 2.1.

The Lambert solver is a C++ routine based upon a MATLAB script written by Dario Izzo which implements an efficient and fast algorithm [4, 5, 6]

Equinoctial Coordinates and Integration

The rocket model proposed in (3) is an approximation of a LEO orbit, which contains a perturbation term due to oblate Earth, which makes numerical integration using Cartesian coordinates impractical. In fact, an high order numerical integration scheme is required to keep the prescribed tolerance with a reasonable time step.

To avoid the usage of an highly accurate numerical scheme, the ODE (3) is reformulated in terms of equinoctial coordinates, where the oblate perturbation is modeled as a low thrust action. This permits to integrate through low order numerical methods [7] that maintains the required accuracy. Equinoctial coordinates and the disturbances vector, due to oblate Earth, are:

$$\begin{aligned} \mathbf{y} &= \begin{pmatrix} p & f & g & h & k & L \end{pmatrix}^T \\ \mathbf{\Gamma} &= \begin{pmatrix} \Gamma_r & \Gamma_t & \Gamma_n \end{pmatrix}^T \end{aligned}$$

and equations of motion for the spacecraft can be stated as:

$$\dot{\mathbf{y}} = \mathbf{A}(\mathbf{y})\mathbf{\Gamma}(\mathbf{y}) + \mathbf{b}(\mathbf{y}).$$

The equinoctial dynamic is well known, but for completeness is hereby reported:

$$\begin{aligned} \mathbf{A}(\mathbf{y}) &= \frac{1}{q}\sqrt{\frac{p}{\mu}} \begin{pmatrix} 0 & 2p & 0 \\ q\,s_L & a_{2,2} & -g\,a_{6,3} \\ -q\,c_L & a_{3,2} & f\,a_{6,3} \\ 0 & 0 & \frac{1}{2}s^2 c_L \\ 0 & 0 & \frac{1}{2}s^2 s_L \\ 0 & 0 & a_{6,3} \end{pmatrix} \\ \mathbf{b}(\mathbf{y}) &= q^2 p^{-3/2}\sqrt{\mu}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T \end{aligned}$$

with:

$$\begin{array}{ll} c_L = \cos(L) & a_{2,2} = (q+1)c_L + f \\ s_L = \sin(L) & a_{3,2} = (q+1)s_L + g \\ C_f = c_L + f & a_{6,3} = h s_L - k c_L \\ S_g = s_L + g & q = 1 + f\,c_L + g\,s_L \\ \alpha^2 = h^2 - k^2 & s^2 = 1 + h^2 + k^2 \end{array}$$

The equinoctial coordinates $\mathbf{y}$ are related to the state $(\mathbf{r}, \mathbf{v})$ expressed in Cartesian coordinates according to the following identities:

$$\begin{aligned} \mathbf{r}(\mathbf{y}) &= \frac{p}{qs^2}\begin{pmatrix} c_L\left(1+\alpha^2\right) + 2hk\,s_L \\ s_L\left(1-\alpha^2\right) + 2hk\,c_L \\ 2\,a_{6,3} \end{pmatrix} \\ \mathbf{v}(\mathbf{y}) &= \frac{1}{s^2}\begin{pmatrix} 2hk\,C_f - \left(1+\alpha^2\right)S_g \\ \left(1-\alpha^2\right)C_f - 2hk\,S_g \\ 2(\mu/p)^{1/2}\left(h\,C_f + k\,S_g\right) \end{pmatrix} \end{aligned} \tag{9}$$

The vector $\mathbf{\Gamma}(\mathbf{y})$, in equinoctial reference frame, is the matrix vector product $\mathbf{Q}(\mathbf{y})^T\mathbf{J}(\mathbf{r})$, where $\mathbf{Q}(\mathbf{y})$ is the orthogonal matrix:

$$\mathbf{Q}(\mathbf{y}) = \begin{pmatrix} \frac{\mathbf{r}}{\|\mathbf{r}\|} & \frac{(\mathbf{r}\times\mathbf{v})\times\mathbf{r}}{\|\mathbf{r}\times\mathbf{v}\|\|\mathbf{r}\|} & \frac{\mathbf{r}\times\mathbf{v}}{\|\mathbf{r}\times\mathbf{v}\|} \end{pmatrix} \tag{10}$$

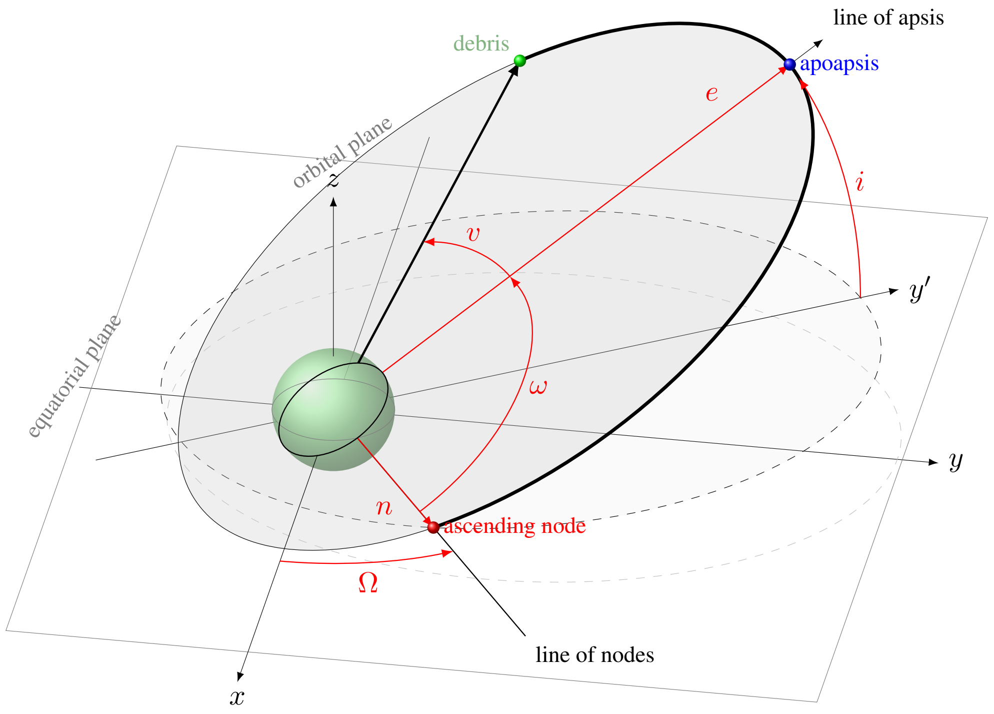


FIGURE 1. *Equinoctial coordinates for Sun-synchronous LEO orbits*

with $\mathbf{r}$ and $\mathbf{v}$ as defined in (9). The use of equinoctial coordinates permits to integrate a trajectory through a second order numerical method with a fixed time step of 6 s, maintaining the required accuracy for a integration domain of 30 day.

The forward integration is the classical second order Runge-Kutta (Heun scheme), while the discretization for the OVP is based upon Crank-Nicolson scheme [8, 9].

Optimal Two-Burn Variational Problem

Lets formulate the problem with cartesian coordinates for the sake of clarity.

Minimize:

$$\Delta V = ||\mathbf{v}(\mathbf{y}(t_0)) - \mathbf{v}_0|| + ||\mathbf{v}(\mathbf{y}(t_f)) - \mathbf{v}_1|| \quad (11)$$

subject to

$$\dot{\mathbf{y}} = \mathbf{A}(\mathbf{y})\mathbf{\Gamma} + \mathbf{b}(\mathbf{y}), \quad (12)$$

$$\mathbf{r}(\mathbf{y}(t_0)) = \mathbf{r}_0, \qquad \mathbf{r}(\mathbf{y}(t_f)) = \mathbf{r}_1, \quad (13)$$

$$||\mathbf{r}(\mathbf{y}(t))|| \geq r_{p_m}, \qquad t \in [t_0, t_f] \quad (14)$$

notice that in cartesian coordinates $\Delta V = ||\delta\mathbf{v}_0|| + ||\delta\mathbf{v}_1||$ and $\mathbf{r}_0$ and $\mathbf{r}_1$ are the initial and final point of the trajectory transfer defined in Section 2.1. Moreover, $\mathbf{r}(\mathbf{y})$ and $\mathbf{v}(\mathbf{y})$ are given in (9) and the value of r_{p_m} is given in [1].

The problem is solved by the custom made PINS solver used in other contexts [10, 11, 12, 13], an indirect problem solver for optimal control problem. PINS is able to solve OCP in the form:

Minimize:

$$\Phi(\mathbf{x}(a), \mathbf{x}(b)) + \int_a^b J(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t)\mathrm{d}t \quad (15)$$

subject to

$$\mathbf{M}(\mathbf{x}(t), \mathbf{p}, t)\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t), \quad (16)$$

$$\mathbf{b}(\mathbf{x}(a), \mathbf{x}(b), \mathbf{p}) = \mathbf{0}, \quad (17)$$

where $\mathbf{u}$ is the control action, $\mathbf{p}$ is a parameter vector, and $\mathbf{M}(\mathbf{x}, \mathbf{p}, t)$ is a nonsingular mass matrix. Problem (11)–(14) fits PINS formulation with $\mathbf{M}(\mathbf{x}, \mathbf{p}, t) =$

$\mathbf{I}$ and empty $\mathbf{u}$ and $\mathbf{p}$. Bound (14) is well approximated in $J(\mathbf{x},\mathbf{u},\mathbf{p},t) \equiv J(\mathbf{x},t)$ using a barrier function.

PINS transforms the variational problem (15)–(17) in a two point boundary value problem that is solved as a non-linear system with a Newton-like iterative method. For the solution of the OVP problem (11)–(14) the initial guess required for the Newton-like method is based upon the Lambert trajectory, built in Section 2.1. The problem has a quite coarse time discretization (600 s), that brings to a very low precision solutions for the trajectory. With this mesh the computational time is very short, and the trajectory is roughly approximated, but still the estimate of the ΔV is good. The computational mean time for problem (11)–(14) on a MacBookPro with 2.9GHz Intel Core i7 is less than one second.

If a specific trajectory is selected as candidate for the final submission, is then re-calculated upon a mesh with a finer time discretization (6 s) that fulfils the tolerance requirements. This new problem uses as initial guess the solution found using the coarser mesh and the mean time on the same hardware is less than ten seconds.

Considering the solution of the integer problem, if the number of estimated revolution is too high—i.e. more than 100 revolutions—computing the solution of 2B-OVP is too costly. To reduce the computational effort, a first part of the trajectory, after the first burn—i.e. the one estimated through the Lambert problem—is integrated forward (see Section 2.1), and only the very few last revolutions of the trajectory—i.e. almost 10—are evaluated through the 2B-OVP, making the whole maneuver a three burns transfer (3B-OVP).

2.2 Database of Missions (DBM)

The second database contains sequences of maneuvres that form a mission. The exploration of database of transfers (DBT), to build the database of missions (DBM), considers the limitations proposed in the problem statement. The sequences are also limited implicitly by the available fuel mass, that is used to generate the pulses. Each time a mission is completed, the sequence is inserted into the mission database.

Even in this case, the number of mission that may be generated is huge, thus some pruning policy must be adopted.

The average cost

In the selection of the better candidate missions the average removal cost is taken into account. This cost is defined as the cost of the mission divided by the number of debris removed:

$$c_{\text{ave}} = \frac{c_b + \alpha \Delta m^2}{n_d} \tag{18}$$

where c_b is the time dependent base cost of the mission and Δm is the fuel consumed in the mission. For simplicity c_b is set to the maximum [1].

The average cost is a projection of the overall cost of the debris removal, and allows to forecast the performances of the solution proposed.

The mean cost has a trend similar to the one depicted in Fig. 2. When the number of debris removed is increasing, the mean cost tends to decrease. This trend is inverted after a certain number of debris removed, when the additional mass Δm required for the mission inverts the cost trend.

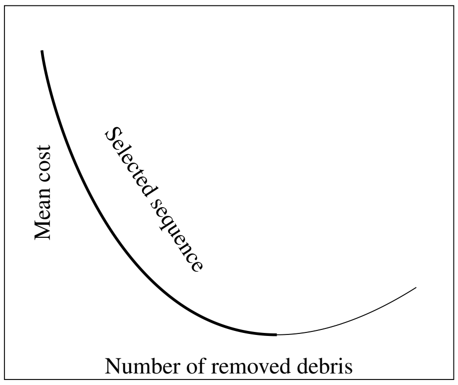


FIGURE 2. *Given a sequence of debris removal the average cost (18) decreases initially, reach a minimum, and then increases. The minimum is an indicator of the optimal number of debris to be removed and of the final cost of the solution The mean cost also permits to compare the solution with the performance of the other competitors.*

The mean cost depends on the length of the mission and it is re-evaluated every time a new transfer is added to the sequence.

Database creation policy

To generate the DBM, the total possible time for performing the debris removal is take into account and

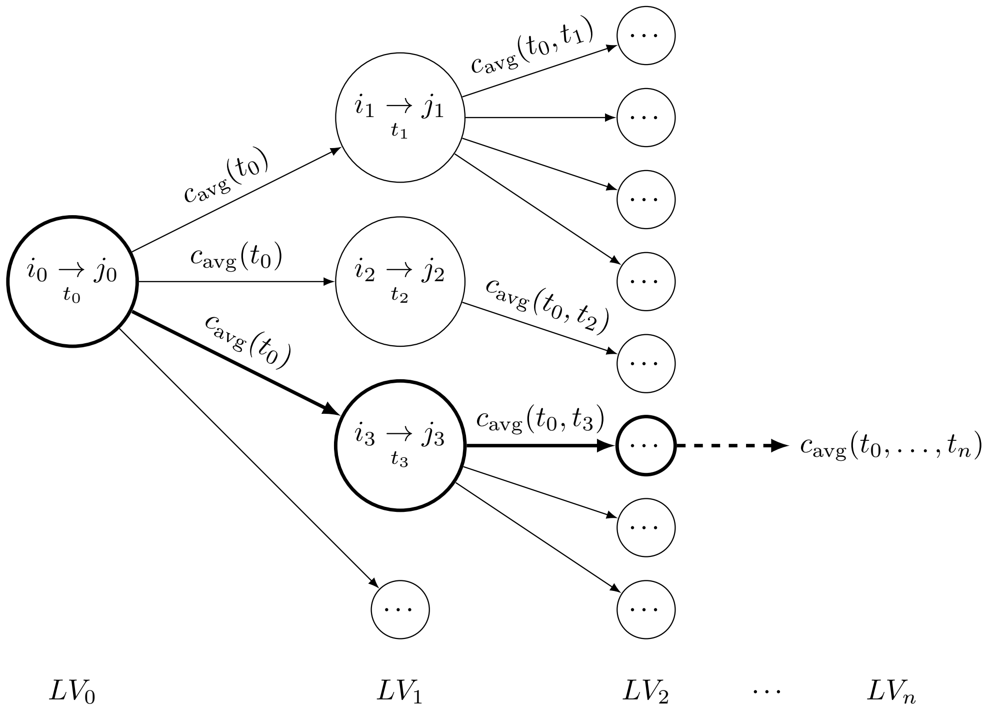


FIGURE 3. *A representation of the graph for the transfer combination*

sliced in time windows $[t_b, t_e]$ where:

$$\begin{aligned} t_b &\in \{0, 100, \dots, 2900\} \\ t_e - t_b &\in \{100, 150, \dots, 400\} \end{aligned} \tag{19}$$

Time windows may overlap. For each time window the N best missions, in term of average cost, are evaluated and stored in DBM.

The construction of missions to be inserted in database is performed in two steps:

- seeds initialization: a seed is the first transfer between two debris in a mission, that determines the starting debris;
- sequences exploration: starting from the seeds, continues the sequences of removal until the maximum fuel consumption limit is satisfied.

The seeds are initialized as follow:

1. from the manouver database (DBM), all the transfers with starting time in a limited initial time frame of the window are selected;
2. for each transfer from debris i to debris j of the previous selection, only the one with minimal average cost is kept;
3. the remaining selection is sorted with respect to the average cost, and only the first N are used as starting seed for the sequence search.

Starting at level 0—i.e. the seed—the exploration continues and at each new level the average cost of the mission is computed. Then, for each level pruning is applied:

- topologically equivalent missions—i.e. with the same set of debris removed and the same arrival debris—are pruned, and only the one with lowest average cost is kept;
- the remaining missions are furthemore pruned and only the N best mission are kept.

Then the exploration continues with the next level. At each level the best N missions are stored in the database.

Each record of the DBM is stored as a sequence of pointers, where each pointer points at a manouveur that is stored in the DBT.

2.3 Missions combination

A series of mathematical objects are defined to formalize the GTOC problem as a minimization problem.

Definition 1 *The set $\mathcal{T}$ is the collection of all possible time intervals for the GTOC problem:*

$$\mathcal{T} := \{[t_0, t_1] \subset \mathbb{R} \,:\, 0 \leq t_0 < t_1 \leq t_{\max}\} \cup \{\emptyset\} \quad (20)$$

Definition 2 *Define with $\mathcal{M}$ the following set:*

$$\mathcal{M} := \left\{[\mathbf{w}, \tau, c] \in \{0,1\}^{123} \times \mathcal{T} \times \mathbb{R}\right\} \quad (21)$$

it represents a minimal encoding of all possible missions. The vector $\mathbf{w}$ marks all visited debris during the mission, the interval τ represents the initial and final time of the mission, which also considers the dead time for the deorbit package release, and c represents the overall cost of the mission.

The *null mission*, i.e. the mission that does not remove any debris is encoded with $m_\emptyset = [\{0\}^{123}, \emptyset, 0]$, while $\widetilde{\mathcal{M}}$ denotes a subset of $\mathcal{M}$ that contains all the possible queries of the *computed* DBM. The set $\overline{\mathcal{M}}$ denotes a subset of $\mathcal{M}$ of all the missions that fulfill the GTOC problem requirements.

Definition 3 *The operators W, T and C*

$$\begin{array}{rccc} W: & \mathcal{M} & \to & \{0,1\}^{123} \\ T: & \mathcal{M} & \to & \mathcal{T} \\ C: & \mathcal{M} & \to & \mathbb{R} \end{array} \quad (22)$$

are defined for $m = [\boldsymbol{\omega}, \eta, \xi] \in \mathcal{M}$ and returns

$$W(m) = \boldsymbol{\omega}, \quad T(m) = \eta, \quad C(m) = \xi. \quad (23)$$

The set $\mathcal{M}^{123}$ contains all the possible sequence of missions for the GTOC problem. For example a sequence of only 5 removal missions can be fit in $\mathcal{M}^{123}$ by appending 118 *null mission $m_\emptyset$*.

Definition 4 *Let $s = [m_1, \ldots, m_{123}]$ a sequence of mission in $\mathcal{M}^{123}$ the operator*

$$Z \,:\, \mathcal{M}^{123} \to \mathbb{N} \quad (24)$$

is the number of debris not removed, i.e.

$$Z(s) = 123 - \mathbf{u} \cdot \mathbf{u}, \quad \mathbf{u} = \bigvee_{i=1}^{123} W(m_i)$$

*where the $\vee$ operator is the **element-wise or**.*

Definition 5 *The overall cost of a (possibly unfeasible) set of missions is:*

$$C \,:\, \mathcal{M}^{123} \to \mathbb{R} \quad (25)$$

which is the sum of the cost of each mission plus the cost of the n_r debris not removed:

$$C(s) = \sum_{i=1}^{123} C(m_i) + Z(s) c_b, \quad (26)$$

for $s = [m_1, \ldots, m_{123}]$ and c_b the maximum submission cost (cfr. equation (18)).

The set $\mathcal{M}^{123}$ permits to define the set of admissible solutions:

Definition 6 *The set $\mathcal{S}(\mathcal{N})$ is:*

$$\mathcal{S}(\mathcal{N}) := \left\{ \begin{array}{l} [m_1, \ldots, m_{123}] \in \mathcal{N}^{123} \,: \\ \textit{for } i \neq j \\ W(m_i) \wedge W(m_j) = \{0\}^{123} \\ T(m_i) \cap T(m_j) = \emptyset \end{array} \right\} \quad (27)$$

*where the $\wedge$ operator is the **element-wise and**, and the set $\mathcal{N} \subset \mathcal{M}$.*

Remark 1 *Notice that the set $\mathcal{S}(\mathcal{N})$ can be extracted from $\mathcal{N}$ easily so that the set is never explicitly constructed.*

With the previously defined mathematical objects, it is easy to state the minimization problem at the core of GTOC.

$$\text{Find } s \in \mathcal{S}(\overline{\mathcal{M}}) \text{ which minimize } C(s). \quad (28)$$

Remark 2 *The problem (28) is a mixed integer minimization problem. Thus, gradient based minimization cannot be used. Moreover, the set $\mathcal{S}(\overline{\mathcal{M}})$ is uncountable. Thus, combinatorial based minimization cannot be used. In this form, the problem (28) is not numerically computable.*

The problem (28) becomes numerically tractable when the set $\mathcal{S}(\overline{\mathcal{M}})$ is reduced to the subset $\mathcal{S}(\widetilde{\mathcal{M}})$ which has finite cardinality:

$$\boxed{\text{Find } s \in \mathcal{S}(\widetilde{\mathcal{M}}) \text{ which minimize } C(s).} \tag{29}$$

Remark 3 *The problem* (29) *is a integer minimization problem defined on a finite cardinality discrete set* $\mathcal{S}(\widetilde{\mathcal{M}})$ *that rules out the use of algorithms based upon gradient or other analytical tools. However, a combinatorial approach is impractical due to the large cardinality, thus evolutionary approach must be preferred.*

The bigger the set $\widetilde{\mathcal{M}}$, the higher the chances that $\mathcal{S}(\widetilde{\mathcal{M}})$ contains a good minimum. Nevertheless, since evolutionary methods have extremely slow convergence rates, a bigger cardinality of such set reduce the chances to find a good approximation of this minimum in the time frame of the GTOC competition.

The pruning policy of section 2.2 is essential to reduce the density of set $\widetilde{\mathcal{M}}$ eliminating solutions in $\mathcal{S}(\widetilde{\mathcal{M}})$ that expose nearly the same structure, retaining only the best one.

2.4 Evolutionary minimization

The mimization of (29) is done through an evolutionary algorithm [14, 15, 16] that is a search technique based on the principles of evolution and natural selection. Schematically, an evolutionary algorithm requires a population of agents—i.e. a solution candidate—with a genomics that encodes a possible solution. An agent is an element of the set $\mathcal{P}$

$$\mathcal{P} := \left\{ \begin{array}{l} [m_1, \ldots, m_{123}] \in \widetilde{\mathcal{M}}^{123} : \\ T(m_i) \cap T(m_j) = \emptyset \;\; \text{for } i \neq j \end{array} \right\} \tag{30}$$

The set $\mathcal{P}$ is larger of the set $\mathcal{S}(\widetilde{\mathcal{M}})$ and an element $s \in \mathcal{P}$ is also an element of $\mathcal{S}(\widetilde{\mathcal{M}})$ it is a solution of the GTOC problem. A population of agents:

$$\boldsymbol{P} = (s_1, \ldots, s_q), \qquad s_i \in \mathcal{P} \tag{31}$$

is evolved to become elements of $\mathcal{S}(\widetilde{\mathcal{M}})$. The *fitness function* of one agent is the the overall cost (26). It is also useful to introduce a function $G : \mathcal{P} \to \mathbb{N}$ that measures the *gap* of an element in $\mathcal{P}$ from the set $\mathcal{S}(\widetilde{\mathcal{M}})$. The *gap* is the number of multiple removed debris:

$$G(s) = \#\{k \,:\, v_k > 1\} \tag{32}$$

where v_k is the number of times the kth debris is removed and is computed as $\mathbf{v} = \sum_{m \in s} W(m)$. The evolutionary algorithm guides the genomic mutation of the population to minimize the fitness and to nullify the *gap* by cyclic on:

1. augment the population by adding random agents;
2. augment the population by cloning with mutation;
3. remove the worst agents with respect to gap;
4. select the next generation with respect to fitness.

In step 1 the population is increased by 10% by selecting randomly from $\mathcal{S}(\widetilde{\mathcal{M}})$. In step 2 the population is increased by 50% through cloning of randomly chosen agents. The cloned agents are then muted accordingly to the rules described in section 2.4. In step 3 the population has more than q agents and it is clustered and sorted in classes with increasing gap. The next population is obtained by the union of the first k classes such that the number of agents in the union is greater or equal to q and minimized—cfr. Figure 4. In this way the evolution rewards a population with smaller gap.

In step 4 the population is sorted by fitness and the first q agents are selected for the next generation.

Mutation rules

Given the agent s the mutation rules change an element m_k of s where k is chosen randomly. The mutated agent is denoted with s' and the mutated mission with m'_k. Let be g the minimum value of $G(s')$ for all the possible mutations $m'_k \in \widetilde{\mathcal{M}}$, then, the set $\mathcal{G}$ contains all $m'_k \in \widetilde{\mathcal{M}}$ such that the gap $G(s') = g$. If the set $\mathcal{G}$ is too small it is enlarged considering elements with such that the gap $G(s') = g + 1$ or more.

M1 Within the set $\mathcal{G}$ select the element that minimizes the cost $C(s')$. It is extremely unlikely to find more than one minima, but in case of duplicates the first found is chosen.

M2 Within the set $\mathcal{G}$ select m'_k randomly with uniform probability.

M3 Within the set $\mathcal{G}$ select m'_k randomly from a conditional probability with respect to the binary vector of the mission. In particular, an higher probability is associated with missions that cross debris that are rarely removed. The rarity of a debris is easily deduced from the database $\widetilde{\mathcal{M}}$.

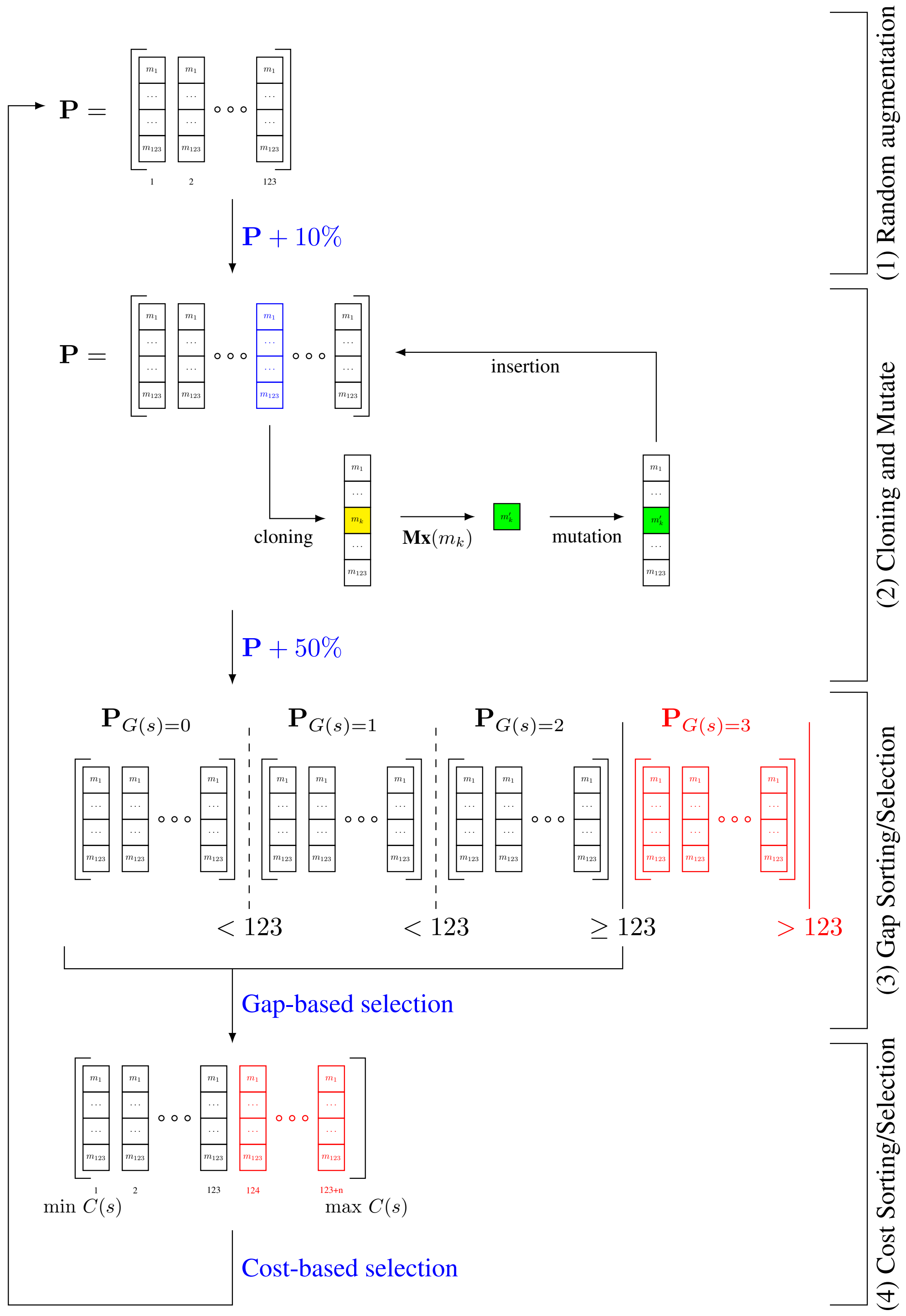


FIGURE 4. *Algorithm main loop*

Only one of the mutations **M1**–**M2**–**M3** is applied, and it is randomly selected.

3 Conclusions

The work presented a strategy to solve the GTOC9 problem, that guarantees a complete compliance with all the constrained provided in the problem statement, and it is a cleaner formulation of the actual strategy used during the competition by the authors. In particular, evolutionary minimization was introduced when it became clear that a simple combinatorial approach was unpractical.

The first step was the construction of a database of transfers, from one debris to another, without considering the removal, upon the whole window allowed for missions in the problem statement. This database is based upon a ΔV minimization. The trajectories in the database where used to build a second database, the database of missions, through combinatorial algorithms. The final step was to chain the different missions in a single solution to be submitted for evaluation. This final solution was identified by a evolutionary approach.

Due to the limited time span for the competition, the very last component of the puzzle, the evolutionary minimization, was not completely developed. Thus an incomplete version of the algorithm was used to explore the solution of (28). Moreover, very few transfers between debris were added to the solution provided by the genetic minimization, using other forms of heuristic approach.

Whit this strategies the ELFMAN team ranked 13th removing 119 debris with a final score of 1107.69367526485.

References

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

[2] Gang Zhang, Di Zhou, and Daniele Mortari. Optimal two-impulse rendezvous using constrained multiple-revolution Lambert solutions. *Celestial Mechanics and Dynamical Astronomy*, 110(4):305, 2011.

[3] M. A. Jenkins. Algorithm 493: Zeros of a Real Polynomial [C2]. *ACM Trans. Math. Softw.*, 1(2):178–189, June 1975.

[4] E.R. Lancaster, R.C. Blanchard, United States. National Aeronautics, Space Administration, and Goddard Space Flight Center. *A unified form of Lambert's theorem*. Technical note. National Aeronautics and Space Administration, 1969.

[5] R. H. Gooding. A procedure for the solution of Lambert's orbital boundary-value problem. *Celestial Mechanics and Dynamical Astronomy*, 48:145–165, June 1990.

[6] Dario Izzo. Revisiting Lambert's problem. *Celestial Mechanics and Dynamical Astronomy*, 121(1):1–15, 2015.

[7] Francesco Casella and Marco Lovera. High-accuracy simulation of orbital dynamics: An object-oriented approach. *Simulation Modelling Practice and Theory*, 16(8):1040–1054, 2008. EUROSIM 2007.

[8] J. C. Butcher. *Runge–Kutta Methods*. John Wiley & Sons, Ltd, 2008.

[9] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6(1):207–226, Dec 1996.

[10] F. Biral, E. Bertolazzi, and P. Bosetti. Notes on numerical methods for solving optimal control problems. *IEEJ Journal of Industry Applications*, 5(2):154–166, 2016.

[11] F. Biral, E. Bertolazzi, and M. Da Lio. *Modelling, Simulation and Control of Two-Wheeled Vehicles*, chapter The Optimal Manoeuvre, pages 119–154. Wiley Blackwell, 2014.

[12] P. Bosetti and E. Bertolazzi. Feed-rate and trajectory optimization for CNC machine tools. *Robotics and Computer-Integrated Manufacturing*, 30(6):667–677, 2014.

[13] P. Bosetti and M. Ragni. Milling part program preprocessing for jerk-limited, minimum-time tool paths based on optimal control theory. *IEEJ Journal of Industry Applications*, 5(2):53–60, 2016.

[14] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.

[15] Randy L. Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition, 2004.

[16] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

Acta Futura 11 (2018) 91-97

DOI: 10.5281/zenodo.1139268

Acta Futura

GTOC 9: Results from the University of Colorado at Boulder (team CU Boulder)

NATHAN L. PARRISH[*], DANIEL J. SCHEERES, SIMON TARDIVEL, CHANDRAKANTH VENIGALLA, JONATHAN AZIZ, MARIELLE PELLEGRINO, OSCAR FUENTES, STIJN DE SMET

UNIVERSITY OF COLORADO, COLORADO CENTER FOR ASTRODYNAMICS RESEARCH
ECNT 320, 431 UCB, BOULDER, CO 80309-0431

Abstract. This paper describes the strategies and results of the GTOC9 competition for the team from the University of Colorado, Boulder. The goal of the competition was to remove 123 pieces of space debris for the lowest cost, with cost defined in Euros as a function of fuel mass, number of launches, and time into the competition window. The overall strategy for this team was: 1) Find the set of all possible low-cost chains of debris to visit, 2) Pick from those to define a series of missions that visit most of the debris, 3) Stitch the remaining 20-30 debris onto the existing chains, 4) Add 1-4 more launches to reach the 5-10 debris that remain after stitching, 5) Adjust the dates of each mission slightly to minimize ΔV, and 6) Find a series of four maneuvers to transfer from each debris to the next in the fully-integrated dynamics. The final solution removed all 123 pieces of debris with 17 launches for a cost of 1150.8 MEUR.

1 Introduction

The GTOC9 competition defines a set of 123 pieces of debris that are in approximately sun-synchronous orbits, with inclination near 98°and semimajor axis near 7,000 km. The motion of spacecraft is defined by numerically integrating Earth point mass gravity perturbed by the J_2 effect. The dynamics of debris are the analytical approximation of J_2 dynamics, with the state at any time given by analytically propagating mean motion, and constant secular drift rates of Right Ascension of the Ascending Node (RAAN) and argument of perigee.

We find that the secular drift of the RAAN due to Earth's J_2 is a primary driver for solutions. The node drift rate is given by

$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{r_{eq}}{p}\right)^2 n \cos i \qquad (1)$$

with semilatus rectum $p = a(1-e^2)$ and mean motion $n = \sqrt{\frac{\mu}{a^3}}$. Since the eccentricity of all the debris pieces is nearly zero, the node drift rate $\dot{\Omega}$ is mostly driven by inclination and semimajor axis. For initial searches to find chains of debris that can be efficiently visited by a single spacecraft, we choose to ignore the drift of argument of perigee. The small eccentricity means that the cost of changing argument of perigee with fuel is very small compared to the cost of changing RAAN.

The goal of the competition is to remove all of the debris for the minimum cost, where cost in MEUR is

[*]Corresponding author, napa0706@colorado.edu

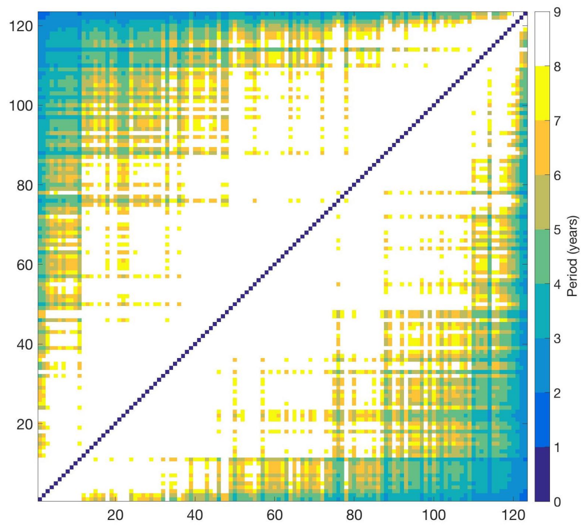


FIGURE 1. *The synodic periods (in years) of the RAAN of each debris object relative to all other debris objects. The debris are sorted by inclination from low to high. Debris that differ significantly in inclination have opportunities for transfers as frequently as every 1-3 years. Debris with very similar inclinations have synodic periods much longer than the 8 year time limit for the competition.*

given by the sum of each launch:

$$J = \sum_{i=1}^{N} \left[c_i + 2 \times 10^{-6}(m_{0_i} - m_{dry})^2\right]. \quad (2)$$

The launch vehicle cost c_i grew linearly from 45 to 55 MEUR over the contest month. The term $(m_{0_i} - m_{dry})^2$ favors lower ΔV per mission. More details on the competition rules are given in [1].

We found that the greatest constraint on the search space is on time of flight between debris, as this limits the feasible transfers to debris with similar RAAN. Figure 1 shows the synodic periods of the RAAN of all the debris. Most pairs of debris have long synodic periods relative to the 8 year time limit of the competition, so transfers are largely limited to rare natural opportunities.

2 Analytical ΔV

Our first approach to finding low cost transfers between debris objects was to look at the natural crossings of debris RAAN. The RAAN drift rate $\dot{\Omega}$ varies per debris object due to variations in semimajor axis and inclination. The differential drift rate between two debris objects causes natural node crossings, which are opportunities for low ΔV transfers between the two objects at a particular time. The initial goal in exploring the problem was to solely use these natural node crossings to build all missions. However, it soon became clear that this approach was too limiting – there are not enough natural node crossings to solve the problem in a small number of launches, again due to time constraints for the competition. To find more options for cheap transfers, we developed analytical approximations for the cost to transfer between any two debris, then surveyed the available transfers. Transfer ΔVs were approximated by computing three terms: the cost of matching RAAN, the cost of matching inclination and semimajor axis, and the cost of matching the orbit phase (defined as argument of latitude).

The change in RAAN rate $\Delta\dot{\Omega}$ required to force the nodes to cross was calculated by setting a fixed transfer time (in most cases, 20 days) and calculating the difference in RAAN between the two debris objects at the fixed rendezvous time. The ΔV required to achieve the desired $\Delta\dot{\Omega}$ was approximated by assuming circular orbits and small changes in either inclination or semimajor axis. Under these assumptions, we get the following expressions for change of the node rate corresponding to a change in inclination ($\Delta_i\dot{\Omega}$) and to a change in semimajor axis ($\Delta_a\dot{\Omega}$).

$$\begin{aligned} \Delta_i\dot{\Omega} &= \frac{\partial\dot{\Omega}}{\partial i}\Delta i && (3)\\ &= -\dot{\Omega}\tan i \Delta i && (4)\\ \Delta_a\dot{\Omega} &= \frac{\partial\dot{\Omega}}{\partial a}\Delta a && (5)\\ &= -\frac{7}{2}\dot{\Omega}\frac{\Delta a}{a} && (6) \end{aligned}$$

We can then write a simple relationship between orbit element change and its corresponding ΔV as follows:

$$\begin{aligned} \Delta i &= \sqrt{\frac{a}{\mu}}\Delta V && (7)\\ \Delta a &= 2a\sqrt{\frac{a}{\mu}}\Delta V && (8) \end{aligned}$$

Combining these last equations gives us final relationships between a desired $\Delta\dot{\Omega}$ and its corresponding ΔV.

$$\Delta V_i = \Delta_i\dot{\Omega}\frac{2}{3J_2R^2}\frac{a^3}{\sin i} \tag{9}$$

$$\Delta V_a = \Delta_a\dot{\Omega}\frac{2}{21J_2R^2}\frac{a^3}{\cos i} \tag{10}$$

Interestingly, comparing the efficiency of changing $\Delta\dot{\Omega}$ with an inclination change vs with a semimajor axis change by taking the ratio $r_{i/a} = \Delta_i\dot{\Omega}/\Delta_a\dot{\Omega}$ shows that they are equally efficient at $\tan i = -7$, or $i = 98.13°$. For inclinations greater than $98.13°$, a semimajor axis change is more efficient, while for inclinations less than $98.13°$, an inclination change is more efficient. One important caveat here is that for more aggressive maneuvers such as those used in "stitching" (see section 4), the assumptions made here break down. In these cases, a less efficient (but more accurate) approximation was used.

For large maneuvers, ΔV was instead approximated by solving Eqn 1 for either semimajor axis or inclination. The ΔV is then computed as the minimum required to change either semimajor axis or inclination. We did not consider cases where both semimajor axis and inclination were changed. If semimajor axis is used to change RAAN, the ΔV comes from the vis-viva equation. If inclination is used, then the maneuver cost is given by $\Delta V_i \approx 2V\sin(\frac{\Delta i}{2})$. Both ΔVs were calculated, and the smaller of the two was selected as the optimal transfer.

The cost to match inclination and semimajor axis of the target debris was approximated with a Hohmann transfer. The inclination change maneuver is combined with one of the semimajor axis maneuvers, and is chosen to occur at the radius of apogee of the orbit with the larger semimajor axis.

The final term of the ΔV approximation is the cost of matching the target debris' argument of latitude. This term was considered separate from the others and was approximated assuming circular orbits and ignoring the precession of the argument of perigee. Two phasing maneuvers are performed: one at the very start of the transfer time, and one at the very end. This approximation is the least accurate of the three terms, but it is also the smallest component. Adding this term effectively penalized shorter transfers, and it brought the total approximate cost closer to the truth.

Overall, we found that the approximate ΔV was accurate to within $\pm 30\%$ of the final integrated transfer.

3 Building Blocks

With the analytical estimate for ΔV described above, we then pre-computed all the possible chains of debris with an efficient algorithm, subject to the following criteria: Transfer time is exactly 20 days, the ΔV for each debris-to-debris transfer is $\leq 500m/s$, and the average ΔV of all the transfers in a chain is $\leq 200m/s$. This resulted in approximately 500,000 chains of debris with between 5-10 debris per chain.

Every chain was sorted by the rarity of the debris it contains, so that the debris that appear least frequently in all of the pre-computed chains are more likely to be selected early in the algorithm, while the debris that appear most frequently are left to the end. This was guided by the intuition that frequently-appearing debris will be easier to stitch on to existing chains later.

A randomized greedy search was then used to combine pre-computed chains together to find campaigns of missions that each visit unique debris at unique times. By randomizing the search, many possible campaigns of missions were generated, and the most promising were passed on to the next step.

After seeing the success other teams had with various genetic algorithms for this step, we recognize that the greatest improvement in score could be made by choosing a better set of initial chains.

4 The Stitcher

The building blocks algorithm was run several thousand times, and it would typically find 10-15 sets of chains that visit 75-95 of the 123 debris. The remaining debris were then left to the "stitcher" to attach to these chains, one by one.

4.1 The stitcher toolset

The stitcher toolset consisted of many routines, from computation of a single stitching action at the lowest level, to updating an entire campaign at its highest level. At this level, it would take as input a campaign, defined as a set of N missions visiting K debris. The algorithm would then proceed to attach the remaining debris to any of the missions. The stitcher would end with two exit conditions: if all debris were successfully attached, or if it was impossible to stitch some last debris to the already existing missions.

We will describe two important aspects of the stitcher: first how we optimized the low-level stitching of a single debris attached to a single mission, then how the high-level algorithms handled these possible stitchings into a sorting tree to output the best result that we could find.

4.2 Optimizing the stitching of a single debris to a single mission

At the lowest level, one of the stitcher routines consisted in optimizing the stitching of single given debris to a single given mission. The goal was to minimize the total ΔV for the mission.

Two options were possible. The first one, simple fit, consisted in keeping the original sequence of debris as-is, and simply finding where the debris would fit best. The initial order of the mission was conserved, and the additional debris simply inserted between any two other debris. The complexity of this algorithm was linear of the number of debris already attached to the chain, and made for a very fast computation (usually on the order of a few milliseconds).

The second option was much more powerful but required factorially more time. It would fit the debris anywhere in the chain, but also reorder the chain. Because of the complexity of the constraints between debris, we adopted a brute force method for testing the reordering of the chain: in practice, all possible arrangements of the debris were tested. The second option would thus call the "simple fit" function $K!$ times for a mission visiting K debris. The algorithmic complexity was then quite punitive for long missions. In effect, this option was instantaneous for missions shorter than 6 debris, and would have taken more than a year for missions longer than 13 debris. For this reason, this option was never used until the very end of the competition: if the numerical integrator failed to realize a planned mission, this problematic mission was given back to the stitcher. We would remove a debris and try to stitch it back, with instructions to reorder the mission. The result would generally lower the required ΔV by a few hundreds of meters per second and allow the numerical integrator to make it into a real mission.

4.3 Finding the best stitching

The hard part of the stitching operation was to decide what to stitch where. At this higher level, the algorithm had a campaign of N missions, visiting K debris. Usually N was between 10 and 15 while the K would range between 75 and 95. Among the remaining 30-50 debris, which one would be best to stitch where and when?

Over the last two weeks of GTOC, several versions of the algorithm were created, each attempting to respond to the increasing leaderboard competition. The final version used involved a tree search with partial randomization. Instead of looking at a single campaign, we would create alternate scenarios, depending on which debris was stitched to which mission. The algorithm would be manipulating a number of scenarios (10-20) at any given step. From each of these, it would create many more (20-50) for the next generation by trying out tens of different stitchings to each of the manipulated scenarios. Finally, it would select which scenarios to keep among the best ones with an element of randomness. The best scenarios were determined as the ones with the lowest added ΔV (sometimes this ΔV would even be negative), as it usually output the longest chains and lowest numbers of debris left after the algorithm had run. Finally, at each step, the algorithm would check that all considered scenarios were indeed different from each other, as it was common for multiple scenarios to arrive at the same "best" solutions.

Since there were many scenarios coexisting at a given step in the search, we needed to quicken our computation of the stitching of each debris to each mission. To do so, we would create a stitching matrix that would follow a scenario and its children if they got selected, meaning that many scenarios could be explored at the same time for very little added computation time. This matrix had 123 rows and N columns, where N was the number of missions (from 12 to 15 usually). Each cell (k,n) of this matrix contained the information on the stitching of debris k to mission n. A stitching (cell) was recomputed if and only if a modification to the other missions, through a previous stitching, affected its feasibility.

Although the algorithm certainly dismissed many good solutions too early, it still allowed reaching unexpected solutions that would prove beneficial in the long term. It was however our impression that this exploration was only as good as the criterion used to rank the scenarios: "lowest added ΔV" was a good enough measure of optimality initially but it appeared quite clearly that it was too greedy an approach to capture the best solutions available. We were however unable to find a better measure of optimality. It is likely that, given a subset of each mission of the winning solution and only 40 debris left to place, this algorithm would still have missed the full winning solution unless massive amounts of computation time, unrealistic for GTOC, had been dedicated to it.

In the end, this tree search with randomization would output nearly-complete campaigns. Usually there would still be 2-8 debris remaining. Although it may have been possible to stitch them through reordering, we did not have the computational capabilities to perform this operation for most missions. There was therefore the need to "finish" the campaign with additional missions.

The final solution submitted is shown in Figure 3. We see that, as expected, each mission is largely chosen based on the RAAN of each debris.

5 The Finisher

The algorithm known as the "finisher" aims to obtain a 123 debris campaign given the previously stitched chains and the remaining 2-8 pieces of debris. In a certain way, it repeats the first steps of computation using computational brute force; i.e., considering a big portion of the possible combinations between spares.

First, we find the remaining time gaps in which there are no missions scheduled. Then, applying the time margins to be held due to operational constraints, we now have the time windows to compute transfer maneuvers between spare debris.

We compute links between each pair of sp are debris. This ΔV is computed for varying transfer times, initial times, debris objects and time gaps. The ΔV is then stored and sorted to find the maximum number of possible pairs of single launches that can be combined to launch together.

We sort the sets of debris pairs by total ΔV of the respective maneuvers between the pairs found. The next step is to use the "stitcher" [4] again to reduce the number of missions of every set of pairs of debris by stitching the remaining spare debris to these pairs. The combination of missions that minimizes the total cost of the campaign is chosen.

As an example, a campaign of missions may have 7 spare debris after running the "stitcher" algorithm. Without further work, each of those 7 debris will require separate launches, which is very expensive. The cost of the spare debris can be reduced significantly by combining the debris into fewer launches. The first pass through the "finisher" may find 3 pairs of debris that can be launched together, bringing the number of launches down from 7 to 4 (3 launches which each visit two debris, and 1 single launch). The second pass through the "finisher" will reduce these 4 launches into perhaps 2 or 3 launches. While the cost of removing these debris is still high, it is greatly reduced from the cost of 7 single launches.

6 The Wiggler

After creating full campaigns of missions, the "wiggler" tool was used to slightly adjust the date at which each debris was visited. A nonlinear programming (NLP) problem was defined with the times between each debris rendezvous as the optimization variables. The analytical approximate ΔV of the whole mission was minimized, subject to operational constraints from the problem statement. This NLP was solved with MATLAB's fmincon solver, using the Interior Point method. The debris at the beginning of the mission and at the end of a mission were held constant to avoid inadvertently invalidating other missions, while the times between debris within the mission were allowed to vary between 7 and 29 days. Typically, the "wiggler" tool would adjust each date by a fraction of a day, and it would reduce the ΔV of each mission by 2-10%.

7 Final Optimization & Integration

A two-step algorithm was developed to transition from the approximate, analytical model described above to the fully-integrated solution for submission. During the final optimization and integration, the arrival times at each debris were held fixed, and the transfer between each pair of debris was considered separately. The algorithm, variables, and series of maneuvers are shown graphically in Figure 2.

The first step of the algorithm is to choose maneuver $\Delta\vec{V_1}$ at time t_1 so that the RAAN Ω and argument of latitude u of the spacecraft match the corresponding elements of debris $i+1$ at time t_2 (approximately 20 days later). Time t_1 is chosen to be the time when the spacecraft's argument of latitude is equal to zero. We never use the propulsion system to directly change Ω — rather, we change semimajor axis a, eccentricity e, and inclination i to indirectly change Ω by leveraging the natural dynamics. Maneuver $\Delta\vec{V_1}$ is defined in a VNC (velocity, normal, co-normal) frame, with components in the V and N directions. The component in the V direction immediately changes the semimajor axis and

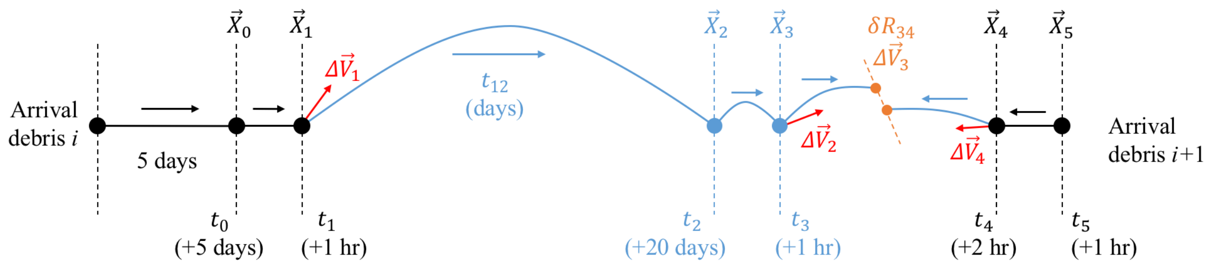


FIGURE 2. *Schematic showing the four maneuvers used in the final integration step to transfer from one debris to another. The dates of arrival at debris i and $i + 1$ are held fixed. Black, straight lines indicate motion according to the debris' approximate equations of motion, while blue, curved lines indicate motion according to numerically propagating the Earth J_2 equations of motion. The times t_0 through t_5 are indicated, with typical values given of each relative to the previous time. The horizontal arrows indicate whether a segment was propagated forward or backward.*

eccentricity of the spacecraft's orbit, while the component in the N direction immediately changes the inclination. Only the V component affects the argument of latitude target, while both components have an indirect effect over time on the RAAN target because of the J_2 dynamics. For a given number of orbital revolutions, there is an exact solution for the V and N components of the maneuver that satisfy the constraints on Ω and u. After performing $\Delta\vec{V}_1$, the spacecraft coasts for up to approximately 25 days to state $\vec{X}_2$ at time t_2 (exactly three hours before the nominal rendezvous with debris $i + 1$), which can be up to approximately 400 orbital revolutions. The number of orbital revolutions was chosen to minimize the magnitude of $\Delta\vec{V}_1$. Later insights revealed that it would be more optimal to choose the number of orbital revolutions to minimize the total ΔV, but it was not possible to implement this given the time constraints of the competition.

The second step of the algorithm is to choose maneuvers $\Delta\vec{V}_2$, $\Delta\vec{V}_3$, and $\Delta\vec{V}_4$ to adjust the spacecraft orbit's inclination, semimajor axis, argument of perigee, and true anomaly to rendezvous with debris $i+1$. To do this, we defined an optimization problem with 8 variables: t_{23} (the forward propagation time after t_2 until performing maneuver $\Delta\vec{V}_2$), t_{45} (the backward extra time from the nominal arrival time at debris $i + 1$), the vector elements of $\Delta\vec{V}_2$, and the vector elements of $\Delta\vec{V}_4$. We propagate forward (with numerical integration) from time t_2 to time t_3 and perform maneuver $\Delta\vec{V}_2$ at time t_3. We also propagate backward (according to the debris dynamics) from t_5 to time t_4 and perform maneuver $\Delta\vec{V}_4$ at time t_4. We then shoot forward from time t_3 and backward from time t_4, constraining the position discontinuity $\delta\vec{R}_{34}$ to be $\vec{0}$ and defining the velocity discontinuity to be $\Delta\vec{V}_3$. MATLAB's fmincon optimizer is used with the Interior Point algorithm to minimize the total ΔV for maneuvers 2, 3, and 4 and remove the discontinuity at the midpoint of times 3 and 4.

It was found that the ΔV for the integrated solution and for the analytical estimate agreed well for transfers under 1 km/s. However, the algorithm had difficulty converging to an optimal solution when the total ΔV exceeded 1 km/s. We expect that further refinements to the final optimization algorithm could have improved the cost of these high-ΔV transfers, but we also acknowledge that these high-cost transfers could be removed entirely by better pruning techniques in the analytical search.

8 Discussion & Conclusions

Although the problem was definitely very hard to solve, we think that many constraints actually narrowed the strategy possibilities. For instance, by limiting the number of days between debris to 30, it was not possible to have a single mission waiting for an extended period of time. The total duration and the number of debris already imposed a fast rhythm of encounters (24 days on average), hence allowing to wait between debris would have added an interesting element or risk-reward: maybe a mission can get one more debris if it waits for 60 days, but is the time wasted really worth it?

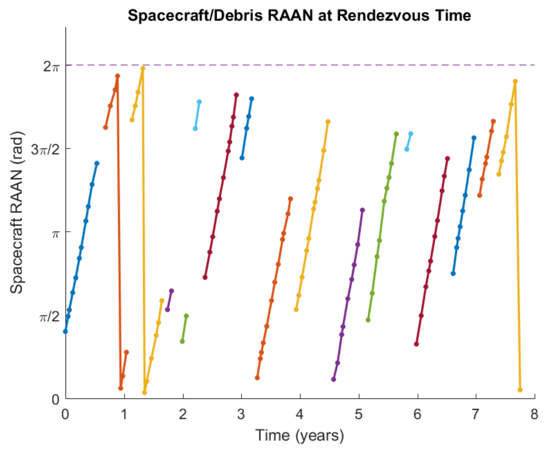


FIGURE 3. *The final solution of team CU Boulder. Each line series is a separate launch, with 17 total. The clear trend apparent in the RAAN of debris visited captures the driving dynamic of the problem.*

The real-time variation of the costs of each mission was an interesting element of the competition, but we feel it was ultimately a distraction that, if anything, only benefited those teams whose time availability happened to coincide with the competition. In the end, almost every team's best solution was submitted in the final hours, so there was no tangible advantage to submitting earlier. In a complete solution, all the missions are tightly related to each other, so it is not practical to submit one single mission early. Future competitions could make better use of this time-varying mechanic by making component parts of the full solution more separable from the whole. For example, if there were thousands of debris, and each team were to remove only some subset, then it would be feasible to design a single mission early in the competition that did not interfere with other missions designed later.

The leaderboard mechanic was very exciting for the competition, and also helped our team know what the ideal solution should look like. Towards the end of the competition, we were able to constrain our search space to be similar to the best solutions on the leaderboard.

Team CU Boulder is grateful to Dr. Dario Izzo for his work organizing a challenging, well-organized competition. Thank you.

9 Acknowledgements

We would like to thank the following individuals for their advice and help: Prof. Natasha Bosanac (CU Boulder), Dr. Jeffrey Parker (Advanced Space, LLC), and Prof. Christoffer Heckman (CU Boulder). This work was supported by a NASA Space Technology Research Fellowship.

References

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

Acta Futura 11 (2018) 99-107

DOI: 10.5281/zenodo.1139284

Acta Futura

GTOC 9: Results from Michigan Technological Univeristy and University of Michigan (MTU-UoM)

Ehsan Taheri[‡], Di Wu[‡], Ossama Abdelkhalik[†][*], Shangyan Zou[†], Kaushik Prabhu[†]
Shadi Darani[†], Brandon Jackson[†]

[‡]Department of Aerospace Engineering University of Michigan, Ann Arbor, MI 48109 (USA)
[†]Department of Mechanical Engineering-Engineering Mechanics Michigan Tech University
Houghton, MI 49931 (USA)

Abstract. This paper presents the methods developed by the Michigan Tech Univeristy and University of Michigan (MTU-UoM) team in the 9^{th} GTOC along with the obtained results. Several concepts were investigated, specially regarding the selection of the sequence of debris to be removed in each mission. These concepts will be briefed in this paper and the concept that produced this team's best solution is presented in detail. A genetic algorithm is used as an outer loop optimization tool to determine the sequence of debris to be removed. An inner loop optimizer is used to tune the individual transfers in each mission. This team's best solution consists of 16 missions that removes 122 debris with a cost of 1192.74 MEURs.

1 Introduction

The GTOC9 problem description is detailed in reference [1], and it is not presented here to avoid duplication with other papers in this special issue of the journal. The overall goal is to remove 123 debris in a Low-earth orbit (LEO); to avoid the Kessler effect [2]. Some of the orbital elements of the debris are very close to each other, whereas the introduction of the J_2 perturbation changes the right ascension of the ascending node (RAAN) and argument of perigee of the orbits (see Appendix).

For two-body Kepler orbits, it is possible to use the well known Lambert solver to compute the impulsive maneuver needed to rendezvous with a debris, given the initial position of the spacecraft, the final rendezvous position, and the time of flight of the maneuver. Due to the J_2 effect included in this competition, this tool cannot be used to compute an exact transfer. As a result, this team has developed a modified Lambert solver during this competition that results in solutions that are closer to the exact solution compared to the two-body Lambert solver. As a result this modified Lambert solver enabled the optimizer to find better solutions. This modified Lambert approach is described in the Appendix. The search for the best combination of missions resulting in the lowest value of the cost function was performed in two main steps. The first step is a global search among the 123 debris to generate individual missions in a sequential manner. The number of the remaining debris gets smaller as more missions are constructed. Each mission consists of a number of

[*]Corresponding author. E-mail: ooabdelk@mtu.edu

legs; each leg defines a trajectory between two debris. Few different strategies were investigated in this missions construction step.

In the strategy used in the submitted solution, the launch date and flight times of all legs, for each mission, are the design variables of an optimization problem, in which the goal was to minimize the the total impulse value. Later the goal was to find the most efficient missions defined by the ratio of the number of visited debris to the consumed propellant. The optimization problem exploits solutions to multi-revolution modified Lambert problem.

The second step involves a local optimization which is performed over the missions obtained in the first step. Assuming a fixed sequence of debris, the design variables of the local optimization are the launch date and the flight times of all legs of an individual mission.

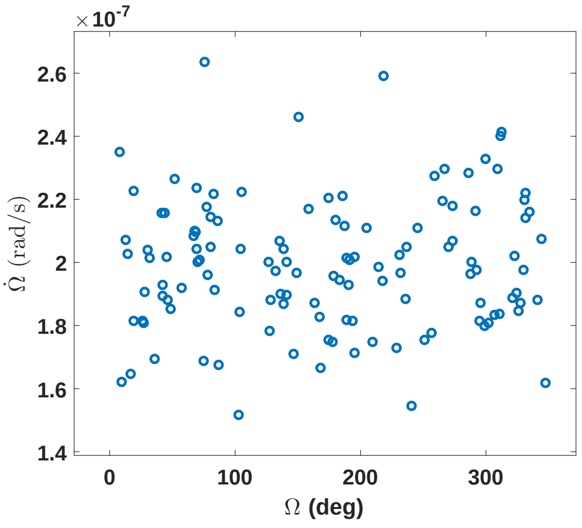


FIGURE 1. *Plot of $\dot{\Omega}$ versus Ω for the respective epoch time of each debri.*

2 Sequence of Debris

Given the large number of permutations of the sequence of debris (large design space), it is vital to exploit very rapid measures for conducting a broad search to find the sequence of debris in each mission. Several concepts were investigated; here the most significant of them are briefed and the concept used to generate this team's best solution is detailed at the end.

Semi-Free Rides

Generally speaking, plane-change maneuvers are more expensive compared to in-plane maneuvers. Two angles determine the plane: the inclination and the RAAN. The inclinations of the debris do not change due to the debris motion while the RAAN changes due to the J_2 effect. So, the concept presented in this section utilizes this change in the RAAN due to the J_2 effect, to schedule rendezvous times when the values of RAANs of two debris are very close; hence the spacecraft can wait with one debris until a good time (when another debris has the same RAAN) and then start a maneuver to rendezvous with the new debris. The spacecraft will then wait with the new debris until a new debris achieves a RAAN that is close to the spacecraft's RAAN, and so on.

This concept generated a nice sequence of debris with very low cost maneuvers; however the time of flight in each maneuver is significantly high violating the 30 days constraint defined in the problem description. As a result the sequences generated using this approach were not submitted.

The planes of all the debris are changing over time. We can compute the date at which two arbitrary planes would have equal RAAN as follows:

$$\Omega_1 = \Omega_{10} + \dot{\Omega}_1 (T - T_{epo1}), \tag{1}$$

$$\Omega_2 = \Omega_{20} + \dot{\Omega}_2 (T - T_{epo2}), \tag{2}$$

where T_{epoi} is the epoch date of debris i and T is the current time. If we solve for $\Omega_1 = \Omega_2 + 2n\pi$, we will have:

$$T = \frac{\Omega_{20} - \Omega_{10} + \dot{\Omega}_1 T_{epo1} - \dot{\Omega}_2 T_{epo2} + 2n\pi}{\dot{\Omega}_1 - \dot{\Omega}_2}. \tag{3}$$

The feasible range for the MJD of this problem is from 23467 to 26419. For the cases when the date of the intersection is out of the feasible range, the data is overwritten by a big number.

Although we have the best date to maneuver between the two planes, we still cannot guarantee the spacecraft is exactly at the intersection point. In this work, it is decided that a plane-change maneuver will only be conducted when the spacecraft is close to the intersection point of the two planes so that the cost of the maneuver is lower. So then we compute the time of wait unit the spacecraft can reach the intersection point. The argument of latitude of the intersection and the argument of the spacecraft are computed:

$$\begin{aligned} A_{sc} &= \omega_{sc} + \theta_{sc}, && (4)\\ \delta\Omega &= \Omega_f - \Omega_i, && (5)\\ \cos\alpha &= \cos i_i \cos i_f \\ &\quad + \sin i_i \sin i_f \cos\delta\Omega, && (6)\\ \sin A_{la} &= \sin i_f \sin\delta\Omega / \sin\alpha, && (7) \end{aligned}$$

where A_{sc} is the argument of the position of the spacecraft, the A_{la} is the argument of latitude of the intersection. Ω_f and Ω_i are the right ascension of the initial plane and the final plane, respectively. i_i and i_f are the inclination of the initial plane and the final plane. α is the angle of the plane change. When we find $A_{la} = A_{sc}$ we can start the maneuver. First a single-impulse plane change maneuver is computed. Then an in-plane maneuver is computed to rendezvous the spacecraft with the debris. The combination of the previous two maneuvers is a two-impulse maneuver that will rendezvous the spacecraft with the debris. The cost of the plane-change maneuver can be computed as:

$$\Delta V = 2V_i \sin(\Delta\alpha/2), \qquad (8)$$

where V_i is the velocity of the spacecraft on the initial orbit and $\Delta\alpha$ is the difference in the inclination of the orbits. The cost of the in-plane transfer can be computed by solving Lambert problem. To have an initial guess for the real cost of the in-plane transfer, we will hold the plane still, and assume the spacecraft has the unperturbed Keplerian motion. Finally, the plane-change maneuver and the departure cost of the in-plane transfer will be combined. This combined maneuver can also be obtained using the modified Lambert algorithm presented in the Appendix.

Hidden Genes Genetic Algorithms

In this approach, a hidden genes genetic algorithm (HGGA) was implemented to carry out a global search as opposed to a sequential search. Details of the HGGA can be found in [3, 4, 5].

To solve the problem, it can be divided into several missions (m_i) and in each mission some debris (N_{di}) can be captured by a spacecraft. In general, m_i and N_{di} are not known a priori. If we assume that each mission is solved at a time, the only variable that makes the problem a Variable Size Design Space (VSDS) problem is the number of debris at each mission. The design variables in each mission are launch time, arrival time, number of debris (N_{di}), debris IDs (debris that are captured in the mission), wait time, and Deep Space Maneuvers (DSMs) direction and magnitude.

The problem is solved in two phases. In the first phase, the J_2 effect is ignored and launch/arrival time, time of flight, number of debris, and debris IDs are optimized, and in the second step, the effect of J_2 is corrected by adding a DSM in each leg. It is assumed that there is no DSM in the first phase and there is only one DSM in each leg in the second phase. In the first phase, the Lambert problem is solved to find the trajectory between each two debris.

Assume that the current debris is D_i and the next debris is D_{i+1}. At the end of the wait time at D_i and before the departure impulse, the position of the spacecraft is known (similar to the position of debris D_i). Since the time of flight is known, the Lambert problem can be solved to find the departure and arrival impulses. This can be done for all the legs until the last debris of the mission. After the first phase, the effect of the J_2 is corrected by assuming a DSM in each leg. In this algorithm, debris selection is done automatically and there is no need to categorize them into groups. The solution generated using HGGA was not competitive due to the large design space that the HGGA needs to work with.

Sequential Search

Our broad search strategy uses the remaining fuel, where trajectories are built in a sequential manner by adding new legs. The maneuvers are already impulsive for which Lambert problem with a bi-impulsive transfer is considered in the preliminary phase. In the final step, the missions are optimized individually using a local optimizer while taking into account the J_2 perturbation into the governing equations of the spacecraft motion.

Some of the orbital elements of the debris are very close to each other, whereas the introduction of the J_2 perturbation varies the RAAN and the argument of perigee of the orbits (see Appendix). Therefore, the debris have different values for $\dot\Omega$.

Figure 1 depicts the distribution of the data points on the $\dot\Omega - \Omega$ plot. A more important factor, though, is the evolution of the RAAN over certain time intervals for it is possible to glean closeness information (in terms of RAAN) in order to form clusters of debris. The evolution of the RAAN is a linear relation,

$$\Omega = \Omega_0 + \dot\Omega \times (t - t_0), \qquad (9)$$

where Ω_0 is the value of the RAAN at the epoch time

(t_0). This linear relation can be utilized to construct a closeness criterion to be used for clustering debris. On the other hand, the number of debris considered in this problem is significantly smaller than the involved bodies of the previous GTOC problems. While the RAAN has a significant effect on the value of the impulses, initially, we decided to look for generating a series of missions that has the lowest value of cost. Then, we would perform a post-analysis to switch the debris between missions based on the closeness of the RAAN.

Our primary broad search method was to construct individual missions, in a sequential manner, by using a branch-and-prune tree search algorithm. Each mission is built by connecting a series of legs. The building up of sequential legs consists of two main loops: the first loop iterates over the departure debri ID and the second loop iterates over the arrival debri ID.

To find an optimal solution between each pair of debris, a hybrid optimization method is devised. First, a standard genetic algorithm (GA) performs a broad search over the departure time (MJD_{dep}) and time of flight (TOF) within their defined ranges. The departure time is defined in the range $MJD_{dep} \in [MJD_{LB}, MJD_{UB}]$ where the lower bound and upper bounds of the departure time are $MJD_{LB} = MJD_{arrival} + Stay_{time}$ and $MJD_{UB} = MJD_{LB} + 29$, respectively. The $Stay_{time}$ of 5 days is one of the constraints necessary for deploying the de-orbiting package. In addition, the transfer time between any two debris should not take more than 30 days.

The time of flight of each leg is also defined in the range of $TOF \in [TOF_{LB}, TOF_{UB}]$ where the lower bound and upper bounds of the time of flight are $0.1 \times T_{orb}$ and $5 \times T_{orb}$. Semi-major axis of the debris are relatively close to each other and the period of their orbit is approximately the same. Therefore, we defined the limits of the TOF in terms of the average orbital period T_{orbit} and its value is set to 100 minutes. Two important parameters of a GA are the number of generations and populations. We set those parameters to 20 and 50, respectively.

Eventually, the solution of the GA is used as an initial guess for a local optimizer to further reduce the cost function. For each leg, the departure time (MJD_{dep}) and time of flight (TOF) are the two design variables, and the optimization objective was to minimize the sum of the two impulses,

$$J = \min_{MJD_{dep}, TOF} \Delta v_1 + \Delta v_2, \tag{10}$$

where Δv_1 and Δv_2 are the magnitude for impulses at departure and arrival instances, respectively. For the hybrid optimization we did not define any constraint mainly due the fact that the constraint handling is dealt with at the final verification stage in which we use a local optimizer. Each individual execution of GA-Fmincon hybrid optimization takes on average 0.2 seconds (running on 8-cores).

The solution to the Lambert problem is used extensively in the hybrid optimization method, for which we used a multiple-revolution Lambert solver [6] and set the maximum number of revolutions to 5. In addition, we used a compiled Mex file C++ implementation of the Lambert solver. Note that the actual transfer occurs during a short interval. This will simplify the local optimization step during which the accumulative effect of J_2 perturbation becomes small.

Once a solution, which consists of individual missions, was generated through the broad search algorithm, we performed a post-analysis to modify the missions by performing two major changes. The first change was to remove the last missions that may consist of only one leg, i.e., single-leg missions and to re-assign their debris to the previous missions. The second change was to inspect all of the missions and remove those legs that required significantly greater values of impulse compared to the other legs, and re-assign those debris to other missions.

Debris Re-assignment

The re-assigning strategy that we considered exploits the RAAN closeness which is explained in this section. For any debri which is to be re-assigned, we calculated the closeness criterion

$$\eta = \frac{\sum_i^n (\Omega^*_{debri}(t_i) - \Omega_{debri}(t_i))^2}{n}, \tag{11}$$

where Ω^*_{debri} denotes the RAAN of the debri, which is going to be re-assigned (evaluated at the descretized points), and $t_i \in [MJD_{LB}, MJD_{UB}]$. Note that MJD_{LB} and MJD_{UB} correspond to the lower and upper bounds of the mission MJD time interval and are known values. We adopted a simple equi-distant discretization of the mission time interval. n is the number of dicretization points (that depends on the step size used for discretization) and Ω_{debri} is the RAAN of one of the debri to which we compare the relative differences. The minimum value of the closeness criterion gives us a measure to assign any new debri to a particular mission.

For instance, if there is a mission which already contains 10 debris, we calculate the above parameter by comparing the closeness criterion between the new debri and each of the 10 debris, and take the lowest value. Then, we would repeat the same procedure for the other missions and store the respective RAAN closeness value, η. Finally, the minimum value of η determines the mission to which we should assign the new debri. The above steps are followed for the other debris until all of them are re-assigned. In addition, we can avoid re-assigning new debris to the original missions from which we picked them. This will ensure that the debris are assigned to new missions.

After performing the above steps, some of the missions will be modified and a new tree-search optimization is performed to achieve a minimum-cost mission that visits all of the debris within each mission. Another consideration is to modify the allowed duration interval of a mission to make sure that there is enough time to visit all of the derbi within each mission. The task of modifying time is the tricky part of the re-assignment. However, the optimization has to be performed over a reduced number of debris within a mission (usually on the order of 25 or less).

3 Results

The final solution consists of 16 missions that deorbits 122 debris with a total cost of 1192.743 MEURs.

Tables 1 to 16 summarizes the individual missions of the submitted solution. Note that each row of the table only reports the dates corresponding to the departure and arrival impulse dates, MJD_{dep} and $MJD_{arrival}$, respectively, between the departure debri ID and the arrival debri ID. The spacecraft de-orbits a considerable number of debris during the first three missions. Despite our efforts to remove the last two missions and to re-assign their debris into the previous missions, our tree-search algorithm was not capable of finding feasible missions after re-assigning them to other missions.

Figure 3 depicts the variation of the RAAN of the debris visited in the first mission. Note that the apparent separation of bands of lines with similar slope is due to the fact that the angles are not brought into the interval of $[0, 2\pi]$

TABLE 1. *Summary of mission #1*

MJD$_{dep}$	MJD$_{arrival}$	Dept. Deb #	Arri. Deb #
23680.147	23680.175	19	61
23704.058	23704.303	61	107
23727.367	23727.547	107	30
23736.618	23736.734	30	85
23764.976	23765.083	85	41
23782.362	23782.529	41	45
23789.213	23789.415	45	11
23801.012	23801.203	11	82
23816.460	23816.490	82	71
23844.768	23844.932	71	115
23867.137	23867.386	115	43
23873.892	23874.146	43	47
23900.408	23900.656	47	26
23918.438	23918.679	26	109
23929.680	23929.868	109	7
23958.427	23958.531	7	2

TABLE 2. *Summary of mission #2*

MJD$_{dep}$	MJD$_{arrival}$	Dept. Deb #	Arri. Deb #
24007.834	24008.027	72	51
24014.289	24014.398	51	10
24020.713	24020.965	10	69
24037.514	24037.657	69	66
24046.992	24047.242	66	73
24074.129	24074.159	73	28
24099.494	24099.639	28	64
24115.494	24115.684	64	52
24144.436	24144.732	52	12
24166.146	24166.399	12	3
24193.416	24193.677	3	31
24202.086	24202.332	31	65
24227.436	24227.702	65	91

TABLE 3. *Summary of mission #3*

MJD$_{dep}$	MJD$_{arrival}$	Dept. Deb #	Arri. Deb #
24279.950	24279.998	81	13
24288.991	24289.174	13	32
24315.744	24316.037	32	22
24331.734	24331.777	22	17
24352.800	24353.076	17	105
24381.354	24381.452	105	59
24404.094	24404.245	59	98
24426.367	24426.654	98	46
24444.178	24444.216	46	83
24467.346	24467.380	83	48
24495.328	24495.623	48	99
24504.925	24505.101	99	96
24533.435	24533.635	96	114

TABLE 4. *Summary of mission #4*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
24593.279	24593.535	0	122
24622.047	24622.337	122	74
24640.378	24640.552	74	119
24667.886	24668.010	119	104
24674.232	24674.284	104	24
24680.834	24681.046	24	108
24707.583	24707.619	108	37

TABLE 5. *Summary of mission #5*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
24769.598	24769.714	55	93
24779.413	24779.534	93	100
24801.111	24801.140	100	90
24807.694	24807.994	90	9
24815.556	24815.841	9	33
24842.602	24842.728	33	21
24853.887	24854.082	21	106
24871.127	24871.394	106	68
24893.691	24893.833	68	118
24917.925	24918.101	118	113

TABLE 6. *Summary of mission #6*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
24966.805	24967.058	76	27
24974.934	24975.190	27	20
24996.868	24997.089	20	102
25021.045	25021.332	102	80
25033.293	25033.326	80	121
25062.310	25062.567	121	116
25087.865	25088.165	116	4
25115.532	25115.757	4	15

TABLE 7. *Summary of mission #7*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25159.145	25159.411	35	1
25167.633	25167.878	1	40
25173.101	25173.358	40	62
25187.643	25187.861	62	54
25212.934	25213.041	54	89
25239.149	25239.255	89	112
25263.523	25263.714	112	87

TABLE 8. *Summary of mission #8*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25327.371	25327.669	60	103
25341.232	25341.436	103	39
25346.811	25347.072	39	5
25371.826	25372.103	5	53
25377.357	25377.602	53	101
25400.698	25400.955	101	78

TABLE 9. *Summary of mission #9*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25447.187	25447.477	110	79
25473.209	25473.464	79	34
25485.266	25485.520	34	97
25510.400	25510.575	97	50
25535.869	25536.150	50	86
25542.053	25542.285	86	6

TABLE 10. *Summary of mission #10*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25584.694	25584.991	25	94
25594.707	25594.828	94	120
25618.420	25618.691	120	38
25623.691	25623.839	38	42
25641.463	25641.636	42	56
25653.755	25653.820	56	111

TABLE 11. *Summary of mission #11*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25710.022	25710.291	95	8
25734.377	25734.542	8	49
25739.921	25740.060	49	84
25746.989	25747.024	84	36
25769.358	25769.460	36	75

TABLE 12. *Summary of mission #12*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25871.606	25871.896	88	117
25877.672	25877.972	117	18
25884.791	25885.050	18	70

TABLE 13. *Summary of mission #13*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
25933.550	25933.729	14	58
25962.043	25962.246	58	63

TABLE 14. *Summary of mission #14*

MJD_{dep}	$\text{MJD}_{arrival}$	Dept. Deb #	Arri. Deb #
26098.760	26099.087	57	67
26107.636	26107.911	67	44

TABLE 15. *Summary of mission #15*

MJD$_{dep}$	MJD$_{arrival}$	Dept. Deb #	Arri. Deb #
26151.463	26151.788	77	29

TABLE 16. *Summary of mission #16*

MJD$_{dep}$	MJD$_{arrival}$	Dept. Deb #	Arri. Deb #
26205.748	26206.085	23	16

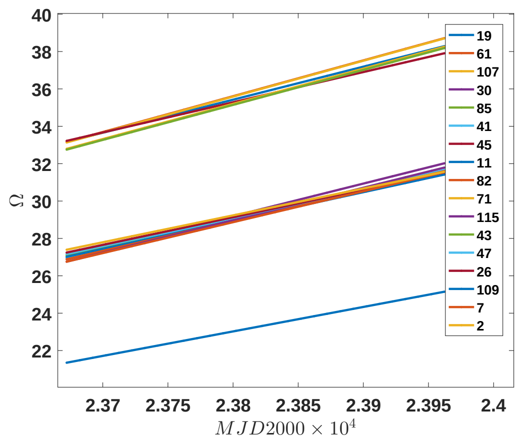


FIGURE 2. *Evolution of the RAAN of the debris in the first mission $MJD2000 \in [23672.248, 23963.531]$.*

Acknowledgment

This work was partially funded by National Science Foundation, award # 1446622

4 Conclusion

Team MTU-UoM employed a set of tools which were sufficient to find a good solution to GTOC9 problem. A major enhancement would have been to utilize an efficient tree-search algorithm. In addition, it would be ideal to perform, early on, a clustering strategy in terms of the right-ascension of the ascending node, and then focus on visiting the debris within each cluster. In addition, for each mission, plot of $\dot{\Omega} - \Omega$ is helpful in fixing the sequence of debris (transfers) and consider only the stay time and the time of transfer as design variables. The debri re-assignment strategy that we considered during the competition time can be performed more efficiently.

Although we made progress in GTOC9, there is a considerable gap between our solution and the solutions submitted by the top-rank teams. There are still a lot of works for us to do in trajectory design and optimization. We have only used personal desktop computers and exploited parallel capability of MATLAB running our codes on eight cores. It is reasonable to run our codes on clusters with access to a greater number of cores. We should also consider developing our codes on compiled programming languages, such as C or Fortran. In addition, developing a capable local optimizer (other than MATLAB's *fmincon* is quite important for achieving improved solutions.

Appendix: Modified Lambert Solver

The Lambert solver finds the bi-impulse maneuver necessary to rendezvous with a debris given the initial spacecraft position, the final rendezvous position, and the time of flight of the maneuver, assuming Keplerian motion. Due to the oblateness of the Earth, the keplerian motion will be perturbed by the J_2 effect. The J_2 effect will influence the right ascension, the argument of the perigee and the mean anomaly. The latter was neglected in GTOC 9.

$$\dot{\Omega} = -\frac{3}{2} J_2 (\frac{r_{eq}}{p})^2 n \cos i, \tag{12}$$

$$\dot{\omega} = \frac{3}{4} J_2 (\frac{r_{eq}}{p})^2 n (5 \cos^2 i - 1). \tag{13}$$

The change rate of Ω and ω is linear.

$$\Omega - \Omega_0 = \dot{\Omega}(t - t_0), \tag{14}$$

$$\omega - \omega_0 = \dot{\omega}(t - t_0). \tag{15}$$

To solve the two-body transfer problem with the J_2 effect, the following strategy is developed. For a two-body transfer problem without J_2 effect and with the date of the departure, the arrival and the fixed time of flight (TOF), usually Lambert problem is used to find the transfer orbit. The required impulse for the departure Δv^d and the arrival Δv^a can be calculated. However, the problem under study takes into account the J_2 effect. Here, the solution obtained from Lambert is used as initial guess. An optimization algorithm will be introduced here for solving the perturbed transfer orbit. The objective function is constructed as:

$$J = \|\vec{r}^a_{sc} - \vec{r}^a_{Debris}\|, \tag{16}$$

where $\vec{r}^a_{sc}$ is the position vector of the spacecraft at the final time which is also the arrival time. $\vec{r}^a_{Debris}$ is the position vector of the debris at the arrival time. The goal of the optimization is to minimize the objective function which means we want to satisfy the rendezvous condition. The variables to be optimized is $\vec{v}^d_{sc}$, the velocity of the spacecraft at departure time. The optimization algorithm is setup as:

$$\begin{aligned}
Min \quad &: \quad J = \|\vec{r}^a_{sc} - \vec{r}^a_{Debris}\|, \\
s.t \quad &: \\
\ddot{x} &= -\frac{\mu x}{r^3}(1 + \frac{3}{2}J_2(\frac{r_{eq}}{r})^2(1 - 5\frac{z^2}{r^2})), \\
\ddot{y} &= -\frac{\mu y}{r^3}(1 + \frac{3}{2}J_2(\frac{r_{eq}}{r})^2(1 - 5\frac{z^2}{r^2})), \\
\ddot{z} &= -\frac{\mu z}{r^3}(1 + \frac{3}{2}J_2(\frac{r_{eq}}{r})^2(3 - 5\frac{z^2}{r^2})), \\
&\|\vec{r}^a_{sc} - \vec{r}^a_{Debris}\| < 0.1,
\end{aligned}$$

where the first three constraints represent the perturbed Keplerian motion. The last constraint is for the rendezvous condition - the difference between the position of the spacecraft and the position of the debris cannot be bigger than 0.1 km. Due to the J_2 effect, if we propagate the perturbed trajectory with the $\vec{v}^d_{sc}$ obtained from original Lambert solution, the final position of the spacecraft does not reach $\vec{r}^a_{Debris}$. To take advantage of J_2 effect, the Lambert problem can be used to solve the transfer orbit between $\vec{r}^d_{sc}$ and $\vec{r}^a_{temp}$. A temporary position $\vec{r}^a_{temp}$ is computed from a modified set of orbital elements. Assume the orbital elements at the arrival time are $[a^a,\, e^a,\, i^a,\, \Omega^a,\, \omega^a,\, M^a]^T$. The modified orbital elements are computed from:

$$\Omega_{temp} = \Omega^a - \dot{\Omega}T, \tag{17}$$

$$\omega_{temp} = \omega^a - \dot{\omega}T, \tag{18}$$

where T is the time of flight. So the temporary position is computed from the modified orbital elements: $[a^a,\, e^a,\, i^a,\, \Omega_{temp},\, \omega_{temp},\, M^a]^T$. In this competition, $\dot{\Omega}$ is always a positive number while $\dot{\omega}$ is always negative. If Ω^a is greater than Ω^d which is the right ascension of the orbit before we apply the departure impulse which means we have the possibility to take advantage of J_2 effect. After we compute Ω_{temp}, if we found $\Omega_{temp} > \Omega^d$, then we are able to apply the modification for the orbital elements, otherwise we will apply $\Omega_{temp} = \Omega^d$. If Ω^a is smaller than Ω^d, which means we are moving against J_2 effect, the modification for the orbital elements would not be applied, we will have $\Omega_{temp} > \Omega^a$. Since $\dot{\omega}$ is negative, so the opposite algorithm will be applied to compute ω_{temp}. The description above can be summarized as

```
**if** $\Omega^a < \Omega^d$ **then**
  $\Omega_{temp} = \Omega^a$
**else**
  **if** $\Omega_{temp} > \Omega^d$ **then**
    $\Omega_{temp} =$ Eq. (17)
  **else**
    $\Omega_{temp} = \Omega^d$
  **end if**
**end if**
**if** $\omega^a > \omega^d$ **then**
  $\omega_{temp} = \omega^a$
**else**
  **if** $\omega_{temp} > \omega^d$ **then**
    $\omega_{temp} = \omega^d$
  **else**
    $\omega_{temp} =$ Eq. (18)
  **end if**
**end if**
```

So the above logic along with the Eqs. (17) and (18) will be applied to compute the temporary position. The Lambert problem will be solved between the initial position and the temporary position. The solution from Lambert problem will be taken as the initial condition of the optimization. Finally, we can use Eq. ((17)) to optimize the velocity of the spacecraft at the departure point to reach the final position. The solution will be the real cost of the perturbed transfer between two points.

References

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

[2] Donald J Kessler and Burton G Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83(A6):2637–2646, 1978.

[3] Ossama Abdelkhalik. Hidden Genes Genetic Optimization for Variable-Size Design Space Problems. *Journal of Optimization Theory and Applications*, 156(2):450–468, 2013.

[4] Ossama Abdelkhalik and Shadi Darani. Hidden Genes Genetic Algorithms for Systems Architecture Optimization. In *Genetic and Evolutionary Computation Conference*, Denever, CO, July, 20–24 2016. Association for Computing Machinery Special Interest Group on Genetic and Evolutionary Computation, Advancing Computing Machinary (ACM).

[5] Ahmed Gad and Ossama Abdelkhalik. Hidden Genes Genetic Algorithm for Multi-Gravity-Assist Trajectories Optimization. *AIAA Journal of Spacecraft and Rockets*, 48(4):629–641, July-August 2011.

[6] RH Gooding. A procedure for the solution of Lambert's orbital boundary-value problem. *Celestial Mechanics and Dynamical Astronomy*, 48(2):145–165, 1990.

Acta Futura 11 (2018) 109-115
DOI: 10.5281/zenodo.1139384

Acta Futura

GTOC 9: Results from the Astrodynamics Research Group of Penn State (team ARGoPS)

Davide Conte[*], Andrew Goodyear, Jason Reiter, Ghanghoon Paik, Guanwei He, Mollik Nayyar, Matthew Shaw, Jeffrey Small, and Jason Everett

Astrodynamics Research Group of Penn State, The Pennsylvania State University
229 Hammond Building, University Park, PA 16802

Abstract. Presented in this paper are methods and results of the Astrodynamics Research Group of Penn State (ARGoPS) for the 9th Global Trajectory Optimization Competition. The optimization strategy utilized a beam search method to determine the optimal sequence of missions and transfers between debris, including the order in which to visit each group, the timing of each visit throughout the mission window in order to minimize the number of missions, and the total cost per transfer. Particle Swarm Optimization (PSO) was applied to the optimized ordering of debris visits in order to determine the departure date and time-of-flight combination which best minimize the fuel required for each transfer. This method led to a solution that collected all 123 pieces of debris from their Sun-sychronous orbits in 20 total missions.

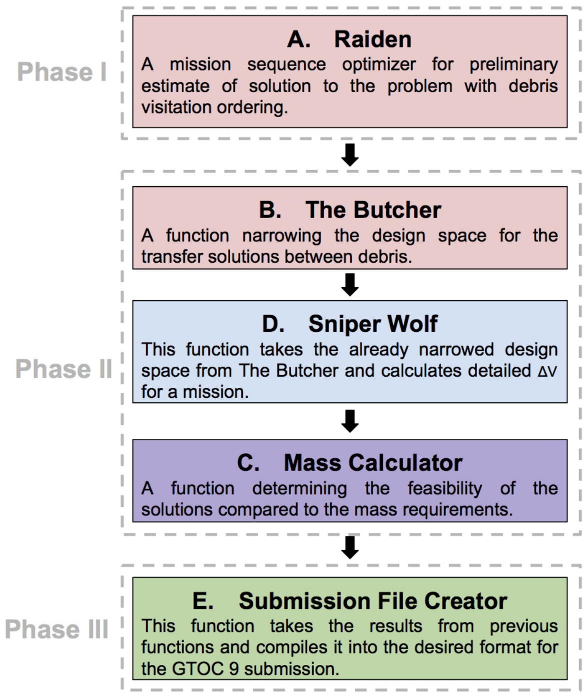
1 Introduction

The 9th edition of the Global Trajectory Optimization Competition (GTOC 9) was organized by The European Space Agencys Advanced Concepts Team in the spring of 2017. In this paper, the optimal strategy developed by team ARGoPs (Astrodynamics Research Group of Penn State University) is described. The problem presented for GTOC 9, named the ”Kessler Run”, suggests a spacecraft mission design in which a set of 123 pieces of debris are removed from orbit in order to prevent the ”Kessler effect”, a scenario in which the explosion of a satellite leads to cascading collisions of resident space objects. The detailed problem description and equations can be read in the problem description of the 9th GTOC, written by Dr. Dario Izzo [1]. Presented here is the approach and all formal details on the space debris removal mission design that was created during the month of April, 2017 by team ARGoPs.

The solution was developed by writing and running an optimization procedure in both MATLAB and the C++ programming languages. The procedure consists of three steps: Phase I (*Raiden*), Phase II (which is comprised of both *The Butcher (of Groznyj Grad)* and *Sniper Wolf*), and Phase III. Phase I of the program determines the order in which the individual debris pieces are visited and takes the first steps to narrow the search space for each transfer by using what is referred to as a beam search clustering method, as explained in Section 2. Phase II of the program takes the transfer sequence output from Phase I and further refines the search space for individual transfers via a function referred to as *The Butcher (of Groznyj Grad)*,

[*]Corresponding author. E-mail: davide.conte90@gmail.com

or simply *The Butcher*, which uses a Lambert solver within a Particle Swarm Optimization scheme to search for useful solutions within the time bounds determined by *Raiden*. Once *The Butcher* refines each search space even further, *Sniper Wolf* takes over and implements another search for transfer trajectories using Particle Swarm Optimization to find and calculate the transfers between debris objects using the full J_2-affected dynamics to ensure that the transfer will converge to a valid and sub-optimal solution (see Section 3 for further details on *The Butcher* and *Sniper Wolf*). The results from *Sniper Wolf* are checked for mass feasibility before any final solution is deemed appropriate. Phase III is then initiated, where the verified results are saved in the correct output format necessary for upload to the online submission system. The program structure is summarized in Figure 1.



A. Raiden
A mission sequence optimizer for preliminary estimate of solution to the problem with debris visitation ordering.

B. The Butcher
A function narrowing the design space for the transfer solutions between debris.

D. Sniper Wolf
This function takes the already narrowed design space from The Butcher and calculates detailed ΔV for a mission.

C. Mass Calculator
A function determining the feasibility of the solutions compared to the mass requirements.

E. Submission File Creator
This function takes the results from previous functions and compiles it into the desired format for the GTOC 9 submission.

FIGURE 1. *Program Flowchart.*

Each function used is shown in Figure 1. as an individual block. The red colored function blocks indicate lower-fidelity solutions to the problem, whereas the blue colored function block is the higher-fidelity solution. The lavender colored function block represents a final mass check and then the green colored function block indicates the final submission file creator function. The search space is progressively narrowed as more fidelity is determined for the solution. A visual representation of this search space is captured in Figure 2.

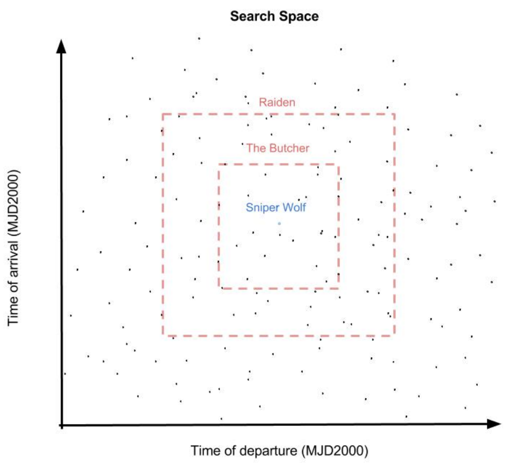
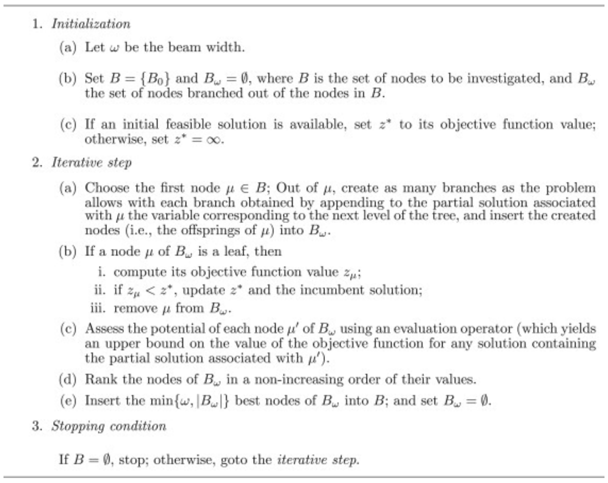


FIGURE 2. *A visual representation of the solution search space for transfers between debris.*

2 The Beam Search Clustering Method

Beam search is a heuristic algorithm that builds search trees with a restricted number of states at each level, referred to as beam width. By limiting beam width, beam search can minimize complexity and memory usage, which is beneficial for problems with a large search space. A basic understanding of this algorithm is shown in Figure 3.

A fast-paced beam search method capable of dividing the given timeline into a sequence of missions was used. This method, referred to as *Raiden*, was implemented to define the sequence by linking together debris based on the right ascension of ascending node (RAAN) and inclination of each orbit. Each mission was constructed such that no individual transfer exceeded a maximum plane change magnitude while attempting to distribute the required propellant mass evenly between missions. *Raiden* is structured to contine to link pieces of debris together until either a maximum amount of objects per mission is achieved, or the estimated propellant usage exceeds the allowable estimated propellant usage per mission. The departure



```
1. Initialization
    (a) Let $\omega$ be the beam width.
    (b) Set $B = \{B_0\}$ and $B_\omega = \emptyset$, where $B$ is the set of nodes to be investigated, and $B_\omega$
        the set of nodes branched out of the nodes in $B$.
    (c) If an initial feasible solution is available, set $z^*$ to its objective function value;
        otherwise, set $z^* = \infty$.
2. Iterative step
    (a) Choose the first node $\mu \in B$; Out of $\mu$, create as many branches as the problem
        allows with each branch obtained by appending to the partial solution associated
        with $\mu$ the variable corresponding to the next level of the tree, and insert the created
        nodes (i.e., the offsprings of $\mu$) into $B_\omega$.
    (b) If a node $\mu$ of $B_\omega$ is a leaf, then
          i. compute its objective function value $z_\mu$;
         ii. if $z_\mu < z^*$, update $z^*$ and the incumbent solution;
        iii. remove $\mu$ from $B_\omega$.
    (c) Assess the potential of each node $\mu'$ of $B_\omega$ using an evaluation operator (which yields
        an upper bound on the value of the objective function for any solution containing
        the partial solution associated with $\mu'$).
    (d) Rank the nodes of $B_\omega$ in a non-increasing order of their values.
    (e) Insert the $\min\{\omega, |B_\omega|\}$ best nodes of $B_\omega$ into $B$; and set $B_\omega = \emptyset$.
3. Stopping condition
    If $B = \emptyset$, stop; otherwise, goto the iterative step.
```

FIGURE 3. *Standard beam search algorithm. [2]*

and arrival epochs in between each piece of debris were chosen such that the transfer is performed when the two debris pieces are near their closest approach and the transfer could be completed with plenty of time to spare given the required plane change.

The beam search algorithm was applied on both the debris-to-debris scope, and the mission-to-mission scope. The starting epoch of each mission was probed randomly throughout the allowable mission timeline for a set amount of iterations and, at each probed starting epoch, *Raiden* was used to find the locally next best missions and store them in a custom beam search map structure designed to branch from in future iterations. After the stopping criteria is met that terminates the debris-level beam search algorithm, the mission-level algorithm branches from all locally optimal missions to find the next best set of missions. Each branch of the mission-level beam search algorithm is partnered with a custom time cell structure that contains information about available time intervals for future missions based on the previous missions in that specific branch. The only stopping criteria of the mission-level beam search algorithm is the remaining number of debris available to be linked in a mission, based on the locations of the debris throughout the allowable time intervals. The various mission sequences that were developed using Raiden were named as follows (in order of increasing optimality, and amount of debris captured): *The Pain*, *The Fear*, *The End*, and *The Fury*. The Fury is the mission sequence that was ultimately chosen.

3 Trajectory Optimization

In this section, the trajectory optimization methods used to compute the individual orbital transfers between debris pieces, referred to as *The Butcher* and *Sniper Wolf*, are presented. A Keplerian approximation (solution to Lambert's problem) was used with a particle swarm algorithm to narrow the search space and give *Sniper Wolf* better initial guesses (*The Butcher*). The particle swarm algorithm is then given the derived departures and time-of-flights from the Keplerian approximation as intial guesses to find the optimal time of flight and corresponding transfer velocity in the J_2 dynamical model (*Sniper Wolf*).

Given the computationally expensive process involved in optimizing trajectories where J_2 perturbations are considered, it was necessary to find a way to decrease the search space by utilizing a lower-fidelity approximation to each orbital transfer. In fact, this lowers computational time while giving the high-fidelity optimizer a better initial guess used to eventually minimize propellant consumption. It was determined that the solution to the Keplerian Lambert's problem was a "good enough" estimate to pass to the optimizer that would take into account J_2 effects. Oblateness effects the spacecraft position drift over time from the two-body problem model (see Figure 4). In order to account for this perturbation, a deviation in the initial spacecraft velocity is needed. Thus, an accurate search space for the perturbed Lambert's problem solution was needed in order to ensure that the spacecraft would rendezvous with the debris and that the propellant needed to accomplish such maneuvers would be minimized. After multiple tests, the average transfer time between debris pieces corresponding to the optimal solutions was found to be less than half of a day, thus validating the fact that the Keplerian solution can successfully narrow down the search space for the majority of transfers (except for those transfers requiring much longer time of flight). In order to determine such a search space, a Particle Swarm Optimization (PSO) technique was used. [3] Given the desired debris ID numbers and the window available for the transfer as determined by the beam search algorithm, the available departure epoch in the transfer window and the transfer time (based on the remaining time available to complete the transfer) were used as particles in the optimization. The subsequent cost for each particle at each iteration was found by using the Keplerian Lambert's solution to determine the Δv cost of the transfer. This optimization method resulted in a single departure epoch and transfer time that would minimize the Δv cost for the transfer under purely Keplerian dynamics. Bounds were then applied to these numbers (based on the expected convergence under J_2 oblateness dynamics) in order to provide a search region for *Sniper Wolf*. This algorithm was named *The Butcher* due to the fact that it is supposed to return rough estimates that needed to be refined. Additionally, running this optimizer for all possible ranges of departure and arrival dates would result in porkchop plots, which are contour plots of Δv on departure vs. arrival date axes.

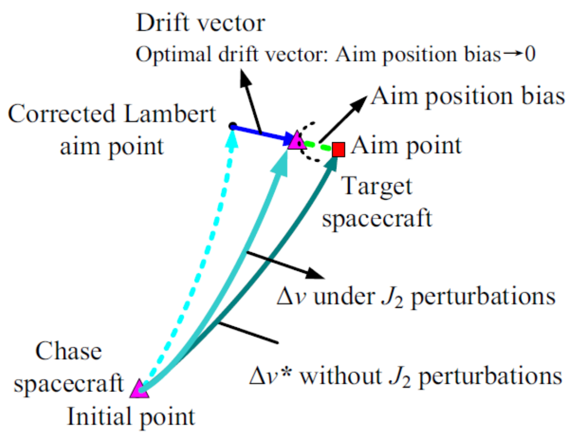

Drift vector
Optimal drift vector: Aim position bias→0
Corrected Lambert aim point
Aim position bias
Aim point
Target spacecraft
Δv under J_2 perturbations
Chase spacecraft
Initial point
Δv* without J_2 perturbations

FIGURE 4. *Drift vector effects due to J2 dynamics. [4]*

In order to find the exact necessary Δv and time-of-flight between two given debris for a range of departure and arrival dates (accounting for J_2 effects), a multi-objective PSO was implemented. This algorithm, referred to as *Sniper Wolf*, i.e. the precise, patient and (almost) infallible J_2-perturbed Lambert solver, is tasked with finding valid transfer solutions that satisfy the relative position and velocity constraints for rendezvous (100 meters and 1 m/s, respectively) while minimizing the total Δv needed to accomplish such maneuver. The basic pseudocode is shown in Figure 5. Note that only two-burn maneuvers were considered to keep the optimization tradespace dimensions (and thus computational time) to a minimum. In order to increase the robustness of *Sniper Wolf*, the optimizer would run until a valid ($\delta r < \delta r_{\mathrm{TOL}}$) and optimal ($\Delta v < \Delta v_{\mathrm{estimated}}$) solution was found. The inputs for *Sniper Wolf* are directly taken from the outputs of *The Butcher* as described above and are:

1. IDs of the departure and arrival debris pieces.
2. Departure and arrival time ranges.
3. Estimated Δv and orbit type (long vs. short way solution, etc.).

The outputs of *Sniper Wolf* are:

1. Optimal Δv$_1$ and Δv$_2$ needed to initiate and complete the rendezvous maneuver.
2. Departure time and TOF corresponding to the optimal transfer found.
3. A flag corresponding to whether the optimizer was able to find a valid solution within the prescribed number of iterations.

```
Sniper Wolf - Pseudocode($P$)
while $\delta r > \delta r_{\mathrm{TOL}}$ && $\Delta v > \Delta v_{\mathrm{estimated}}$
PSO particles are randomly initialized using departure time, TOF,
  and 3-component velocity vector deviations as particle elements
for $1:1:iterations$
  for $1:1:particles$
    $r_1, r_2 \leftarrow$ Propagate starting and arrival debris pieces
    $\Delta v_1, \Delta v_2 \leftarrow$ Solve Keplerian Labert's problem using $r_1, r_2$
      if Periapse constraint is not met
        $J = 10^7$ // Particle receives a large penalty
      else
        Integrate s/c trajectory using the solution to Lambert's
         problem with the particle's velocity vector deviations
        $\delta r \leftarrow$ Compute drift vector magnitude
        $J = 0.001 * [10 * (\delta r - \delta r_{\mathrm{TOL}}) * u(\delta r - \delta r_{\mathrm{TOL}}) + \ldots$
         $\Delta v] \leftarrow$ Compute cost
      end
  end // end of particle computations
  Determine the best particle based on $J$
  Update "velocity" and "position" vectors for each particle
  if $J$ stagnates
    Reset half of the particles which have the highest $J$
  end
end // end of PSO iterations
$P_{\mathrm{Best}}, J_{\mathrm{best}} \leftarrow$ Find best solution among the particles
if new $J_{\mathrm{best}} >$ old $J_{\mathrm{best}}$
  $J_{\mathrm{best}}$ = new $J_{\mathrm{best}}$ and Solution = $P_{\mathrm{best}}$
end
end // end of the while loop
Return the best Solution
```

FIGURE 5. *Sniper Wolf basic pseudocode.*

TABLE 1. *Summary of all of the mission of the sequence The Fury.*

The Fury - Mission Summary			
Mission Number	Number of Transfers	Δv [m/s]	Propellant Mass [kg]
1	11	3978.62	4951.58
2	10	3929.44	4881.75
3	9	3316.76	3599.01
4	10	3923.04	4764.82
5	8	3408.66	3776.52
6	7	3984.92	4841.94
7	6	2258.99	2027.75
8	6	2765.71	2685.57
9	6	2313.49	2102.76
10	5	2785.08	2715.98
11	5	2508.40	2339.24
12	4	2506.96	2308.16
13	4	2787.16	2701.60
14	3	1467.72	1140.72
15	3	1251.12	941.42
16	2	1384.10	1049.31
17	1	2373.11	2106.17
18	1	1479.99	1134.23
19	1	2861.90	2759.22
20	1	2479.13	2239.80

4 Results

After running *Raiden*, the top-level optimizer, multiple times using the beam search method described in Section 2, the most optimal solution that was found, *The Fury*, consisted of 20 missions with a total of 103 transfers between debris pieces. The timeline of the mission is shown in Figure 6, where each segment corresponds to each individual mission (as defined by the number above it) launched as part of *The Fury*. The first mission in chronological order, Mission 20, starts on MDJ 23476.38. A summary of all of the missions and transfer Δvs and propellant mass is given in Table 1. The average required Δv and transfer time values per transfer for each mission are shown in Figure 7.

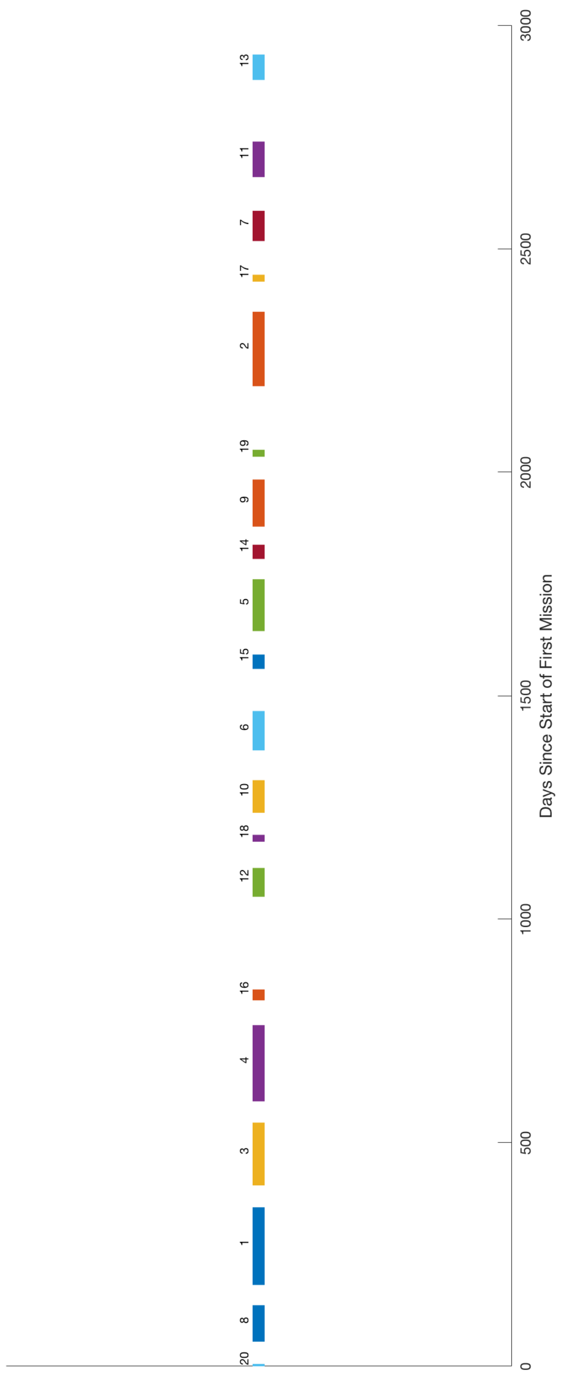


FIGURE 6. *The Fury - Mission timeline.*

The intention in the mission planning was to attempt to gather as many debris pieces as possible while keeping fuel requirements within the mass restrictions defined by the problem. Figure 7 shows that this objective was achieved with success for larger missions, such as Missions 1 through 8. These larger missions maintained higher efficacy in using the fuel that was allocated for their duration. However, as large missions are created by sorting debris pieces into similar orbital planes, the

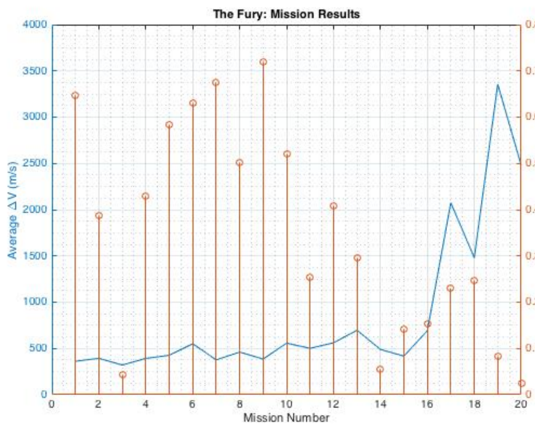


FIGURE 7. *The Fury averaged results.*

likelihood of being able to group the rest of the debris similarly decreases. There is a trend that appears, where an increase in the ratio of fuel cost per debris captured manifests as the number of debris available decreases to a small number (such as a single transfer). Some missions consisted of lower quantities of debris removed with larger fuel masses, such as Missions 17 through 20. Although this is an undesirable effect, it is inherently a part of the problem.

In order to remove larger quantities of debris in less missions, the conditions that determine how similar orbital planes are classified for groups of debris needed to be relaxed. The relaxation of these conditions can result in costly transfers that are high in fuel consumption. The higher fuel consumption at times would fall outside of the bounds of the mass capability set forth by the problem statement. The alternative approach was to take a larger number of missions in order to be capable of transferring between all debris pieces without exceeding the mass restrictions that govern the problem. The required mass ended up being lower than the maximum carrying capability available at times, but this was useful in lowering the cost function for specific missions directly.

Generally, the missions consisting of longer average times of flight between objects also required less fuel, as seen in Figure 7. However, outliers like Mission 3 exist that do not fit that pattern. This likely occured because not only are variation in semimajor axis and eccentricity not accounted for when determining the mission sequence, but the optimization methods used are not guaranteed to find global minima.

Future improvement to this method of solution could consist of isolating the high fuel single transfer missions in order to find a way to group them with nearby missions that fall within similar time-frames. Another aspect of the solution that could be analyzed further is the number of burns per transfer. The ability to increase the number of burns beyond two could yield more favorable fuel consumption requirements for most transfers, especially those with larger plane change magnitudes.

5 Discussion and Conclusions

In this paper the methods and results of the Astrodynamics Research Group of Penn State in the 9th Global Trajectory Optimization Competition were described [1]. An optimization strategy using a beam search clustering method (*Raiden*) was used to group orbital debris with similar orbital planes and RAAN to determine the order and timing of each visit. A Keplerian Lambert's problem solution (*The Butcher*) was applied to narrow down the search space for a high-fidelity optimizer using Particle Swarm Optimization (*Sniper Wolf*), which determined the best departure date and time of flight for each visit. While the method did result in four one-transfer missions with high fuel costs, all 123 objects were captured in 20 missions, giving ARGoPS a final score of 1512.60176793564.

Further improvements require more computational power to find a better mission sequence and lower individual transfer Δvs.

References

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

[2] H. Akeb, M. Hifi, and R. MHallah. A beam search algorithm for the circular packing problem. *Computers & Operations Research*, 2009.

[3] M. Pontani and B. Conway. Particle Swarm Optimization Applied to Space Trajectories. *Journal of Guidance, Control, and Dynamics*, 33(5):1429–1441, 2010.

[4] H. Ma, S. Xu, and Y. Liang. Global optimization of fuel consumption in J2 rendezvous using interval analysis. *Advances in Space Research*, 59:1577–1598, 2017.

Acta Futura 11 (2018) 117-123

DOI: 10.5281/zenodo.1139947

Acta Futura

GTOC 9: Results from the Military College of South Carolina and Des Moines Area Community College (team Citadel-DMACC)

Patrick Bass [*], Nathan Washuta,

Department of Mechanical Engineering, The Citadel, The Military College of South Carolina

Uriah Tobey

Department of Engineering and Mathematics, Des Moines Area Community College

Rafael Gonzalez

Department of Mechanical Engineering, The Citadel, The Military College of South Carolina

Abstract. This report presents the results, methodology, and lessons learned for the Citadel-DMACC group concerning the 9th Global Trajectory Optimization Competition. As first-time entrants, the focus was on developing a fundamental understanding of the project scope and problem approach, while utilizing this competition format as an opportunity for undergraduate research, and in this instance, specifically for a first-year engineering student. The optimization strategy that was utilized focused on the debris selection process. Initial debris target selection was centered on possible orbital intersections between an initial debris orbit and the instantaneous orbital projection generated from another target debris at a given time step. This report will discuss two target selection processes and their applicability to the problem. Lessons learned from how to approach the problem are presented here along with an analysis on the reasons behind the team's failure to provide a successful submission.

[*]Corresponding author. E-mail: pbass@citadel.edu

1 Introduction

The Global Trajectory Optimization Competition (GTOC) is an international competition open to industry, governmental agencies, colleges, universities, and even savvy space mechanics enthusiasts. The goal is to find unique and creative solutions for solving complex orbital mechanics and trajectory optimization problems wherein the solutions can be applied to real-world missions [1]. The ninth iteration of the competition was developed and organized by the European Space Agency. The competition problem statement outlined the need to deorbit 123 pieces of orbital debris in order to halt the Kessler effect, brought on by the explosion of a Sun-synchronous satellite. The Kessler effect is the idea that a collision in space will generate impact debris, which will then cause more collisions with more debris, and so on until it becomes impractical to send spacecraft into or through a given orbital altitude [2]. In the given problem, the challenge presented is to find efficient procedures for navigating a spacecraft between debris orbits in order to remove this debris while minimizing cost.

The authors were part of team Citadel-DMACC and it was the team's first time competing in GTOC. As part of this first-time competition entry, a first-year undergraduate engineering student was brought onto the team to promote research, develop student interest in orbital mechanics, and reinforce concepts learned in first-year coursework. During the month-long window to generate a solution, the team developed two methods for target selection, which are described in detail herein. As the problem statement laid out, the spacecraft would start at a selected piece of debris (target) and then maneuver to successive targets in order to ultimately clear all of the debris, while minimizing fuel usage. As first-time entrants, this team's initial focus was to reduce the complexity of the problem and focus on finding a pair of orbits that would come close enough to an intersection to allow for a direct orbital transfer. From this point, a corresponding set of orbital maneuvers was calculated that would move the spacecraft from the initial orbit to the target orbit and rendezvous with the second piece of debris, producing a valid submission for this competition. While this target selection process was successful, issues ultimately arose in the precision of position and velocity calculations, which could not be overcome in the competition time frame. This report will therefore focus on the successful orbit selection process and transfer maneuvers utilized, followed by discussion of lessons learned from the competition.

2 Problem Approach

For the competition, the team utilized single-core processor operations on various Windows-based platforms, using MATLAB as the software of choice. The choice of hardware focus allowed code to be written in tandem across a variety of desktop and laptop computers owned by the various team members. Utilizing MATLAB with single-core processing allowed for reinforcement of concepts learned in the first year Engineering Computer Applications course at The Citadel, while directing focus towards learning concepts of orbital mechanics. Future entries will explore how to leverage the parallel processing capabilities of the software to expedite the trajectory optimization process, while expanding beyond the scope of existing course concepts.

The method utilized for optimizing target selection was to keep the process relatively simple and focus on identifying orbital intersections solely between two targets. Due to the wide variety of orbital parameters for the 123 pieces of space debris in this competition, not all of the orbits come close enough to each other to constitute a direct intersection, defined as having a minimum distance of less than 100 meters. Once an intersecting pair of orbits was found, the spacecraft transferred directly from the initial target orbit to the final target orbit at the intersection point. Because the starting orbit of the spacecraft was chosen, the spacecraft orbit was assumed to initially match that of the debris on the first orbit so that the deorbit package could be applied to this first target before the first orbital transfer maneuver was made. The starting orbit was determined based on the first two intersecting orbits that were found with a reasonable for ΔV for transferring orbits. This initial maneuver would enable the spacecraft and the second target debris to have matching eccentricity, inclination, right ascension of the ascending node (Ω), and argument of perigee (ω) to progress from. Upon transferring to the final target orbit, the spacecraft would then have to undergo a series of additional maneuvers, detailed later, in order to physically rendezvous at the final target location.

Time-sensitive J2 perturbations on both the spacecraft and debris, due to the Earth's oblateness, made for a challenging approach to identifying possible initial and final targets. For each debris orbit, it was reasoned that at every instant in time, it is being perturbed by the Earth and therefore its Ω and ω are continually changing. This means that its orbital plane, and consequently its distance to any other orbit, are constantly varying. Therefore, the methodology began with selecting a starting target and calculating its trajectory for a given window of time. This would also constitute the starting position and trajectory for the spacecraft. For a given final target, a complete orbital path was calculated for a given time. When incorporating the J2 perturbations, the Ω and ω vary continually, requiring that the complete orbital path be recomputed each second. Possible intersections between this time-dependent, final-target, orbital path and the perturbed starting orbit trajectory were calculated and repeated until an orbital intersection was found.

Figure 1 is a highly conceptualized representation of the approach taken. In Figure 1.a, the black and blue curves represent the perturbed trajectories of the final target and the spacecraft/initial target, respectively. For each time step, the final target was considered to have an instantaneous orbital projection of possible intersection points for the spacecraft's trajectory, shown as a dashed red line. Figure 1.b, shows this again at a later time. It is not until the minimum cartesian distance between

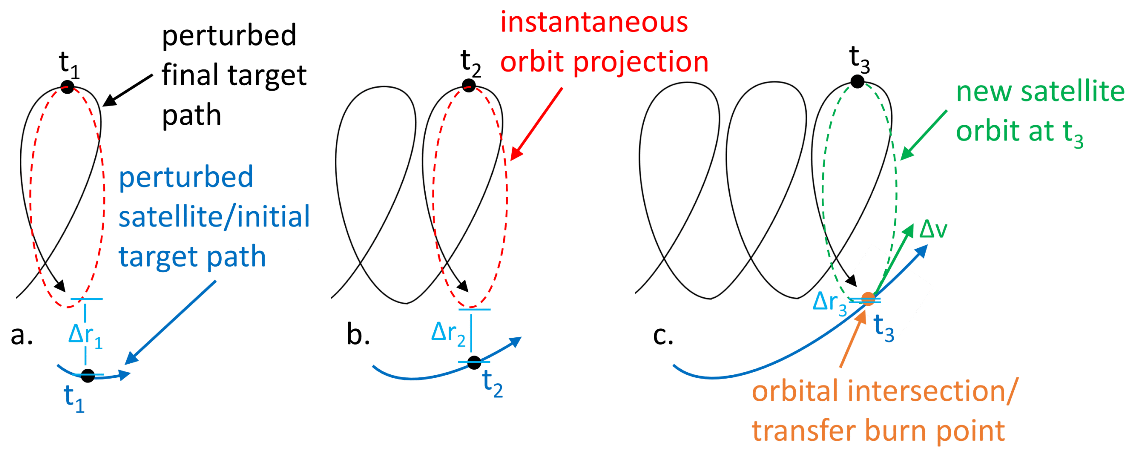


FIGURE 1. *Conceptualized interpretation of a perturbed orbital intersection between a spacecraft trajectory and the instantaneous orbit projection of the target debris*

the spacecraft and the final target's instantaneous orbit falls below 100 m, the threshold set in the competition's problem statement, that the orbits are considered to be intersecting (Figure 1.c).

The first challenge began in trying to determine if two given debris orbits could be considered intersecting. Two separate approaches were explored to determine the minimum distance between the initial and final target orbits before checking if that minimum distance fell below the intersection threshold. Figure 2 illustrates the first approach. In this method, the two orbital trajectories were discretized coarsely (approximately 1000 points each) and an initial guess was made as to which point on each orbit corresponded to the minimum distance. This guess on each of the two orbits is depicted in Figure 2.a as index m on the spacecraft path and index n on the target path. From this initial guess, the next point in each direction was considered (i.e. m-1 and m+1 on the spacecraft path) for each orbit. The cartesian distance between each of these 3 points on the spacecraft path and each of the three points on the target path was then calculated, forming a 3x3 local distance matrix, shown in Figure 2.b. The minimum of this 3x3 matrix was then found and the corresponding index was taken as the new initial guess. This process was repeated iteratively until the minimum of the matrix was found to occur at the center, resulting in a local minimum. Once the overall minimum distance between orbits was reached, the density of points on the orbits was refined by spreading 1000 points over what was previously 1 index on the course grid. This process was repeated until the minimum distance did not vary down to 16 significant figures (the highest default MATLAB precision). At that point, if the minimum distance was less than 100 m, the orbits were determined to have intersected and the analysis continued by calculating the necessary orbital maneuvers. If the minimum distance did not fall below the intersection threshold, two new target orbits were identified and the process was repeated. One drawback of this method is that it is possible to get a local minimum distance versus a global minimum and the particular local minimum is very dependent on the initial guess. Another drawback of this method is that even without incorporating J2 perturbations, the distance two orbits could theoretically have as many as four local minima. Once J2 perturbations are introduced, the minimum distance between orbits varies with each precession and it becomes necessary to analyze multiple orbital periods, each with their own set of local minima.

Incorporating J2 perturbations made it necessary to utilize a different approach for finding orbital intersections. As previously described, this approach involved generating a complete instantaneous orbit for the target orbit at each instant in time in order to find where the perturbed spacecraft trajectory intersected the target orbit as it moved along its own perturbed trajectory. Figure 3 illustrates the process to determine this intersection. For a given time, the Cartesian coordinates for the spacecraft and target debris were calculated based

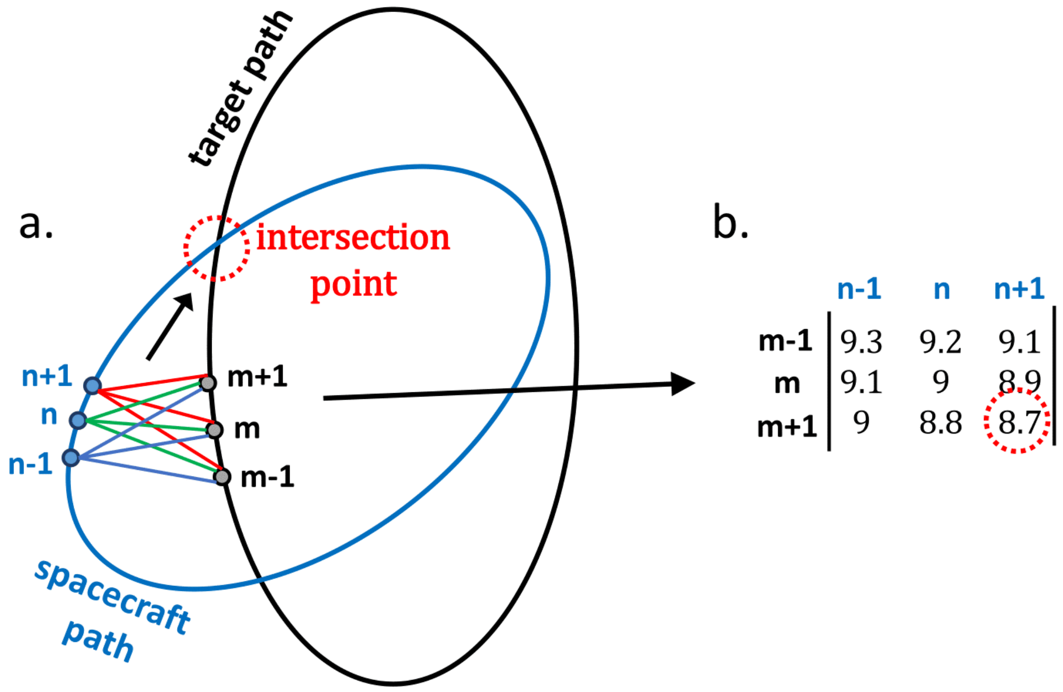


FIGURE 2. *Determining the orbital intersection point of two non-perturbed orbits*

on the given orbital parameters for each piece of debris. Using this target debris position as well as the instantaneous Ω and ω for that orbit, an instantaneous orbit was calculated and discretized into 1000 points. The difference between the spacecraft position at that time and each point on the instantaneous orbit was determined and is shown as a row in Figure 3.b. Each row of this matrix corresponds to the distance from the instantaneous location of the spacecraft at a given time to each point on the instantaneous target debris orbit. This distance calculation was performed for 20 orbits of the spacecraft, each split into 1000 discrete cartesian locations. The resulting matrix formed using this approach had a size of 20000 x 1000 elements. While this is considerably larger in size than the localized distance matrix described in the first approach, the minimum of this matrix represents a global minimum over this 20 orbit sample. This global minimum is a necessity for determining the minimum distance between perturbed orbits given the cyclical nature of the distance between these orbits. If a minimum point for this specified time window was found to fall below 100 m, then an intersection was declared, the mesh was refined at that intersection point with another 1000 points, similar to the previous method, and the process was completed again until a final minimum distance was determined. Figure 4 shows the minimum distance between the path of target 16 and the instantaneous orbit of target 37 over a given 20 orbit sample. It is obvious that a search for a local minimum could result in any number of incorrect choices. The global minimum over this range is identified in the figure and can be seen to cross the 100 m threshold for intersection.

If no intersection was found between the spacecraft and target orbits over this 20 spacecraft period, the analysis would move to the next set of targets and the process was repeated. If an intersection was determined to occur, then a ΔV calculation was conducted to see how fuel intensive the maneuver would be to move from one orbit to the next. It was found that compensating for apse line rotations was the major factor in fuel consumption. Upon completing this analysis for the given targets, it was found that the orbits of targets 16 and 37 intersected at $t = 23472.33$ days and that the fuel consumption for this maneuver (Table 1) allowed for enough fuel for the spacecraft to reach target 37 after

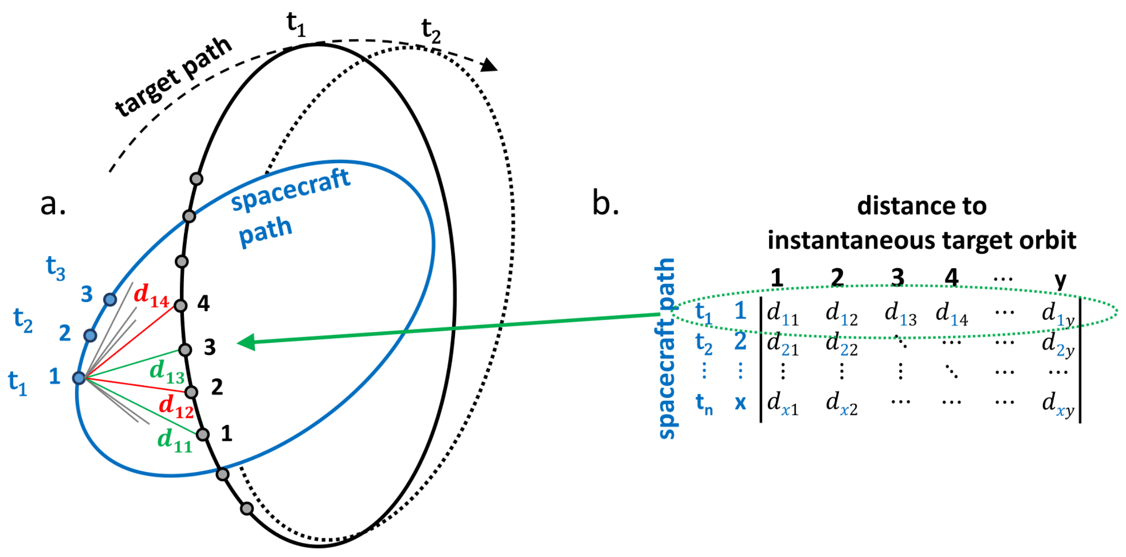

a.
target path
t_1
t_2
spacecraft path
t_3
t_2
t_1
3
2
1
d_{14}
d_{13}
d_{12}
d_{11}
4
3
2
1
b.
distance to instantaneous target orbit
spacecraft path
$\begin{array}{cc|cccccc|} & & 1 & 2 & 3 & 4 & \cdots & y \\ t_1 & 1 & d_{11} & d_{12} & d_{13} & d_{14} & \cdots & d_{1y} \\ t_2 & 2 & d_{21} & d_{22} & \ddots & \cdots & \cdots & d_{2y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \cdots \\ t_n & x & d_{x1} & d_{x2} & \cdots & \cdots & \cdots & d_{xy} \end{array}$

FIGURE 3. *Determining distance between the time-sensitive, perturbed spacecraft path and the time-sensitive instantaneous-orbit projection from the perturbed target path*

TABLE 1. *Mission Submission for Transiting from Target 16 to 37*

t (mjd2000)	x (km)	y (km)	z (km)	v_x (m/s)	v_y (m/s)	v_z (m/s)	m (kg)	ΔV_x (m/s)	ΔV_y (m/s)	ΔV_z (m/s)	event id
23467	-4149.3	-5770.5	-606.9	-535.5	1098.9	-7351.1	4560	0	0	0	16
23472.33	2680.5	5652.9	-3270.3	-452.8	-3592	-6580.1	2236	2104.3	-1053.4	31.84	16
23472.36	-2671.6	-5640.8	3267.3	445.93	3543.3	6481.7	2153	-7.533	-59.95	-109.7	-1
23473.33	-2597.9	-5857.6	2927.5	245.84	3275.1	6771.2	2074	4.078	54.488	112.7	37
23478.33	8516.9	3161.1	6306.3	2253.3	6411.9	-3208.8	2044	0	0	0	37

two additional phasing burns.

Given the spirit of the competition, optimizing the orbital maneuvers that the spacecraft would undergo when moving from target to target is of clear importance. However, as will be discussed in further detail below, errors in calculating spacecraft position and velocity propagated with respect to time and these inaccuracies could not be overcome in time for a successful rendezvous submission. This prevented full vetting and optimization for the intended maneuvering procedure, when transiting from one target to another, so these maneuvers will only be mentioned briefly in this report. Upon selecting targets, as detailed above, and making the commensurate burn to move the spacecraft onto the target's orbit, two more burns were reasoned to be needed for moving the spacecraft into a rendezvous position with the target debris. A phasing maneuver was initiated with the final phasing maneuver occurring 15 orbits after the initial burn was conducted [3]. The initial and final burns were performed at the target's perigee point. This procedure consisted of 3 total burns for moving between two targets, consisting of: 1) RAAN/inclination/AP burn from starting target to final target orbit burn, 2/3) initial and final phasing burns. For the targets selected (transiting from target 16 to 37), these maneuvers resulted in a total ΔV of 2.6145 km/s and a consumption of 2522.0 kg of fuel.

3 Lessons Learned

Despite the inability to overcome errors in position and velocity calculations in order to submit a successful solution, a variety of lessons were learned throughout this process and these lessons will inform the team's approach to future competitions. First, during the solu-

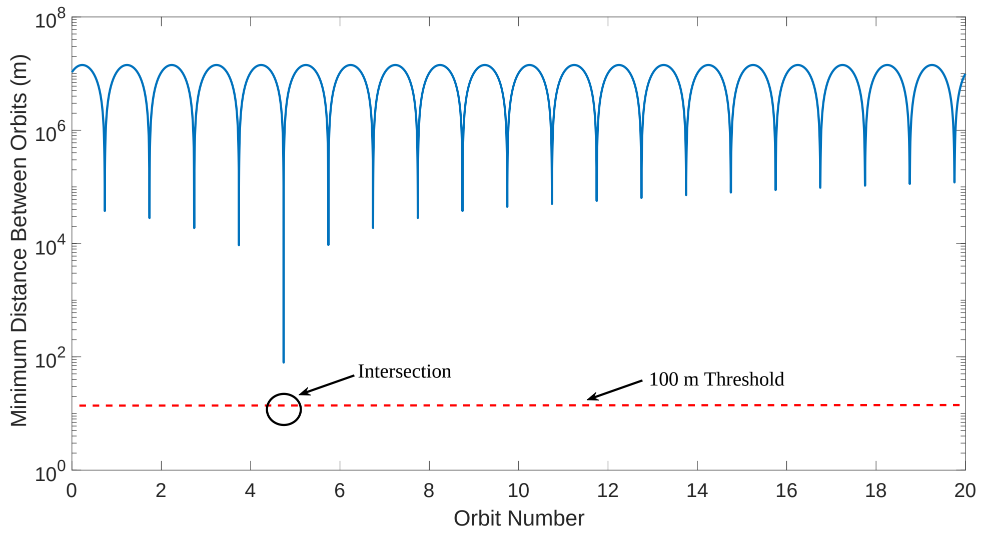


FIGURE 4. *Minimum orbital distances between debris 16 and 37, as a function of debris 16's orbit number, with an indicated intersection point between orbits 4 and 5.*

tion process, the team was able to work through most of the issues for making simple maneuvers between targets, but in the end, propagating errors with the position/velocity vectors were found. The code used was repeatedly refined and re-compared to the charts provided in the problem statement, but the differences could not be overcome in the allotted timeframe and therefore, no solution was submitted successfully. At best, the team could match 6-7 significant digits with the numbers provided in the problem statement which had 17 significant figures. When submitting possible solutions, it was found that the propagating error resulted in position differences of 300 m between the calculated and actual positions. After reviewing procedures graciously posted by one of the other competitors, it was found that the issue with the calculations was with the level of precision that was being rendered through MATLAB when solving the set of ordinary differential equations that describe the motion of the spacecraft [4]. For future competitions and research, understanding how to effectively use the precision options for MATLAB's ordinary differential equation solvers is paramount to obtaining accurate solutions. The second lesson learned stems from an understanding of the equations provided in the problem statement for determining the Cartesian vectors of the spacecraft/target position and velocity. It was initially incorrectly assumed that these relationships provided the same results as if positions were calculated in the geocentric equatorial frame. This error was discovered at roughly the half-way point of the competition and meant a near total rework of the code produced. The last lesson to be discussed stemmed from a passage in the problem statement, where it is mentioned that the spacecraft feels the full J2 perturbations once it leaves a given target. From this statement, it was incorrectly assumed that the transit time of the spacecraft was the only time that perturbations were necessary to consider. It was found that the intent of the statement was quite the opposite and meant that perturbations had to be considered at all times, with the effects needing to be calculated separately when the spacecraft was away from any target. In future competitions, these lessons will be taken into consideration and will form a basis from which to implement more complex concepts into the approach to a solution.

4 Conclusions

In all, this was a challenging competition that allowed for creativity and flexibility in solving a complex, open-ended problem. The competition prompted many interesting discussions and brain-storing sessions, as well as a number of teachable moments, for both the student and faculty members. Had the team been able to overcome the precision errors that were encountered, the target selection processes detailed herein would have been expanded to transiting between multiple targets versus the bulk of our endeavors being solely focused on moving between two. As an opportunity to promote undergraduate research, this competition has been a great success, by reinforcing and giving engineering context to concepts delivered in first-year coursework. This experience was a great opportunity for fostering the student's academic interests and he is excited about contributing to future GTOC iterations.

References

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

[2] D. Kessler and B. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. *Journal of Geophysical Research: Space Physics*, 83:2637–2646, 1978.

[3] H. Curtis. *Orbital Mechanics for Engineering Students*. Elsevier Butterworth Heinemann, 2013.

[4] M. Hallmann. Sharing code - position and velocity vectors. `https://kelvins.esa.int/gtoc9-kessler-run/discussion/368/`, 2017. Online; accessed 4-May-2017.