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

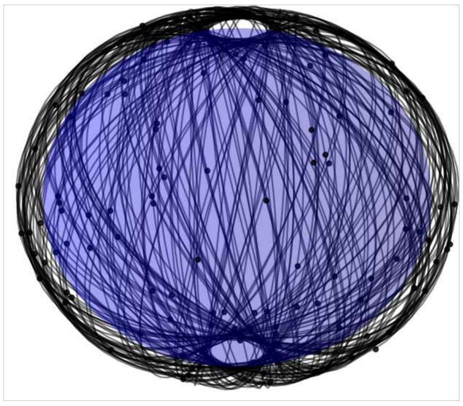

FIGURE 2. *Visualization of the M orbits of the M debris at some fixed epoch. The earth is visualized as the transparent blue sphere. See also* `https://youtu.be/zvxZx-QnqQ0`

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

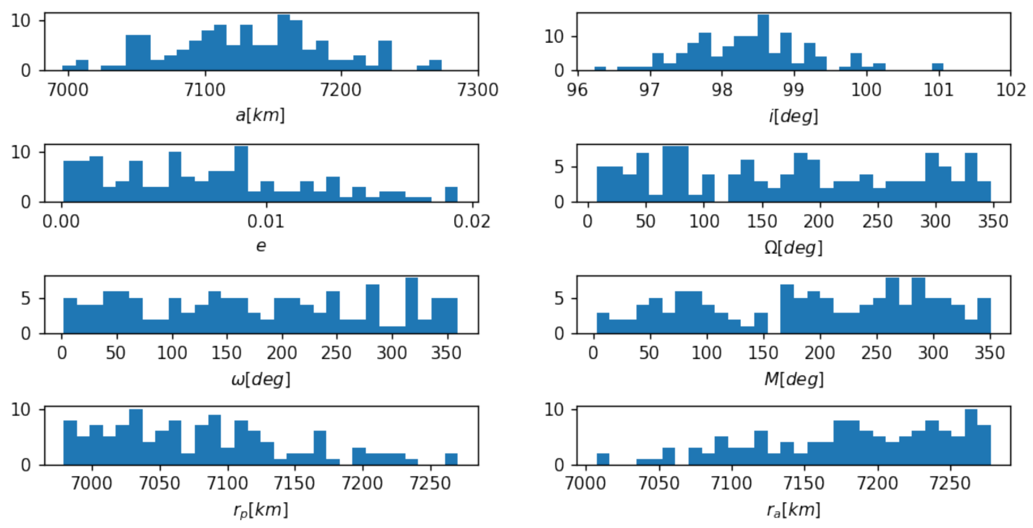

FIGURE 3. *Histograms for the various orbital parameters of the M debris orbits.*

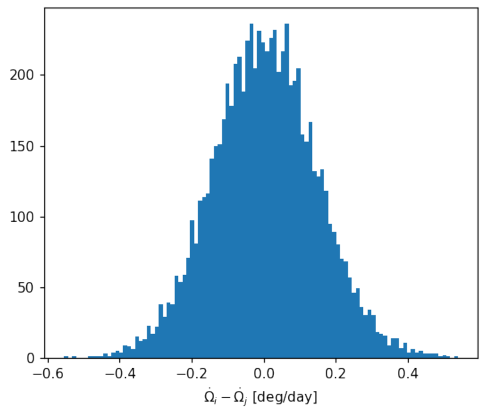

FIGURE 4. *Histogram for all differential RAAN drifts between pairs of debris.*

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

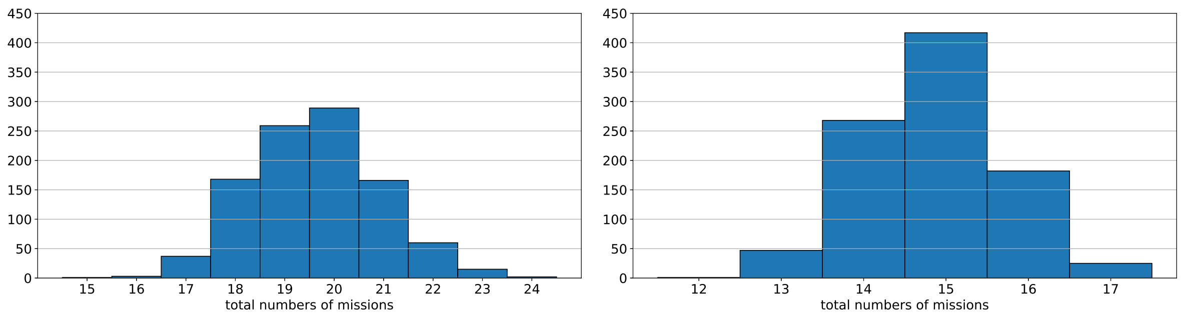

FIGURE 5. *Score J_S (total numbers of missions) achieved by 1000 different runs of (left) incremental greedy search and (right) incremental BS with $bw = 30$.*

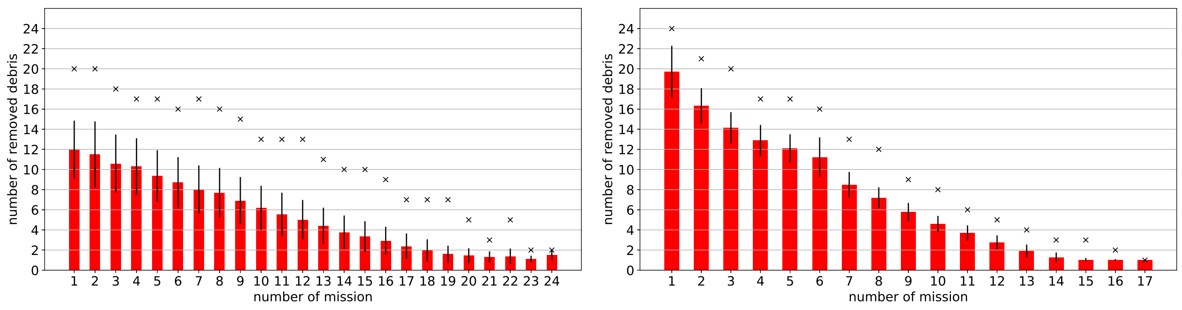

FIGURE 6. *Number of debris removed per mission in 1000 different runs of (left) incremental greedy search and (right) incremental BS with $bw = 30$. The black crosses mark the biggest positive outliers, i.e. a trajectory that would be most likely part of the final solution.*

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?

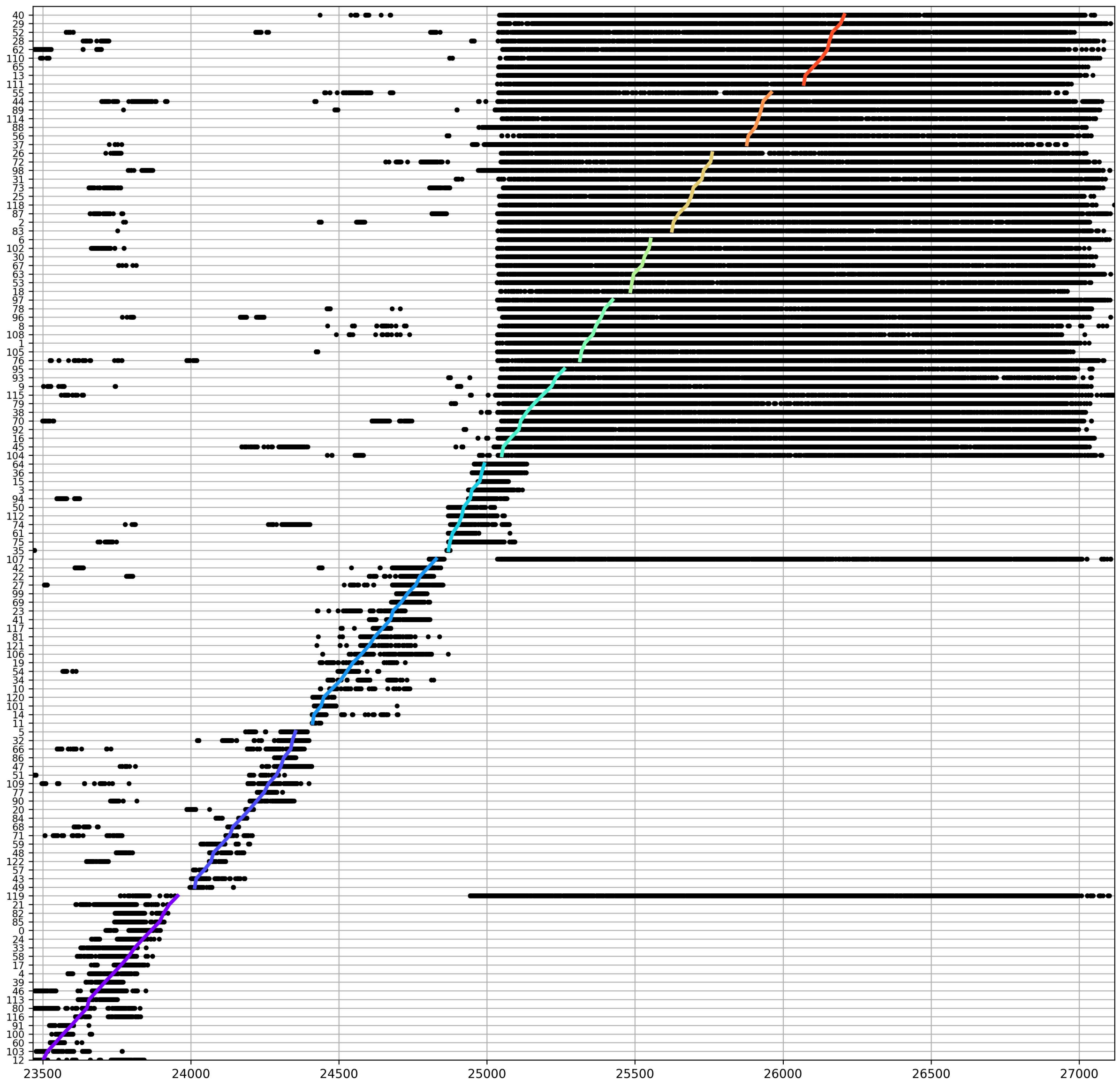

FIGURE 7. *Visualization of the full solution found by the hybrid strategy together with all possible debris removals generated for its construction. Vertical axis: id of removed debris, ordered in sequence of removal by the found solution. Horizontal axis: epoch in MJD. Each black dot marks the epoch 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 $U = \{1, 2, \dots, n\}$ be a set of n elements, k be an integer and $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ a collection of subsets $S_i \subseteq U$ for $i = 1, \dots, m$, $m > k$. 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.