GTOC 9, Multiple Space Debris Rendezvous Trajectory Design in the J2 environment

Marcus Hallmann,* Markus Schlotterer, Ansgar Heidecker, Marco Sagliano
Frederico Fumenti, Volker Maiwald, René Schwarz

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

May 11, 2017

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 trajectory design.

1 Introduction

The paper is organized as follows: section 2 recaps the relevant figures of 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.

*E-mail: Marcus.Hallmann@dlr.de

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:

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

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: $m_0 = m_{dry} + m_p + N m_{de}$, with $m_{de} = 30\,\text{kg}$. The α parameter is set to be $2.0 \cdot 10^{-2}\,\text{MEUR/kg}^2$.

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 deploy-

ment 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 $r_m = 6600\,\text{km}$.

The spacecraft dynamics is described by the following set of Ordinary Differential Equations:

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

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):

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

It can be seen that the ascending node is the orbital element which encounters the most variations caused by J_2. That will have an impact on our 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 we have to solve 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 our case we have 123 debris to sort for the best sequence (which gives 123! permutations) and than we have to chop them into n missions, which causes another dimension. 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 how we solved the combinatorial part of the problem.

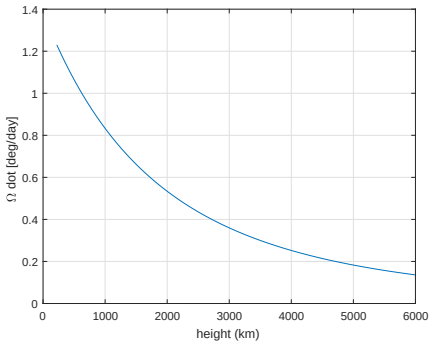

Figure 1: $\dot{\Omega}$ over height

Before we look into the transfer we analyze the design space to get some reasonable boundaries for our design variables like transfer time, delta v range, number of missions we need 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 2 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° to 102° and the orbital height (or $a - r_{eq}$) goes from 600 km to 900 km. These elements do not change over

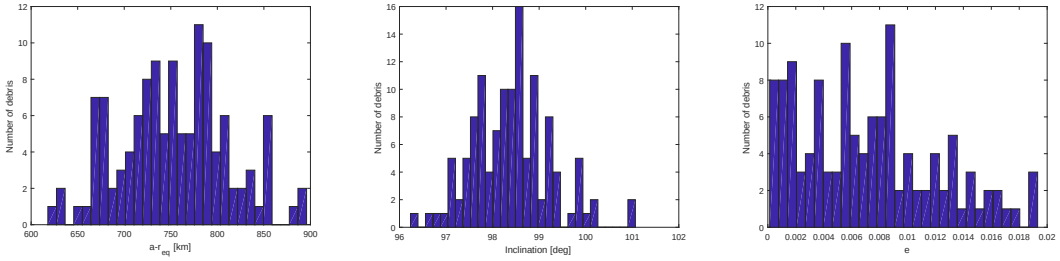

Figure 2: Debris orbital elements

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:

$$\Delta V_{inc} = 2V \sin((i_A - i_B)/2)$$
$$\Delta V_{sma} = \sqrt{\mu/r_1}(\sqrt{2k/(1+k)} - 1) + \ldots$$
$$\sqrt{\mu/(r_1 k)}(1 - \sqrt{2/(1+k)})$$
$$\Delta V_{AB} = \Delta V_{inc} + \Delta V_{sma}$$

Where k is the ratio between a_A and a_B, assuming circular orbits. With that equations we can set up a cost matrix which gives us the cost for going from one debris to the next debris, ignoring the phasing in true anomaly and right ascension, and we can apply a genetic algorithm implemented in *Matlab* to find the optimal route. Figure 3 shows the result of one run with 500 populations and $1 \cdot 10^5$ iterations. The total distance we get is around 2654 m/s, so an average ΔV of 21.7 m/s for one transfer. In theory and with no constraints on the mission time, this result would equal to a J value below 100 MEUR. In practice we have to fulfill the 8 years mission time constraint and the 25 day transfer time constraint.

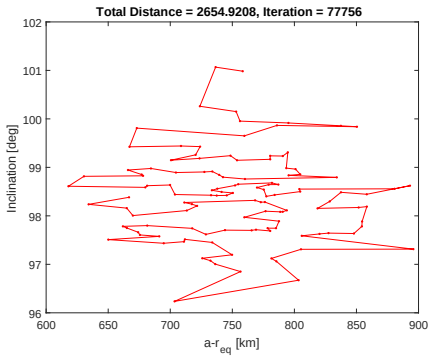

Figure 3: Optimal Path in the i and a TSP

That forces us to invest some reasonable ΔV 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

Let us compare the ΔV for both cases. If all other elements besides Ω are the same we can use a similar equation like we used for the inclination change for the direct plane change:

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

Like the inclination change its a quite cost intensive maneuver. If there is enough time

available an indirect transfer maybe cheaper. An important figure to look at is the change in right ascension for the debris which is plotted in figure 1 over height (with an inclination equal to 98 deg). Out of that we can see that the orbits are drifting between 1.2 deg/day and 0.2 deg/day. Depending on our initial height value we can get a differential drift around 0.5 deg/day. Assuming a 20 day transfer time we could overcome a delta of 10 deg in Ω. For the indirect change we perform two Hohmann like transfers, the first one to reach the desired drift orbit, the second one to get the semi major axis of the arrival debris.

Let us compare some use cases to see if the drift approach is better or not. Our orbital elements for both debris are: $a = 600\,\text{km} + r_{eq}$, $e = 0$, $i = 98\,\text{deg}$. So we only consider a change in Ω. On Figure 4 we can see that the

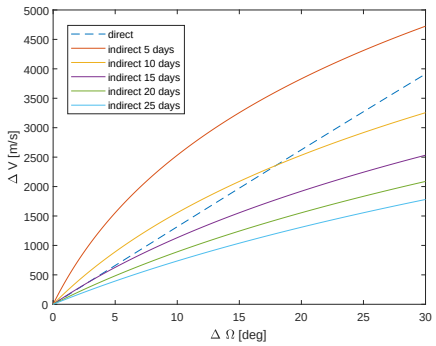

Figure 4: ΔV over $\Delta\Omega$

ΔV needed for a direct change goes nearly linear with $\Delta\Omega$ (dashed line). The other curves in the figure are representing an indirect transfer for different transfer times (5:5:25 days). We can conclude 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 we may save some ΔV cause of different semi major axis and inclination of the departing and arrival debris. The inclination change we can either perform before or after the drift change maneuvers. And the choice has an impact on the required ΔV for the drift change as it can be seen when looking at equation 3, cause its a function of inclination. With that thoughts we have a good set up for our combinatorial problem, which will be discussed later.

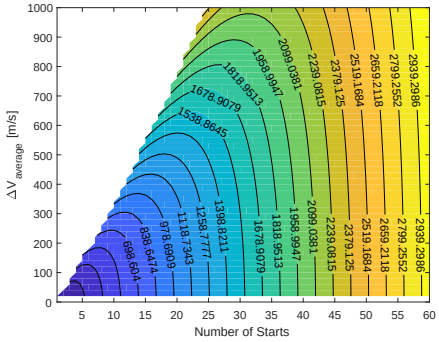

Figure 5: J over n and $\Delta V_{average}$

The next interesting question to look at is how many missions we may need (do we stack one launcher full as possible) and what's the J looks like. Assuming a range of different average transfer ΔVs and number of missions(n) we get figure 5, which shows the J function in MEUR. It can be seen that we should not use the maximum propulsion available, cause if we analyze the J value for an average ΔV of 300 m/sec, the J value would be for 9 starts 904.1 MEUR, 12 starts 827.6 MEUR and after that it goes up again. Although we would have 12 times the base cost for the launcher instead of only 9 times, the used propellant mass goes in quadratic. And if we use the total allowed 5000 kg we are not optimal. So we have to reduce the total allowed fuel or ΔV per mission by 10 % to 20 %, depending on how many missions we need. The issue is that we do not

know that before hand.

4 Combinatorial Problem

In section 3 we set up our strategy how to tackle the problem. We can estimate the cost or ΔV to perform a debris to debris transfer. The issue is, the problem is time variant, cause our cost matrix depends on at which epoch we perform the transfer. Or using the TSP syntax its not a city to city routing problem, its a boat to boat one, where the boats are sailing around.

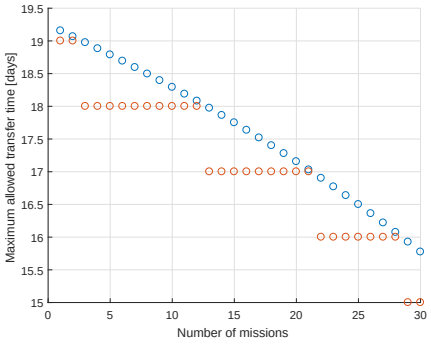

Figure 6: Maximum transfer time over n

Before we go into detail of the graph implementation we have to look a bit deeper into the handling of the transfer time. We have seen for the drift change maneuver, it would be better to have a large transfer time available. So it looks obvious to use the maximum allowed 25 days. But if we use that for all of the transfers, we are ending up having a total mission time which is larger than the allowed 8 years. And that is even dependent on how many missions we need. Figure 6 shows the maximum allowed transfer times over the number of missions we need, which is again not known before we have solved the graph problem. In fact we can only use average transfer times between 15 and 19 days. In that case the next question is, if we let the transfer time be a design variable in the combinatorial problem or if we keep it to an average fixed value. We decided to keep it fixed, cause the permutation space is large already. That approach allows us also to use a look up table for all possible transfers with a 1 day grid size, which is precalculated and loaded into the graph algorithm. In the cost matrix we can also 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 we tried to use the same genetic algorithm, which was delivering good results for the inclination-sma routing problem, on the time dependent one, but it didn't brought any good results. Instead we developed a graph algorithm which uses a certain beam width.

Each mission can be represented as a graph or tree (see figure 7). For the first mission we have 123 nodes or debris as an option to start from. Keeping in mind that the J-function doesn't give any penalty at which debris we start our mission, it's a free design parameter. So our initial beam width would be 123. Than we can calculate for each of that 123 debris 122 possible debris transfers, cause each debris should be only visited ones. That gives us 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
- Greedy Search

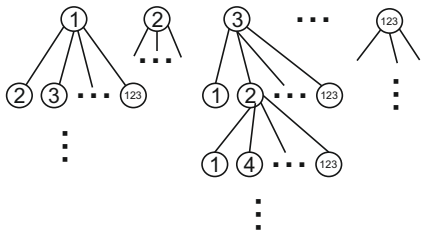

Figure 7: Mission Graph

The Depth First Algorithm travels along the left or right side of the tree. In our case it would just visit the debris either in the sequence 1:1:123 or 123:-1:1. We could add some backtracking if we get stuck somewhere or if we apply some heuristic, like average ΔV is getting to worth 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 our 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. Thats the first method we 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 we still have to eat them, cause we have to collect the entire set and not only a subset. And that's something we 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 we can apply a beam search. Instead of only traveling along one path we select k best options and take them with us. Depending on the beam width k the computational time of course grows in that case. In our case we have an initial beam width of 123 for the first mission. On the next level we would have $123 \cdot 122 = 15006$ maximum beam width and so on. We limited our beam width for the first runs to 2000 and got already good results for J around 1000. The entire idea behind that method is, that we can not look into the future, so we have to take also some bad paths with us to hope that they turn into the golden paths in the end.

At each level we check each possible solution 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 we would have the same node, in that case node 2, to start from and the left over debris-set is the same. So we only take the best sequence out of that two, because our beam width is limited and we want as many different permutations to find the golden path.

The graph algorithm is implemented in *matlab* and takes roughly 1 hr computation time on a Intel Xeon CPU E3 3.50GHz, with a beam width set to 20000. When we run the tree after the first mission we have the advantage that the left nodes are getting less and the computation time goes down.

With the 5000 kg propellant we can roughly achieve a maximum ΔV of $5000\,\mathrm{m/s}$. But with the results we had from our qualitative J-function analysis we set our maximum to 4500 m/s and we also made runs with lower values. For the first mission we were able to perform 23 transfers. But the beam width at that point was only around 10 to 20. So in that case we don't have enough permutations for the next

missions. So instead of taking the maximum transfer solution we took the one with a higher beam width. Our algorithm is setup 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.

Our final sequence will be discussed in the Results section 6

5 Transfer Problem

Before we solve the debris to debris final transfer we re-optimize the sequence coming out of the beam search algorithm. Like we discussed in section 4 already we kept the transfer time for all transfers to a fixed value. That will be 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. Our design variables are the transfer times. The upper bound is set to 25 days, constrained by the problem statement. The lower bound to 1 day, cause we may need some time for the final phasing of the true anomaly, which we have ignored completely so far. In the combinatorial part we used a fixed grid size for the transfer time (e.g. always 17 days for the first 6 missions and than 19 or 20 for the remaining, depending how many debris is left to collect). We also have to introduce an inequality constraint that the sum of the transfer times is not larger than the old sum of the fixed transfer times (That means we can only shift some transfer time from one transfer to the other). For the re-optimizer we will not use a look up table for the ΔV, we will calculate it during the fmincon call. That gives us the advantage of having a real value for the transfer time. With that approach we saved between 10 % and 25 % ΔV per mission. For each transfer we know the following information:

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

We used again *matlab fmincon* with an interior point method to tackle that problem. The control parameters are the times between maneuvers and the 5 maneuvers itself in Cartesian form. The cost function is quite easy in our case, it's just the sum of all 5 ΔVs we applied. We have 3 deep space maneuvers, one at departure and one at arrival. The more demanding part is the constraint function we pass to *fmincon*. Here we integrate the equation of motion between the maneuvers until we reach our final state. Than the final state should equal the arrival debris state at that time. We have a global parameter in order to activate or deactivate the constraints, and we can choose between the cartesian state vector, keplerian elements, or a mixture, or a subset. Another inequality constraint is also in the game, because we have to take care that the sum off all transfer times between maneuvers is not larger than the transfer time we got out of the tree, otherwise we would mess up the following transfers.

When using an ode-solver in *fmincon* we have the issue that integration errors we get from the ode solver may disturb the Jacobian or Hessian. And that was the case for the first runs we performed and we had to investigate which ode solver brings reasonable results. We came to the conclusion that a fixed

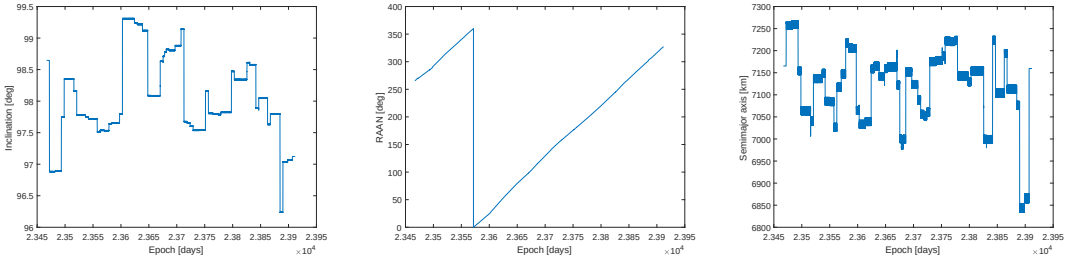

Figure 8: Evolution of orbital elements for mission 1

step size is more stable than a variable step size solver. In the end we used a RK8 implemented in c++ with a step size of 50 sec(the RK4 needed 1 sec step size), cause the *matlab* implementation was to slow. An interesting observation was also when using the debris dynamic model first and rerunning the optimizer with the spacecraft ode we achieved faster and better results.

For the initial guess we set the first and last two maneuvers to the Hohmann like maneuvers coming out of our drift strategy. The maneuver in the middle is set to 0. We didn't touched the inclination change or phasing change and left that for the optimizer to solve. The first and last transfer times where set to a half orbital period. And the remaining two transfer times where chopped up equally (kind off mid course maneuver).

What we found out is that scaling our state vector x, the constraints and the cost function all close to 1 is crucial for success. Although you could 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 we added scalar the inclination change and the drift maneuver. In practice they can be combined and the optimizer seems to have taken care of that.

6 Results

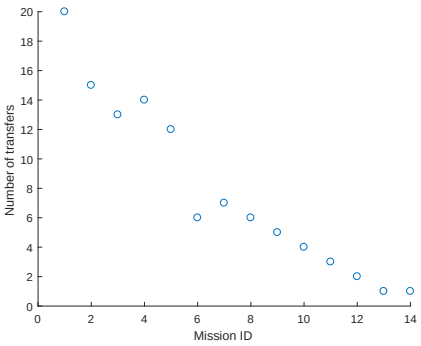

Figure 9: Mission Graph

Our submitted solution had a total of $n = 14$ missions and a performance index $J = 949.85$ MEUR. Figure 9 shows the number of transfers we have 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, that we have the good sequences in the beginning and we obtain bad once in the end. One option to improve that may be to select some feasible continuation beams randomly. In the transfer problem we may use a multiple shooting method to get rid off our ode-integration issue.

References

[1] Dario Izzo. Gtoc 9 poblem statement. 2017.