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, Germany

Outline

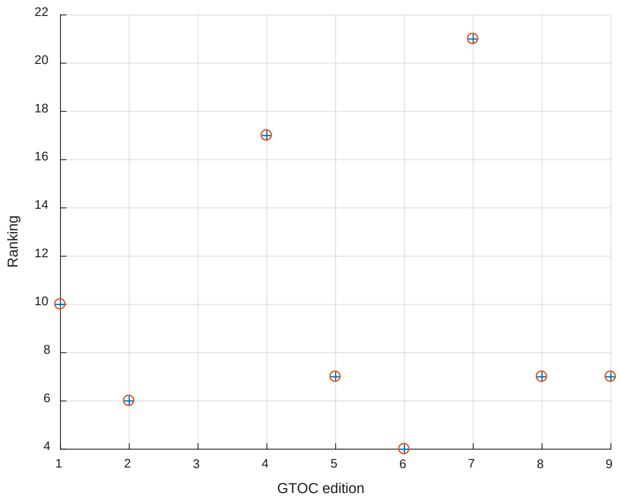
- DLRs history in GTOC
- Recap of the Problem Statement
- Overall strategy
- Combinatorial Problem
- Transfer Problem
- Results
- Conclusions

DLRs history in GTOC

- DLR has been part of GTOC from the beginning with an exception for the third GTOC

- The main motivation for us is to benchmark our own tools and code

- Excellent platform for networking in the space optimization domain (also with people from other domains)



Problem Statement

- Multiple Space Debris Rendezvous
- Scenario with n missions to collect a set of 123 debris pieces, distributed in the SSO region
- Cost function:
    - Base Cost per launch (increasing during competition time) plus used propellant mass square
- Launch Injection Parameters do not impact the cost function
- Five impulsive manoeuvres are allowed from debris2debris transfer
- Several constraints (max transfer time, min pericenter, mission time window)

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

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

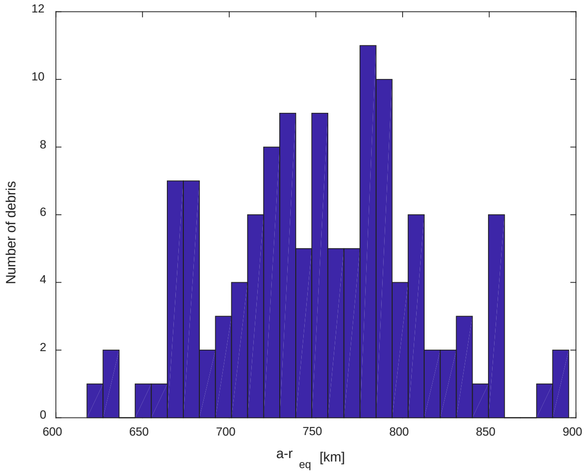
$$\dot{\omega} = \frac{3}{4} J_2 \left(\frac{r_{eq}}{p}\right)^2 n \left(5 \cos^2 i - 1\right)$$

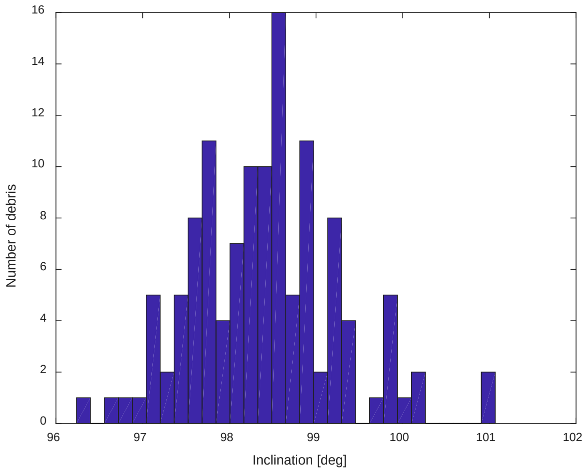
Overall Strategy

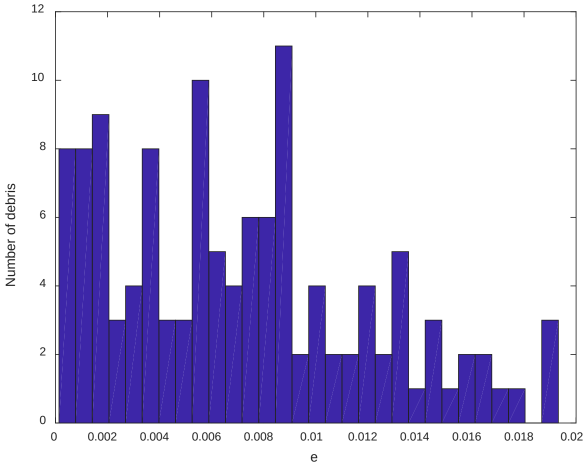
- The problem can be classified as a Time Dependent Traveling Salesman Problem (TDTSP)
- Nested Optimal Control Problem for the Transfer
- Combinatorial Permutation Space is huge
- On the Base Cost we don’t have any influence, besides working fast
- A fast transfer cost evaluation is needed for the combinatorial part

Overall Strategy

- How are the debris pieces spread out regarding eccentricity e, inclination i and semi major axis a ?







=> Nearly Circular, inclination ranges from 96° to 101° and the orbit height ranges from 600 km to 900 km

Overall Strategy

- If we only consider a change in inclination i and semi major axis a the problem can be treated as a TSP
- The cost in that case is the sum of the inclination change ΔV_{inc} plus the Hohmann transfer ΔV_{sma}

$$\Delta V_{inc} = 2V \sin((i_A - i_B) / 2)$$

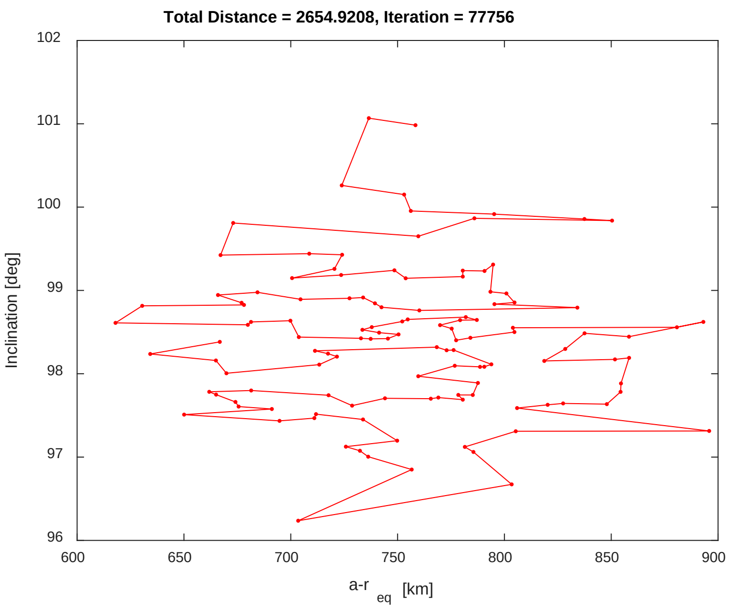
$$\Delta V_{sma} = \sqrt{\mu / r_1}(\sqrt{2k / (1+k)} - 1) + \sqrt{\mu / (r_1 k)}(1 - \sqrt{2 / (1+k)})$$

- Where *k* is the ratio between the two semi major axis a_A and a_B (assuming circular orbits)

$$\Delta V_{AB} = \Delta V_{inc} + \Delta V_{sma}$$

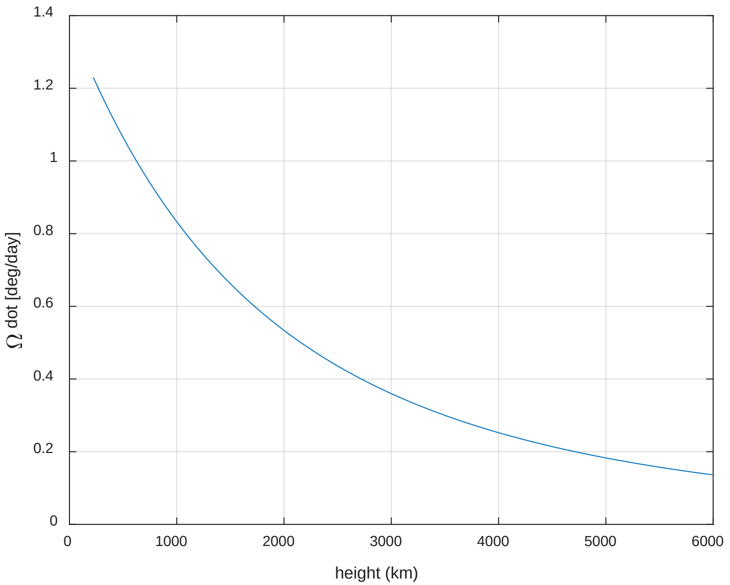
Overall Strategy, inc sma TSP

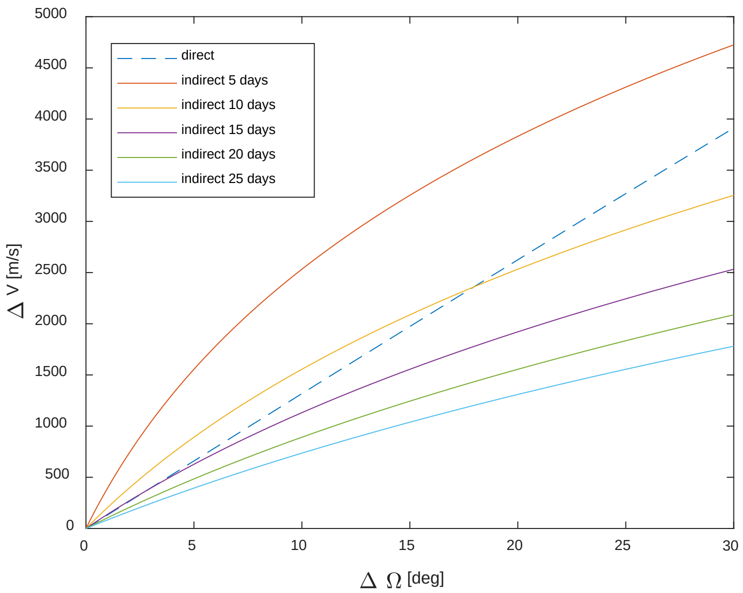
- With the above equations we can set up a cost matrix 123x123, which gives the distance in terms of velocity from debris A to debris B
- We used a genetic optimizer in *matlab* to find the shortest path between all 123 debris
- Population was set to 500 and 10^5 iterations were performed
- The total distance we get is around 2654m/s, so an average ΔV of 21.7m/s for one transfer



Overall Strategy, Change in Right Ascension

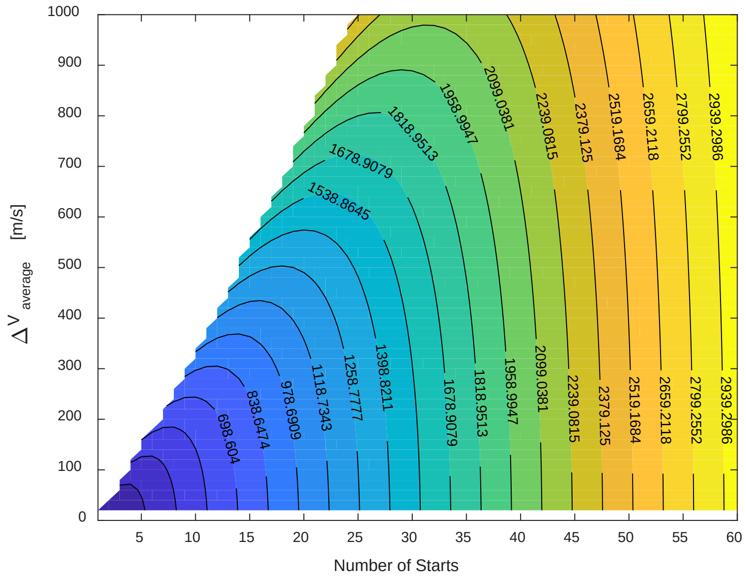
- The change of the Node Ω can be done in two ways:
  - A direct plane change
  - An indirect plane change via a change in semi major axis





Overall Strategy, How many missions?

- An important question to answer is: How many missions we need?
- Do we stack the launcher as full as possible?
- It can be seen that we should not use the maximum propulsion available
- If we analyze the J value for an average ΔV of 300m/sec:
    - 9 missions: 904.1MEUR
    - 12 missions: 827.6 MEUR

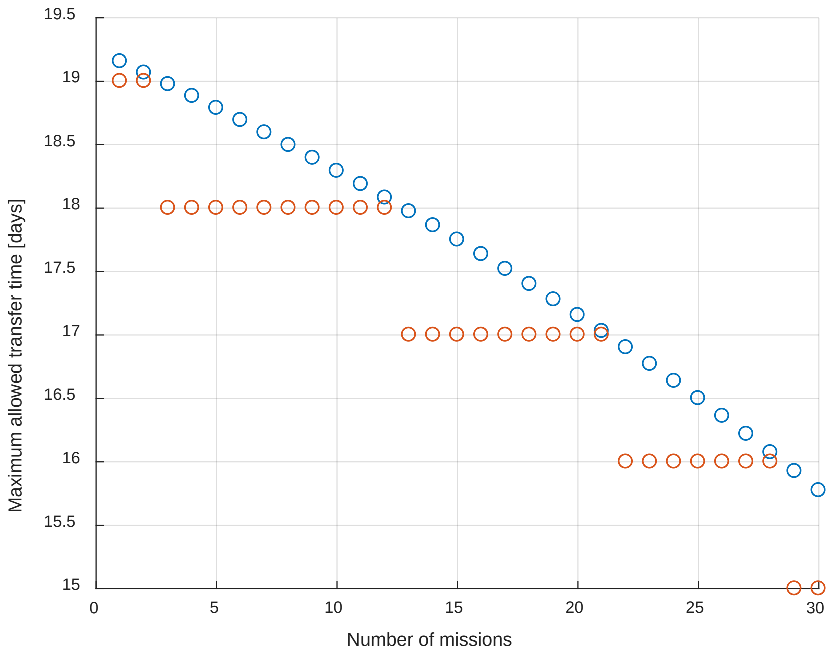


- So we have to reduce the total allowed ΔV per mission by 10% to 20% depanding on how many missions we may need

Combinatorial Problem

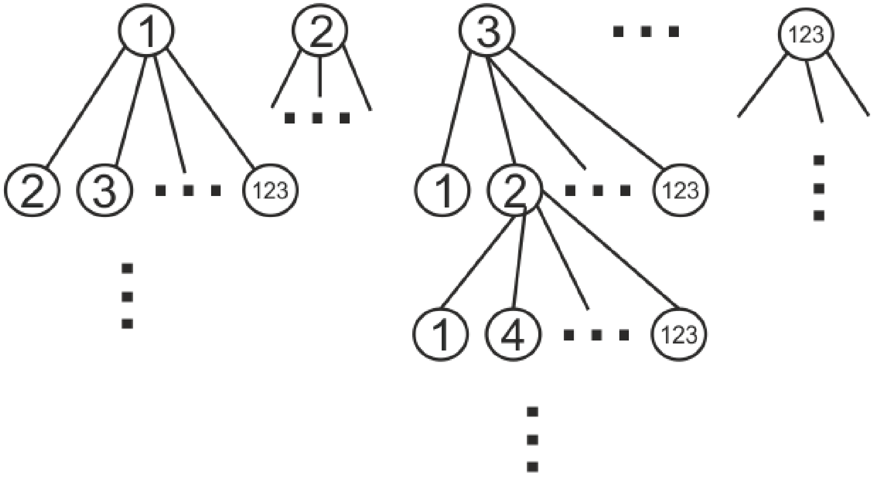
- If we use the allowed 25 days for the transfer we ending up with a total scenario time larger than the allowed 8 years

- In fact we can only use average transfer times between 15 and 19 days



Combinatorial Problem

- We used a graph approach which uses a certain beam width
- For the first mission we have 123 nodes or debris as an option to start from
- For each node the remaining transfers are calculated
- The Transfer Cost was precalculated and stored in a m-file for discret transfer times (14 days … 25 days)



Combinatorial Problem

- A pure Greedy search(depth first along the lowest ΔV) brought results in the range of $J = 2500\ MEUR$
- A full Breadth Search is not possible cause the permutations are to many
- Beam search is like a mixture between Greedy and Breadth search, instead of only looking on the best route, we take up to 50000 routes with us down the tree
- Beam search with a Beam Width between 50000 and 20000 delivered our committed solution

Combinatorial Problem

- After some levels in the graph we would get equal solutions in the tree
- e.g. the sequence 34-12-110-5 is equal to sequence 34-110-12-5
- We introduced a filter to only allow unique solutions to be passed to the next level
- That part is crucial, cause the goal is to get rid of the greedy effect (bad cookies stay over) and we can only improve if we have permutations

Combinatorial Problem

- The tree saves the beam after each level (or number of transfers performed)

- With that files we can enter the next mission

- The issue is when we use the file with a high number of transfers we have several drawbacks:
  - Fuel consumption is high, which may not result in an optimal J-value
  - The beamwidth gets low for the next mission, cause we only have a small beamwidth at that lvl

beamWidth=20000 NumTransfers=10
best_dv =1.7574km/s
beamWidth=20000 NumTransfers=11
best_dv =2.4213km/s
beamWidth=13850 NumTransfers=12
best_dv =2.695km/s
beamWidth=5114 NumTransfers=13
best_dv =2.9421km/s
beamWidth=753 NumTransfers=14
best_dv =4.1597km/s
beamWidth=37 NumTransfers=15
best_dv =4.625km/s
beamWidth=6 NumTransfers=16
best_dv =4.7616km/s
beamWidth=2 NumTransfers=17
best_dv =4.8884km/s
beamWidth=0 NumTransfers=18

Combinatorial Problem

MissionID	NumTransfer	Deb	TT	validBeams	NextBeamWidth		NodesLeft
1	20	21	17	42	4284		102
2	15	37	17	37	3182		86
3	13	51	17	207	14904		72
4	14	66	17	219	12483		57
5	12	79	17	1354	59576		44
6	6	86	17	1031	38147		37
7	7	94	20	321	9309		29
8	6	101	20	415	9130		22
9	5	107	20	1079	17264		16
10	4	112	20	230	2530		11
11	3	116	20	141	987		7
12	2	119	21	52	208		4
13	1	121	23	10	20		2
14	1	123	23				0

Transfer Part

- The debris to debris final transfer was implemented in matlab
- The sequence was re-optimized with respect to the transfer times, cause in the tree that was a fixed value
- The re-optimizer had as cost function the sum of all ΔV, the design variables were transfer time
- 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
- For the re-optimizer we are not using a look up table for the $\Delta \mathrm{V}$, we are calculating it during the *fmincon* call

Transfer Part

- 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
- The more demanding part is the constraint function

Transfer Part, Constraint Function

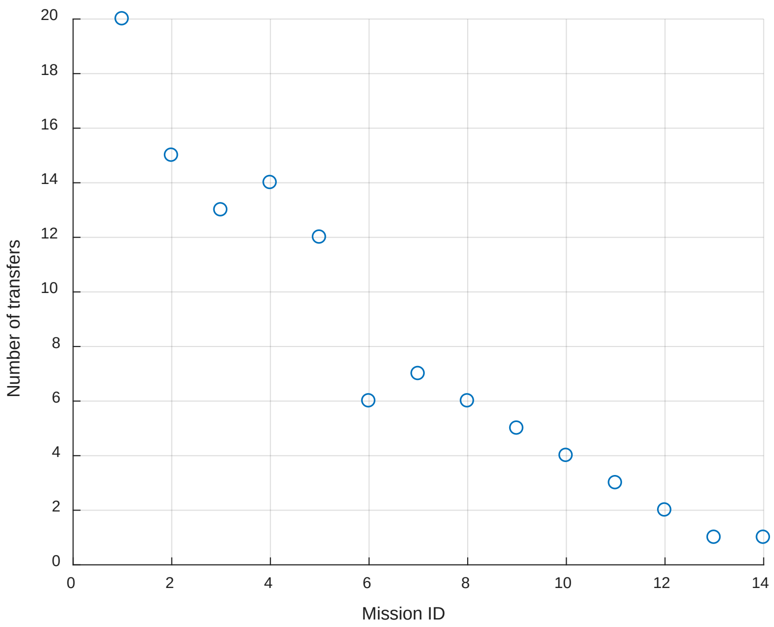
- In the constraint function we integrate the equation of motion between the maneuvers until we reach our final state
- Than the final state should equal the arrival debris state at that time (equality constraints)
- We have a global parameter in order to activate or deactivate the constraints, and we can choose between the Cartesian state vector, Keplerian elements, or a mixture, or a subset
- Another inequality constraint is that the sum off all transfer times between maneuvers is not larger than the transfer time we got out of the tree

Transfer Part, ode solver + *fmincon* issue

- The use of ode solvers in a *fmincon* call caused some issues with respect to stability
- When the error from the ode is to large and variable step size solvers are used the Jacobian and Hessian gets instable
- We implemented a RK8 in c++ with a step size of 50 sec to overcome that issue

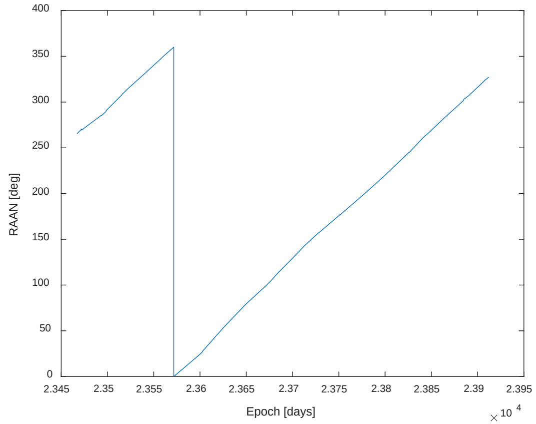
Results

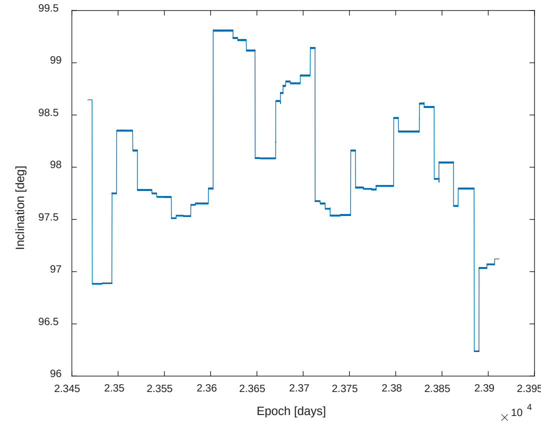
- Our submitted solution had a total of n =14 missions and a performance index J = 949.85MEUR

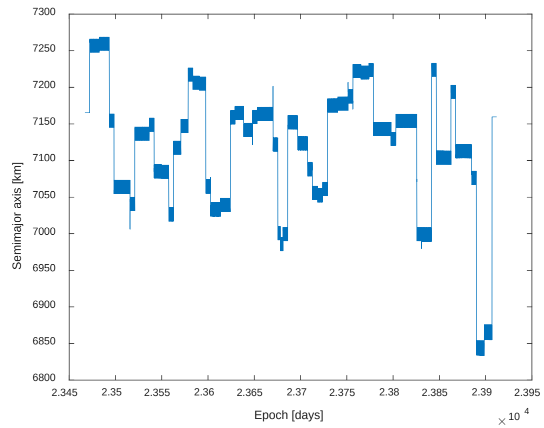


Results

- For the first mission the evolution of Ω, a and i are plotted over time







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
- 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