GTOC 9: Results from the Astrodynamics Research Group of Penn State

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

May 8, 2017

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 [2]. 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.

*Corresponding author. E-mail: davide.conte90@gmail.com

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 [2]. 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)*, 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 suboptimal 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.

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

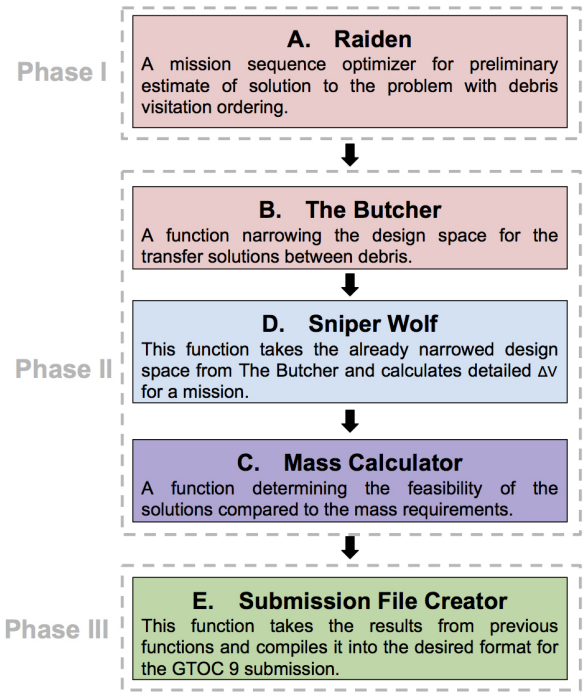

Figure 1: Program Flowchart.

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.

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.

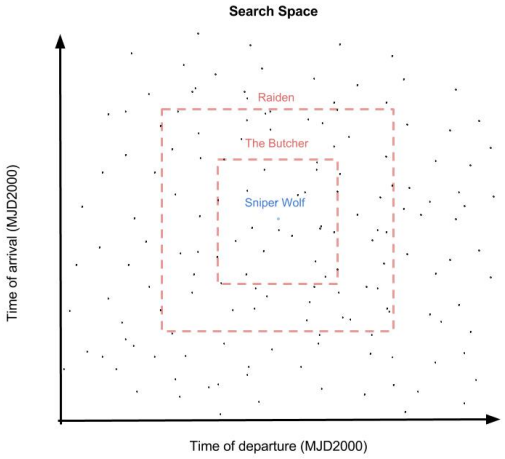

Figure 2: A visual representation of the solution search space for transfers between debris.

1. *Initialization*
   (a) Let ω 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_ω 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 μ, create as many branches as the problem allows with each branch obtained by appending to the partial solution associated with μ the variable corresponding to the next level of the tree, and insert the created nodes (i.e., the offsprings of μ) into B_ω.
   (b) If a node μ of B_ω is a leaf, then
      i. compute its objective function value z_μ;
      ii. if $z_\mu < z^*$, update z^* and the incumbent solution;
      iii. remove μ from B_ω.
   (c) Assess the potential of each node μ' of B_ω 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 μ').
   (d) Rank the nodes of B_ω in a non-increasing order of their values.
   (e) Insert the $\min\{\omega, |B_\omega|\}$ best nodes of B_ω 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. [1]

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 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 through-

out 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 dynamic 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 be passed on 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 applied. [4] 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.

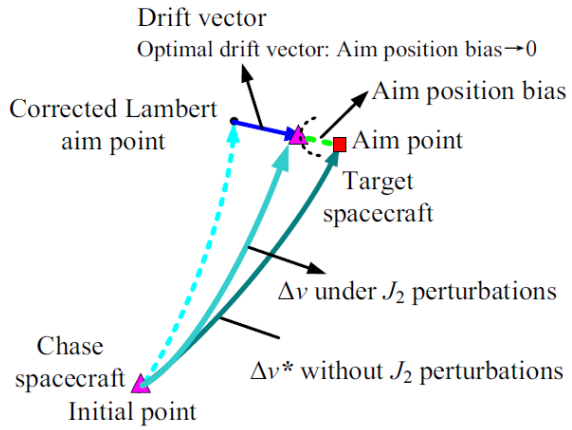

Figure 4: Drift vector effects due to J2 dynamics. [3]

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 ($\delta r < \delta r_{\text{TOL}}$) and/or optimal ($\Delta v < \Delta v_{\text{estimated}}$) 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 Δv1 and Δv2 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.

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 transfers Δv and propellant mass is given in Table 1. The average required Δv and transfer time

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
        δr $\leftarrow$ Compute drift vector magnitude
        $J = 0.001 * [10 * (\delta r - \delta r_{\mathrm{TOL}}) * u(\delta r - \delta r_{\mathrm{TOL}}) + ... \Delta v] \leftarrow$ 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_{best}
  $J_{\mathrm{best}} =$ new J_{best} and Solution $= P_{\mathrm{best}}$
end
end *// end of the while loop*
Return the best Solution

Figure 5: *Sniper Wolf* basic pseudocode.

values per transfer for each mission are shown in Figure 7.

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 likelihood of being able to group the rest of the debris similarly decreases. There is a

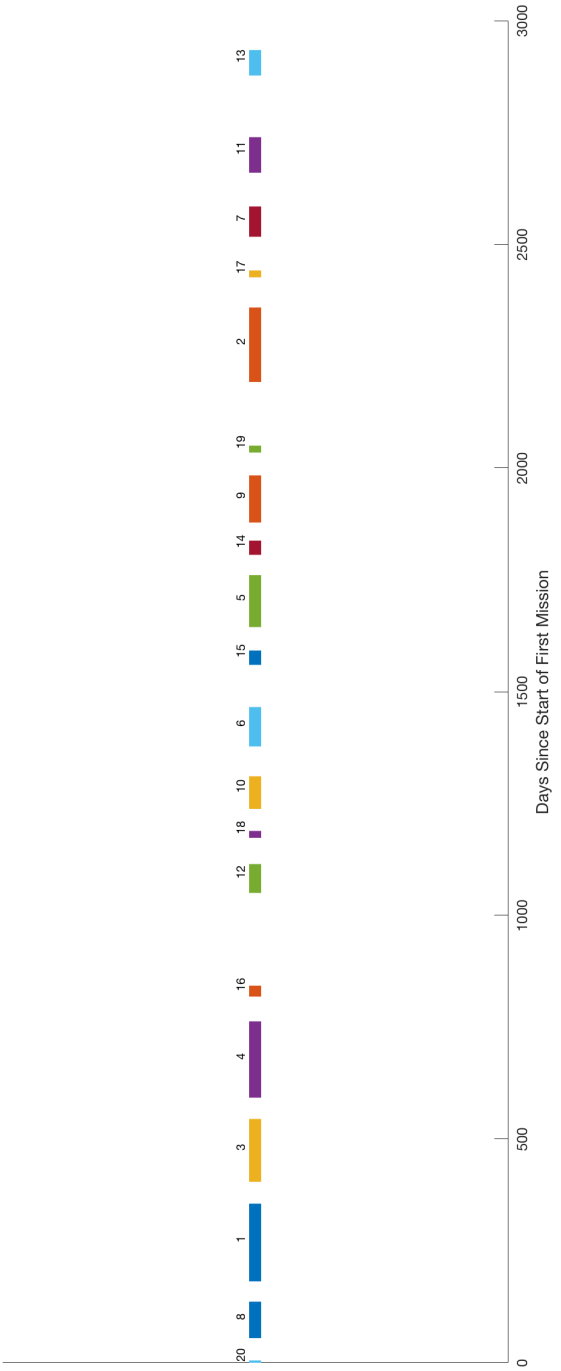

Figure 6: *The Fury* - Mission timeline.

trend that appears, where an increase in the ratio of fuel cost per debris captured manifests as

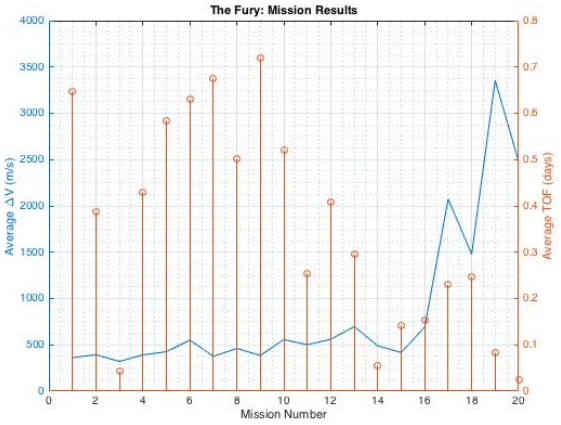

Figure 7: *The Fury* averaged results.

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

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

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 [2]. 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 trajectories.

References

[1] H. Akeb, M. Hifi, and R. MHallah. A beam search algorithm for the circular packing problem. *Computers & Operations Research*, 2009.

[2] Dario Izzo. *Problem description for the 9th Global Trajectory Optimisation Competition*.

[3] 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.

[4] M. Pontani and B. Conway. Particle swarm optimization applied to space trajectories. *Journal of Guidance, Control, and Dynamics*, 33(5):1429–1441, 2010.