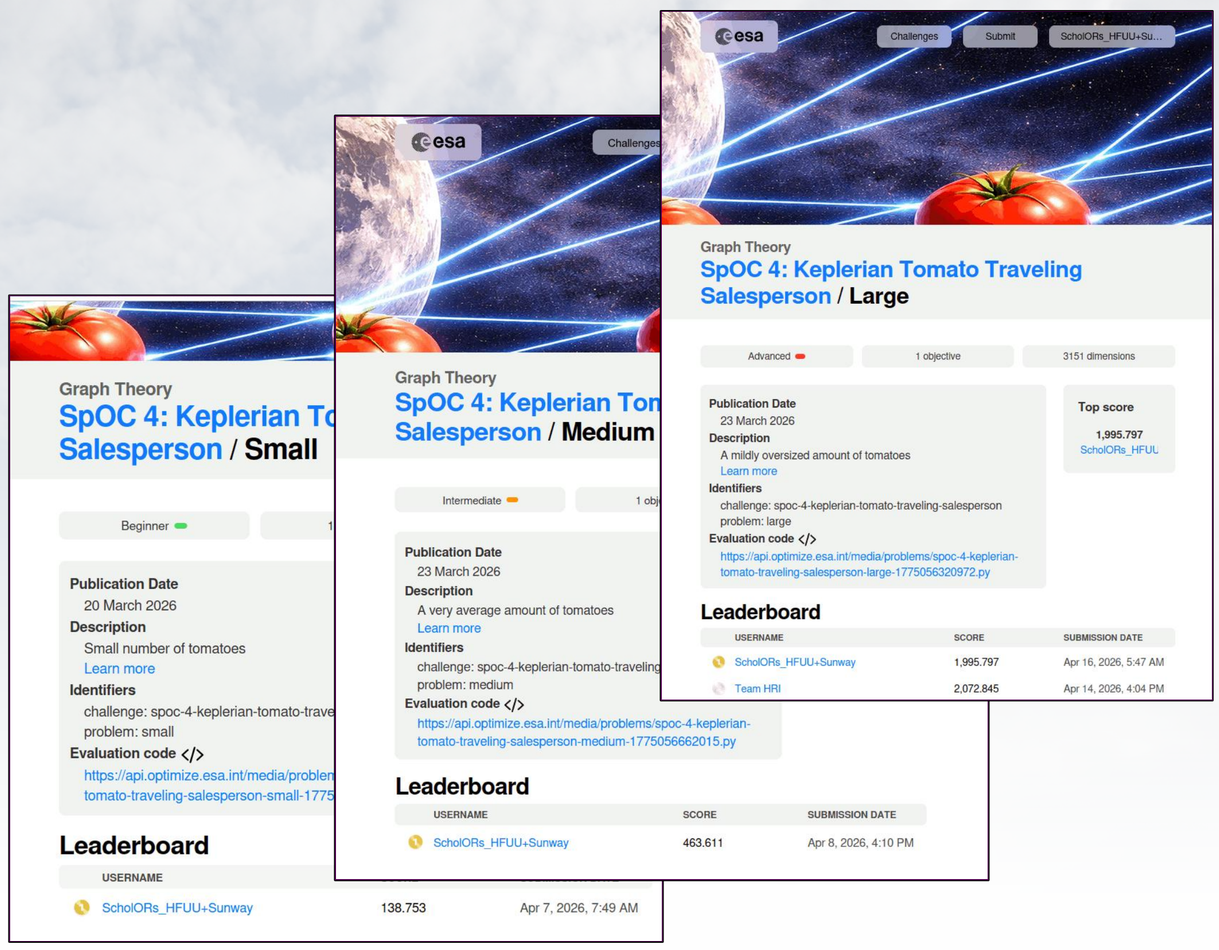
Solving the Space Optimization Competition 4 Challenge 2 of the European Space Agency

— Thomas Weise[1], Sina Abdipoor[2], Zhize Wu[1], and Mingxuan Li[2] —

[1]Hefei University, China and [2]Sunway University, Malaysia

- The Space Optimization Competition (SpOC) is organized by the European Space Agency (ESA) at the Genetic and Evolutionary Computation Conference (GECCO).
- It poses optimization problems and challenges from space exploration and travel.
- We are the ScholORs_HFUU+Sunway team, a cooperation between researchers from Hefei University (合肥大学), China and Sunway University, Malaysia.
- Early on in the challenge, we had good results ... now we rank quite far behind.



The Challenge

- Given are n objects orbiting our moon with their orbital data.
- There are three instances small, medium, and large, with $n=49$, 181, and 1051.
- The goal is to visit each of the objects exactly once as fast as possible, starting at an object of choice.
- For each transfer, our spacecraft may apply a specific change ΔV of velocity.
- ΔV is limited by ΔV_{max}, but 5 times we are allowed to do a larger velocity change $\Delta V \leq \Delta V_{max}^{exception}$
- Limits for the shortest transfer and maximum total time are also given.

Computing Transfers

- We begin by developing a tool $opt\,\Delta V$ that finds the earliest transfer from one object to another one within a given time window $[ts, td]$, given a ΔV-limit ΔV_{max}^{limit}
- If no such transfer exists, the function returns and error.
- We approach this as numerical minimization problem with objective function ψ:

$$\psi = \begin{cases} \Delta V & if\ \Delta V > \Delta V_{max}^{limit} \\ -1/t_b & otherwise \end{cases}$$

- If the transfer at times $\{t_a, t_b\}$ is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{limit}$ is reached
- Once we found such transfer, $-1/t_b$ minimizes the arrival time t_b.
- We use a variety of algorithms, including Nelder&Mead, BFGS, and CMA-ES.
- If a feasible transfer with end time t' is found, we search again in $[t_s, t')$ until the search fails, retaining and returning the best result.

Infeasible Transfers and Preprocessing

- We first assemble a transfer matrix $\mathcal{T}$ for each instance, telling us whether we can travel from object to another either not at all, with excess velocity change, or with a normal velocity change.
- The matrices $\mathcal{T}$ are sparse, most transfers are infeasible!
- On instance small, only 2.8 other objects can be reached with $\Delta V \leq \Delta V_{max}$ in average!
- The vast majority of visit sequences will not be feasible, so always using $opt\,\Delta V$ would be a huge waste of runtime

Permutation Encoding

- We use an encoding based on permutations x of the first n natural numbers.
- A permutation x is decoded to a solution vector y with the actual travel times using a three-stage decoding function γ.
- First, $\gamma(x)$ processes x from beginning to end and divides it into fragments π using the transfer matrix $\mathcal{T}$.
- A fragment π is a sub-sequence of x such that normal ΔV can be used to travel from one object to the next.
- Fragments that can be stitched together by pre-pending and appending without violating this condition are merged.
- Fragments that can be inserted into each other without violating the condition are merged.
- This is repeated with the relaxed requirement to allow excess velocity changes.
- Finally, all remaining fragments are stitched together regardless, yielding a new permutation x'.
- We use the transfer matrix $\mathcal{T}$ to count the expected constraint violations in x'.
- If constraint violations are expected, the decoding stops — this is very fast, no physics computation needed at all.
- Otherwise, we process the permutation x' from beginning to end computing transfers using the actual physics computation $opt\,\Delta V$.
- We make sure that each solution uses all 5 permitted excess velocity changes (even if it does not need them), because then we can make travel faster.

Optimization

- We apply a randomized local search (RLS) to minimize the objective function f:

$$f(y) = \begin{cases} p(y) & if\ p(y) > 0 \\ -1/t_{tot}(y) & otherwise \end{cases}$$

- A solution y is infeasible if $p(y) > 0$, in which case the penalty p is returned.
- For feasible solutions, the total time t_{tot} is minimized.

SUNWAY UNIVERSITY

HEFEI UNIVERSITY