Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson

Thomas Weise
tweise@hfuu.edu.cn
School of Artificial Intelligence and Big Data,
Hefei University
Hefei, Anhui, China

Sina Abdipoor
sinaa@sunway.edu.my
Department of Data Science and Artificial Intelligence,
Faculty of Engineering and Technology,
Sunway University
Selangor, Malaysia (corresponding author)

Mingxuan Li
mingxuan.li@se23.qmul.ac.uk
BUPT International School,
Queen Mary University of London
London, United Kingdom

Zhize Wu
wuzz@hfuu.edu.cn
School of Artificial Intelligence and Big Data,
Hefei University
Hefei, Anhui, China

Abstract

The goal of the second challenge of the 2026 Space Optimization Competition (SpOC4) is to find a shortest path visiting a set of objects in lunar orbits, while obeying constraints on maximum velocity change and travel time. We transform this task to a combinatorial problem by first constructing a matrix storing which orbit can be reached from which other (and how). Using this information, the optimization algorithm can concentrate on finding potentially feasible solutions without using any physics computation. Once potentially feasible solutions are found, the actual physical transfer information is computed.

CCS Concepts

• **Computing methodologies** → **Discrete space search**; **Randomized search**; *Heuristic function construction.*

Keywords

SpOC, Space Optimisation Competition, Graph Theory, Orbits

ACM Reference Format:
Thomas Weise, Sina Abdipoor, Mingxuan Li, and Zhize Wu. 2026. Solving the SpOC 4 Challenge 2 with a Multi-Step Permutation Encoding for the Keplerian Traveling Salesperson. In *Genetic and Evolutionary Computation Conference (GECCO Companion '26), July 13–17, 2026, San Jose, Costa Rica.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3795101.3815557

1 Introduction

The second challenge of the 2026 Space Optimisation Competition (SpOC4), dubbed "Keplerian Tomato Traveling Salesperson," is focused on finding the shortest path through a set of objects orbiting moon. The three instances–**small**, **medium**, and **large**, of this problem provide the following data: There are n objects orbiting moon, with $n = 49$, 181, and 1051 for the three instances, respectively. For each of them, we are given the the semi-major axis a in meters, the eccentricity e, the inclination i in radians, the right ascension Ω of the ascending node in radians, the argument ω of periapsis in radians, and the mean anomaly ν in radians. The goal is to find a method to visit each of the objects exactly once with a spacecraft, starting at an object of choice. We do *not* need to return to the origin, but must observe a set of constraints. For each transfer, our spacecraft may apply a specific change ΔV of velocity. It must hold that $0 \leq \Delta V \leq \Delta V_{max}$ for all transfers, with the exception of $E = 5$ occasions, where we are allowed to use $0 \leq \Delta V \leq \Delta V_{max}^{exception}$, with $\Delta V_{max} = 100$ and $\Delta V_{max}^{exception} = 600$ for all instances. Furthermore, the mission starts at time $t_0 = 0$ and ends the latest after t_{max} days and by then, all objects must have been visited. No transfer can be shorter than Δt_{min} days. For **small**, **medium**, and **large**, it respectively holds that $t_{max} = 200$, 500, and 3000 and $\Delta t_{min} = 0.001$, 0.01, and $1/1140$.

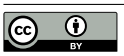
We need to find the sequence in which the n objects are visited as well as the departure times and times of flight for each of the $n-1$ transfers. As tool, we are provided a function $\delta(o_s, o_d, t_s, \Delta t)$, which can compute the corresponding ΔV value for a transfer from object o_s to object o_d starting at day t_s with a flight time of Δt days.

2 Preprocessing: Computing Transfers

The first step to solve the problem is to develop a tool that finds optimized transfer times $\{t_s^\star, \Delta t^\star\} \leftarrow opt\Delta V(o_s, o_d, t_s, t_d, \Delta V_{max}^{lim})$ for objects o_s and o_d within a time window $[t_s, t_d]$ such that $0 \leq t_s \leq t_s^\star < t_s^\star + \Delta t_{min} \leq t_s^\star + \Delta t^\star \leq t_d \leq t_{max}$ and that the resulting $\Delta V = \delta(o_s, o_d, t_s^\star, \Delta t^\star) \leq \Delta V_{max}^{lim}$, allowing us to set ΔV_{max}^{lim} to either ΔV_{max} or $\Delta V_{max}^{exception}$. If no such transfer exists, we assume the function returns $\emptyset$. We implement this as minimization of Equation 1.

$$\psi(t_a, t_b) = \begin{cases} \delta(o_s, o_d, t_a, t_b - t_a) & \text{if } \delta(o_s, o_d, t_a, t_b - t_a) > \Delta V_{max}^{lim} \\ -1/t_b & \text{otherwise} \end{cases} \quad (1)$$

If the transfer is infeasible, the positive ΔV value is returned and can further be minimized until $\Delta V \leq \Delta V_{max}^{lim}$ is reached. Once this happens, $-1/t_b$ minimizes the arrival time. In our implementation, we first sample the interval $[t_s, t_d]$ uniformly to get an initial guess of the solution. Then for sub-intervals of 2 and 5 days as well as



GECCO Companion '26, San Jose, Costa Rica

ACM ISBN 979-8-4007-2488-6/2026/07
https://doi.org/10.1145/3795101.3815557

finally the full interval, we throw a variety of algorithms on this one after the other, including CG, Nelder&Mead, BFGS, and Powell's algorithm. If a feasible solution with end time t' is found, we search again in $[t_s, t')$ until the search fails, retaining and returning the best result.

Applying this approach to the problem, we realize two things: First, it is relatively slow compared to how often this may need to be done. Second, not all objects can be reached from other objects in a feasible way. We therefore first assemble a transfer matrix $\mathcal{T}$ for each instance, such that that $\mathcal{T}_{o_s,o_d}$ is 0 if no feasible transfer is possible between objects o_s and o_d, 1 if transfer requires a ΔV with $\Delta V_{max} < \Delta V \leq \Delta V_{max}^{exception}$, and 2 if transfer is possible with $\Delta V \leq \Delta V_{max}$, at least at some time in $[0, t_{max}]$. Of course, using heuristic optimization to construct the matrix, these values are not necessarily optimal, but for each 1 or 2, at least one proof was found. Constructing the matrices is fast for **small** but took about a week for **large**. The results are quite important, though: On problem **small**, from each object only 2.8 other objects can be reached with $\Delta V \leq \Delta V_{max}$ and 16.76 using $\Delta V_{max} < \Delta V \leq \Delta V_{max}^{exception}$ in average. On **medium** and **large**, the respective values are (45.9, 17.6) and (211.8, 139.4).

3 Approach: Encoding

As search space, we use the set $\mathbb{X}$ of permutations x of the first n natural numbers. This allows us to apply standard metaheuristics to search. Each permutation $x \in \mathbb{X}$ is decoded to a solution vector $y \in \mathbb{Y}$ using a three-stage decoding function $g : \mathbb{X} \mapsto \mathbb{Y}$. Each y has the structure required for challenge submission with the additional attributes θI, θE, ϕI, and ϕE, which we all set to 0 at the beginning of the decoding procedure.

In the first stage, $g(x)$ processes x from beginning to end and creates *fragments*. First, fragment π is a sub-sequence of x such $\mathcal{T}_{\pi_i,\pi_{i+1}} = 2$ for all $i \in 1..len(\pi)-1$, i.e., such that each object can theoretically be reached from its predecessor using normal speed. Then, fragments that can be stitched together by pre-pending and appending without violating this condition are merged even if they are not neighbors in x. Then we attempt to do the same by trying to insert fragments into one another where possible. Then, the requirement is relaxed to $\mathcal{T}_{\pi_i,\pi_{i+1}} > 0$, meaning that excess speeds are permitted. Finally, all remaining fragments are stitched together regardless. This yields a new permutation x'.

In the second stage, we use $\mathcal{T}$ to count the numbers θI and θE of theoretical impossible and excess-ΔV transfers in x'. If $\theta I > 0$ or $\theta E > E$, the decoding stops. No physical computations are made and δ or $opt\Delta V$ are not used. This allows us for a very fast search for solutions that *could be* feasible, only considering permutations.

If the solution could be feasible, we can now process it from beginning to end in the third stage. Starting with the first object at $t = 0$, we try to find the optimal transfer from one object to the next in the sequence using $opt\Delta V$. The arrival time on the previous object becomes the earliest starting time for the next. We always first try to find a move with $\Delta V_{max}^{lim} = \Delta V_{max}$. If that does not work, we use $\Delta V_{max}^{lim} = \Delta V_{max}^{exception}$, incrementing ϕE on success and ϕI at failure.

A solution having $\phi I = 0$ and $\phi E \leq E$ is feasible. We take special care of the case that $\phi E < E$: Here, we did not use all the available excess speeds. This means that we could amend the solution, because using a higher ΔV would lead to an earlier completion time. We thus go back to the first transfer that can be sped up by at least 10% if $\Delta V_{max}^{exception}$ is used as ΔV_{max}^{lim} and reconstruct the rest of the solution. This is done until we arrive at $\phi E = E$.

4 Approach: Optimization

We can now search in the space $\mathbb{X}$ of permutations x of the first n natural numbers using our decoding procedure described above. We minimize the objective function f given in Equation 3, which first computes a penalty term p (Equation 2), that combines the errors that may make a solution infeasible in a prioritization scheme. If no such constraint is violated, we minimize -1 divided by the total time $t_{tot}(y)$ required by the solution, i.e., the time when our spacecraft arrives at the last object.

$$p(y) = \max\{0, \phi E - E\} + (n+1)(\phi I + (n+1)(\max\{0, \theta E - E\} + (n+1)\theta I)) \quad (2)$$

$$f(y) = \begin{cases} p(y) & \text{if } p(y) > 0 \\ -1/t_{tot}(y) & \text{otherwise} \end{cases} \quad (3)$$

We apply a local search procedure based on randomized local search. This procedure begins by sampling a permutation x uniformly at random from $\mathbb{X}$. As unary operator to sample the neighborhood of the current best solution, a swapping operator is applied, which swaps two elements with probability 0.5, 3 with probability 0.25, 4 with probability 0.125, and so on. We incorporate restarts as ultima ratio against stagnation into this local search and also apply Frequency Fitness Assignment [2, 3] as selection criterion in some of the steps to further exploration.

One vital component during our decoding procedure is the function $opt\Delta V$, which finds good but not necessarily optimal transfer times. We introduce the following amendment: If a search step discovers a new best-so-far solution y by decoding a permutation x, we re-decode x using procedure $opt\Delta V'$, which is identical to $opt\Delta V$ but additionally also uses Bi-Pop CMA-ES. This costs considerably more time, but yields better results (whose objective values are not counted towards the best-so-far statistic, which we reset at each restart).

We implement this using the **moptipy** library [1]. On instance **large**, the first feasible solution, having $t_{tot} = 2689.4$, was sampled after 187 seconds, but not all runs were that fast. Another one took 3.8 hours to reach feasibility.

References

[1] Thomas Weise and Zhize Wu. 2023. Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms. In *Genetic and Evolutionary Computation Conference Companion (GECCO'2023 Companion), July 15-19, 2023, Lisbon, Portugal*. ACM, New York, NY, USA. doi:10.1145/3583133.3596306

[2] Thomas Weise, Zhize Wu, Xinlu Li, and Yan Chen. 2021. Frequency Fitness Assignment: Making Optimization Algorithms Invariant under Bijective Transformations of the Objective Function Value. *IEEE Transactions on Evolutionary Computation* 25, 2 (2021), 307–319. doi:10.1109/TEVC.2020.3032090

[3] Thomas Weise, Zhize Wu, Xinlu Li, Yan Chen, and Jörg Lässig. 2023. Frequency Fitness Assignment: Optimization without Bias for Good Solutions can be Efficient. *IEEE Transactions on Evolutionary Computation* 27, 4 (Aug. 2023), 980–992. doi:10.1109/TEVC.2022.3191698