aerospace

MDPI

Article

Minimum-Fuel On-Orbit Servicing via A* Search Algorithm

Edoardo Maria Leonardi [1,2], Fabio Curti [1,*], Lorenzo Federici [1] and Mauro Pontani [3]

[1] Department of Systems & Industrial Engineering, University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721, USA; emleonardi@arizona.edu or edoardomaria.leonardi@uniroma1.it (E.M.L.); lorenzof@arizona.edu (L.F.)
[2] Department of Astronautical, Electrical and Energy Engineering, Sapienza University of Rome, Via Salaria 851, 00138 Rome, Italy
[3] Department of Mechanical and Aerospace Engineering, Sapienza University of Rome, Via Eudossiana 18, 00184 Rome, Italy; mauro.pontani@uniroma1.it
* Correspondence: fcurti@arizona.edu

Abstract

On-Orbit Servicing (OOS) represents a viable strategy toward a sustainable and extended exploitation of the Low-Earth-Orbit (LEO) environment. The design of OOS missions requires optimizing both the scheduling of visited objects and the transfer trajectory between each pair of orbits, resulting in the great complexity of the global mission planning problem. This research considers a servicing spacecraft equipped with a high-thrust propulsion system, required to perform multiple orbit transfers to visit several Resident Space Objects (RSOs) in a given time frame with minimum fuel consumption. The proposed method leverages a two-stage approach: (i) first, the optimal transfers are computed for all pairs of orbits and discretized dates, and the associated overall velocity changes are stored in a cost matrix; (ii) then, the problem of visiting all RSOs is cast as a search problem, and the solution space is explored through an A* algorithm. The transfer strategy exploits intermediate drift orbits to increase the differential precession due to the J_2 harmonic of the Earth's gravitational potential. Moreover, the A* procedure leverages a heuristic function based on a modified version of the Held–Karp algorithm, which is proven to be admissible and consistent, meaning that the optimal solution is always reached. The proposed strategy is integrated within a flexible architecture, where operational constraints on phasing and servicing activities can be enforced as well. Finally, the methodology at hand is successfully applied to a case study from the literature involving three successive missions, in charge of visiting 5 RSOs each. Different discretization grids are considered, and the results are compared in terms of overall velocity change and computational time.

Keywords: on-orbit servicing; A* search algorithm; traveling salesman problem; low-Earth-orbit

Check for updates

Academic Editor: Gang Zhang

Received: 9 May 2026
Revised: 23 June 2026
Accepted: 25 June 2026
Published: 30 June 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The development of mission analysis techniques for On-Orbit Servicing (OOS) constitutes a research area of undeniable importance. Supporting already operational satellites instead of replacing them represents a cost-efficient approach to space activities, especially for missions in geostationary orbit (GEO), where access is much more expensive. OOS spacecraft in GEO orbits are already operational [1], such as the commercial Mission Extension Vehicle (MEV) [2], while the planned ESA RISE mission will test in-orbit technologies to pave the way for commercial life extension services for geostationary satellites [3]. However, the coming decades will see a sustained utilization of the Low-Earth-Orbit (LEO)

environment, e.g., enabling a continued and larger human presence, reducing space habitation risk, making access to space cheaper, and unlocking commercial opportunities for OOS. In particular, with the development of proliferated constellations (e.g., Starlink, OneWeb, and Kuiper), the opportunities of deploying OOS architectures in the LEO environment is attracting greater interest, and the benefits are the subject of current investigation [1]. For instance, OOS technologies allow refueling, orbital maneuvering, modular assembly of complex space systems, and repairing [4]. Furthermore, the increase in the number of RSOs requires the development of more sustainable approaches to constellation maintenance, as well as the design of Active-Debris Removal (ADR) missions to clean up the more polluted LEO environment [5].

The preliminary mission design for OOS missions involves optimizing the sequence and timing of visited RSOs, i.e., minimizing the overall velocity change or operational time. To this end, in order to reduce the associated cost, several multi-target missions have been proposed. These scenarios involve a mother ship visiting multiple RSOs in order to conduct maintenance operations or to deorbit uncontrolled targets. Specifically, the proposed framework considers OOS mission scenarios where servicing maneuvers (such as for station keeping or deorbiting) are not required (e.g., installing deorbiting equipment for ADR, refueling, assembly, and repairing). This problem can be cast as a variant of the classic Traveling Salesman Problem (TSP) [6], and several strategies have been proposed for solving the combinatorial problem arising from OOS missions through either high [7] or low thrust [8]. These methods for solving combinatorial optimization problems fall into three main classes [9]:

- *direct enumeration*, that consists in the enumeration of all candidate solutions to obtain the globally optimal solution, thus being suitable only for small-scale problems;
- *implicit enumeration*, where pruning strategies are adopted so that only regions close to the optimal solution are explored. Within this area, branch-and-bound methods have been investigated [10], as well as tree search procedures [11], where the optimal sequencing is achieved by iteratively adding one object at a time. However, the effectiveness of these approaches strongly relies on the pruning criteria, which must drive the exploration of the solution space without losing optimality or increasing the computational complexity;
- *stochastic programming methods* represent a good compromise between the two aforementioned strategies, as they allow finding near-optimal solutions without a heavy computational load. To this aim, metaheuristic approaches have been investigated, such as Simulated Annealing (SA) [12–14] or Genetic Algorithms [15].

Recently, machine learning techniques have been proposed for ADR missions, though with modest populations of objects [16,17]. As a matter of fact, these methodologies allow exploring large solution spaces with limited computational cost, but their use in these mission scenarios has only recently emerged in the scientific literature. Moreover, to tackle its complexity, some authors proposed splitting the original combinatorial optimization problem into multiple smaller-scale ones which can be more efficiently solved, such as through tree search approaches [18].

Actually, the scenario at hand differs from the standard TSP, as the position of the targets is time-dependent due to orbit dynamics [19]. Therefore, Izzo et al. developed both a static and dynamic variant of the standard TSP for a considerably larger set of debris through the inver-over algorithm, though they called for better genetic operators for the dynamic problem [20]. As a matter of fact, further complexity arises when the J_2 perturbation is taken into account, as it induces a variation in the orbital plane. In order to reduce the high cost associated with out-of-plane maneuvers, orbital plane precession can be exploited to perform orbit transfers strategically. To this end, Cerf proposed the use

of intermediate drift orbits to leverage the natural precession of orbital planes due to the Earth's oblateness [21]. In his work, nonlinear programming is employed to compute all possible transfers according to the orbits of the RSOs and the rendezvous timing. Then, the sequence and timing of visited objects are identified through an SA algorithm. A similar framework was built by Federici et al., who investigated three different heuristic functions to increase the efficiency of an A* search algorithm that maximizes a score index on the removed debris, assuming a maximum mission duration and overall velocity change [11]. Again, intermediate drift orbits are exploited to leverage natural precession of the orbital planes for fuel-efficient impulsive maneuvers.

The study that follows aims at developing a search algorithm for the design of missions related to OOS operations. A two-stage approach is being adopted as a well-established methodology [22], exploiting natural orbital plane precession for the purpose of reducing the overall fuel consumption:

- first, the overall velocity changes associated with each transfer leg are determined through both analytical models and optimization procedures and stored in related cost matrices;
- then, an A* search algorithm is in charge of determining the optimal sequence and timing of visited objects, i.e., minimizing the overall velocity change.

Specifically, this work is intended to propose the following contributions:

- the A* search algorithm formulated in [11] is modified to the aim of determining the minimum-cost tour within a given set of RSOs, and extended to address operational constraints to allow for phasing and servicing;
- a heuristic function leveraging a modified version of the Held–Karp algorithm [23] is developed to estimate the remaining mission cost and drive node expansion. The design of the heuristic function is intended to guarantee its admissibility and consistency for the purpose of ensuring convergence to the optimal solution;
- the time-dependent TSP problem is also formulated through an Integer Linear Programming (ILP) approach [11], which is revised and extended to address operational constraints and provide a benchmark solution against which the approach at hand can be compared;
- three different test cases from the literature are considered with the aim of assessing the performance of the algorithm under different numerical setups, with an emphasis on the computational cost associated with the search for optimal solutions;
- a scalability study is proposed in order to assess the performance of both approaches along sets of RSOs with increasing size.

In conclusion, this research has the ultimate intent of providing a simple yet flexible architecture for OOS mission design, while focusing on the inherent tradeoff between optimality and computational efficiency. This paper is organized as follows. Section 2 addresses the computation of the optimal transfers between all pairs of orbits and epochs. Section 3 introduces the A* approach to the solution for the combinatorial problem, as well as the formulation of the time-dependent TSP as an ILP problem. Finally, Section 4 shows the numerical results for the case study discussed in the literature.

2. Orbit Transfer Problem Statement and Solution

This section is devoted to introducing the optimization problem at hand and presenting the solution strategy adopted to compute the optimal transfers. The scenario at hand involves a servicing spacecraft, equipped with a high-thrust propulsion system, in charge of performing multiple orbit transfers to visit a set of n RSOs for servicing operations. In the proposed methodology, once a given orbit is reached, a given waiting time T_w is required to

perform phasing and OOS operations, while servicing maneuvers are not considered. Each visited orbit is described in terms of Classical Orbital Elements (COEs) $(a, e, i, \Omega, \omega)$, where a is the semi-major axis, e is the eccentricity, i is the inclination, Ω is the Right Ascension of the Ascending Node (RAAN), and ω is the argument of the pericenter. Moreover, each orbit is subject to nodal precession due to the Earth's oblateness, which strongly dominates above all other sources of perturbation within the range of altitudes of interest [24]. The aim of the optimization strategy is to find the best sequence and timing of the visited RSOs in order to minimize the overall fuel consumption.

2.1. Cost Matrices

As a preliminary step, a mesh of discretized epochs is created to compute the optimal transfers between all pairs of n orbits at different times, and the associated velocity changes are stored in cost matrices. To this aim, the following indices are introduced:

- i: departure orbit ($i = 1, ..., n$);
- j: arrival orbit ($j = 1, ..., n$);
- k: epoch times ($k = 0, ..., N_t$);
- m: transfer duration ($m = 0, ..., M_k$);

where $N_t + 1$ is the length of the vector $\boldsymbol{\tau}$ composed of all discretized epochs τ_k ($k = 0, ..., N_t$) such that

$$\tau_k = k\Delta\tau, \tag{1}$$

where $\Delta\tau$ denotes the discretization interval. Analogously, $M_k + 1$ is the length of the vector $\boldsymbol{\delta}$ composed of all durations δ_m $(m = 0, ..., M_k)$ available for each transfer,

$$\delta_m = m\Delta\tau, \tag{2}$$

with

$$M_k = \min\left(M_{max}, N_t - k\right),\ k = 0, ..., N_t \tag{3}$$

and $\Delta\tau M_{max}$ denoting the maximum time allowed for a transfer. In particular, transfers corresponding to $m = 0$ are associated with direct transfers, i.e., without the use of intermediate drift orbits. Finally, for each pair (i, j), the optimal transfer at epoch τ_k with duration δ_m is identified, and the associated velocity change ΔV_{ijkm} fills the component of a four-index matrix,

$$\Delta V_{mat}[i, j, k, m] = \Delta V_{ijkm},\ \text{with } i, j = 1, ...n;\ k = 0, ..., N_t;\ m = 0, ..., M_k. \tag{4}$$

2.2. Orbit Transfer Strategy

The strategy to achieve orbit transfers is proposed by Cerf [25] and relies on the J_2 nodal precession to drive the orbital plane of the chaser toward that of the target. Each mission consists of three distinct phases (see Figure 1):

- a non-coplanar Hohmann transfer from the initial orbit (subscript i) to a drift orbit;
- a coast phase along the drift orbit (subscript d) to synchronize the RAAN with that of the target orbital plane;
- a non-coplanar Hohmann transfer from the drift orbit to the the target orbit (subscript j).

Some simplifying hypotheses are introduced:

- the orbit of the RSOs and the drift orbit are assumed to be circular. While elliptical drift orbits may alternatively be employed with lower fuel consumption, circular orbits guarantee a faster drift rate and thus are preferred within this framework [25];

- the (modest) propellant consumption associated with phasing operations is neglected [25];
- the short duration of the Hohmann transfers is neglected with respect to the drift time.

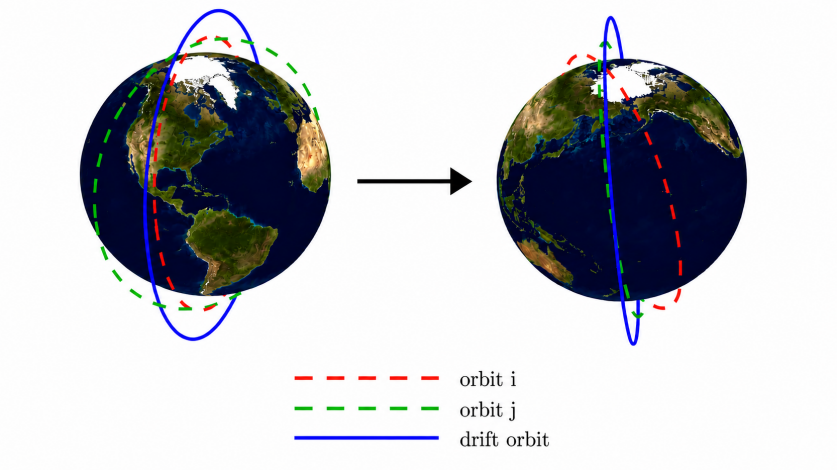


Figure 1. Orbit transfer strategy.

2.2.1. Determination of Drift Orbit Parameters

The computation of the velocity change ΔV_{ijkm} entails determining the semimajor axis a_d and inclination i_d of the drift orbit that minimize the transfer cost. This can be cast as a constrained optimization problem:

$$\min_{a_d, i_d} \Delta V_{ijkm}, \quad i, j = 1, ..., n;\ k = 0, ..., N_t;\ m = 0, ..., M_k; \tag{5}$$

$$\text{with } |\Omega(\tau_{k+m}) - \Omega_j(\tau_{k+m}) \pm 2z\pi| = 0,\ z \in \mathbb{Z}, \tag{6}$$

where $\Omega(\tau_{k+m})$ and $\Omega_j(\tau_{k+m})$ respectively denote the RAAN of the servicing spacecraft and of the target orbital plane at time τ_{k+m},

$$\Omega(\tau_{k+m}) = \Omega_i(\tau_0) + \dot{\Omega}_i(\tau_k - \tau_0) + \dot{\Omega}_d(\tau_{k+m} - \tau_k), \tag{7}$$

$$\Omega_j(\tau_{k+m}) = \Omega_j(\tau_0) + \dot{\Omega}_j(\tau_{k+m} - \tau_0), \tag{8}$$

while the time derivatives of the RAAN due to the J_2 perturbation are given by Equation (9) (with zero eccentricity) [26],

$$\begin{cases} \dot{\Omega}_i = -\dfrac{3}{2} \dfrac{J_2 \sqrt{\mu} R^2}{a_i^{7/2}} \cos i_i \\ \dot{\Omega}_j = -\dfrac{3}{2} \dfrac{J_2 \sqrt{\mu} R^2}{a_j^{7/2}} \cos i_j \\ \dot{\Omega}_d = -\dfrac{3}{2} \dfrac{J_2 \sqrt{\mu} R^2}{a_d^{7/2}} \cos i_d \end{cases}, \tag{9}$$

where μ is the Earth's gravitational parameter, R is its equatorial radius, and J_2 is its oblateness coefficient.

2.2.2. Non-Coplanar Hohmann Transfers

As the spacecraft is equipped with a high-thrust engine, non-coplanar orbit maneuvers are modeled as impulsive velocity changes. Let (i_1, Ω_1) and (i_2, Ω_2) represent the

inclination and RAAN of the generic initial and final orbits, respectively. Then, the two orbital planes form an angle θ such that

$$\cos\theta = \sin i_1 \sin i_2 \cos(\Omega_2 - \Omega_1) + \cos i_1 \cos i_2. \tag{10}$$

The strategy employed in this study is proposed by Vallado [27] and consists in achieving a partial inclination change $s\theta$ ($s \in [0,1]$) during the first impulsive maneuver, and the remaining part $(1-s)\theta$ in the second maneuver. Both velocity changes are combined maneuvers (i.e., changing both altitude and inclination), and thus

$$\Delta V_1 = \sqrt{v_1^2 + v_{tr1}^2 - 2v_1 v_{tr1}\cos(s\theta)}, \tag{11}$$

$$\Delta V_2 = \sqrt{v_{tr2}^2 + v_2^2 - 2v_{tr2} v_2 \cos((1-s)\theta)}, \tag{12}$$

where v_1 and v_{tr1} denote the velocities before and after injection in the transfer orbit, respectively, while v_{tr2} and v_2 are the velocities before and after insertion into the final orbit, respectively. In order to determine the optimal value of s, we can take the partial derivative of $\Delta V_1 + \Delta V_2$ with respect to s to yield [27]

$$\sin(s\theta) = \frac{\Delta V_1 v_2 v_{tr2} \sin(1-s)\theta}{\Delta V_2 v_1 v_{tr1}}. \tag{13}$$

Equation (13) requires a numerical process to yield the solution. However, by ignoring second-order terms (i.e., $\Delta V_1 \Delta V_2$), an analytical approximation can be obtained [27],

$$s = \frac{1}{\theta}\arctan\left[\frac{\sin\theta}{\frac{v_1 v_{tr1}}{v_2 v_{tr2}} + \cos\theta}\right]. \tag{14}$$

3. Combinatorial Problem Statement and Solution

Section 3 introduces the formulation of the combinatorial problem and its solution through either an A* search algorithm or an ILP approach.

3.1. Search Problem Statement

The problem of visiting all RSOs with minimum velocity change can be cast as a search problem. To this end, Federici et al. [11] identify each state χ as a sequence of pairs (s_ξ, t_ξ), where s_ξ is the visited object and t_ξ is the associated rendezvous epoch. Here, the same formulation is adopted and extended to include waiting time (T_w) constraints related to phasing and OOS operations. Once T_w is assigned as a fixed design parameter (which is equal for each visited RSO), the associated constraint can be can enforced through a corresponding index [13]

$$l = \inf\left\{z \in \mathbb{Z} : z \geq \frac{T_w}{\Delta\tau}\right\}, \tag{15}$$

where $\mathbb{Z}$ is the set of integer numbers. Thus, the servicing spacecraft is forced to wait at each destination RSO for a time equal to $l\Delta\tau$ before starting another transfer. The mission begins when the spacecraft leaves the first RSO, while it arrives at the last object at most at $N_t - l$, to allow for phasing and OOS operations at the destination RSO.

Each node in the tree is represented by a data structure with four elements:

- the state to which the node corresponds,

$$\chi = \{(s_1, t_1), \ldots, (s_\xi, t_\xi)\}, \quad \xi \leq n; \tag{16}$$

- its parent node;
- the total cost $g(\chi)$ of the path from the root node to the current node,

$$g(\chi) = \sum_{i=1}^{\xi} \Delta V_{s_i, s_{i+1}, t_i/\Delta\tau, (t_{i+1}-t_i)/\Delta\tau - l}; \tag{17}$$

- the heuristic estimation $h(\chi)$ of the best path from the current node to the goal node.

Given a node and the corresponding state χ, the admissible child nodes are associated with the set of successor states $\Xi(\chi)$, obtained by adding a new object and its rendezvous epoch $(s_{\xi+1}, t_{\xi+1})$:

$$\Xi(\chi) = \{\chi' : \chi' = \chi \cup (s_{\xi+1}, t_{\xi+1})\},\ \xi = 1, ..., n-1, \tag{18}$$

with

$$\frac{t_{\xi+1} - t_{\xi}}{\Delta\tau} \leq M_{\xi}, \tag{19}$$

and

$$M_{\xi} = \min\left(M_{max}, N_t - k - (n-\xi)l\right),\ \ \xi = 1, ..., n. \tag{20}$$

Equation (20) implements a pruning strategy that restricts the search only to feasible nodes such that the maximum duration of each orbit transfer allows a minimum time margin $l\Delta\tau$ for each of the remaining $(n-\xi)$ objects to perform phasing and OOS operations. Furthermore, nodes resulting in orbit transfers requiring a ΔV greater than 1 km/s are discarded to limit the search space.

3.2. A Search*

A general tree search algorithm creates a search tree over the state-space graph, starting from a root node, which corresponds to the initial state of the problem at hand. Then, nodes are added to the tree, with each corresponding to a different state in the state space. The edges between any pair of nodes correspond to the actions that enable transitions from one state to another. Starting from the parent node, child nodes resulting from all admissible actions are added to the frontier. The latter includes the ordered set of nodes that have been reached but not yet expanded. At the first iteration, only a parent node exists, i.e., the root node. At each iteration, the frontier acts as a priority queue, where lower-cost nodes are expanded first. The process ends when the goal state is reached.

The A* algorithm is an informed search strategy, i.e., it exploits additional information about the possible location of the goal to drive the search process. This further indication is given as a heuristic function $h(\chi)$, which provides the estimated cost of the cheapest path from the current node to the goal node. Specifically, the A* algorithm relies on a *best-first* search strategy, where nodes with the minimum value of an evaluation function $f(\chi)$ are selected, representing the overall cost from the start to the goal node. For the A* algorithm, the evaluation function is given by [28]

$$f(\chi) = g(\chi) + h(\chi), \tag{21}$$

where $g(\chi)$ is the cost of the path from the initial to the current node, whereas $h(\chi)$ represents the heuristic estimate of the cost from the current node to the final node. The sufficient condition for A* to be cost-optimal is that the heuristic function is admissible, that is, it never overestimates the actual cost of the path. A slightly stronger property is the consistency. A heuristic function $h(\chi)$ is said to be consistent if, for each state χ and each successor state χ' resulting from an action α, we have

$$h(\chi) \leq \delta(\chi, \alpha, \chi') + h(\chi'), \tag{22}$$

where $\delta(\chi, \alpha, \chi')$ is the cost associated with the transition from state χ to state χ' through the action α. Since every consistent heuristic is also admissible, consistency is a sufficient condition for cost optimality. An A* with a consistent heuristic is also optimally efficient, that is, any other algorithm that uses the same heuristic function will expand all nodes expanded by A*.

However, the number of nodes can exponentially increase when large-scale problems are addressed, resulting in heavy computational cost in terms of memory usage and speed. Therefore, a careful definition of the heuristic function allows pruning away nodes that are unnecessary for finding an optimal solution, potentially yielding a significant reduction in the computational burden [11].

3.3. Heuristic Function

A careful selection of an admissible heuristic function is necessary to reduce the computational load of the search algorithm. To this aim, once a given node is expanded at time $t_{\tilde{\xi}}$, a two-index matrix $\Delta V_{t_{\tilde{\xi}}}$ is computed starting from the four-index matrix $\Delta V_{mat}[i, j, k, m]$ with all possible orbit transfers (see Section 2.2),

$$\Delta V_{t_{\tilde{\xi}}}[i, j] = \min_{\substack{t_{\tilde{\xi}}/\Delta\tau + l \leq k \leq N_t - l \\ 0 \leq m \leq M_{\tilde{\xi}}}} \{\Delta V_{mat}[i, j, k, m]\}, \tag{23}$$

for each pair (i, j) of RSOs still not visited. Therefore, matrix $\Delta V_{t_{\tilde{\xi}}}$ contains for each pair of remaining objects the minimum velocity change to achieve the orbit transfer, regardless of the duration and epoch (which must be greater than $t_{\tilde{\xi}}$ anyway). Then, the minimum-cost tour along the matrix $\Delta V_{t_{\tilde{\xi}}}$ that visits each object is computed as

$$h(\chi) = f_{HK}(\Delta V_{t_{\tilde{\xi}}}), \tag{24}$$

where the function f_{HK} implements a variant of the so-called Held–Karp algorithm, a dynamic programming algorithm that solves the TSP with a given cost matrix [23].

3.4. Held–Karp Algorithm

The Held–Karp algorithm [23] is a dynamic programming formulation of the TSP that consists in finding the shortest path to travel from a given city to $n - 1$ cities, and then return home, given the distance $d_{\alpha\beta}$ among each pair of cities in the tour. Equivalently, if 1 denotes the starting city, this corresponds to finding a permutation $P = (1, \sigma_2, ..., \sigma_n)$ of the integers from 1 to n that minimizes the quantity

$$d_{1\sigma_2} + d_{\sigma_2\sigma_3} + d_{\sigma_3\sigma_4} + ... + d_{\sigma_n 1}, \tag{25}$$

where $d_{\alpha,\beta}$ is a set of real (positive) numbers. Given a set $\Sigma \subseteq \{2, 3, ..., n\}$, with dimension $\eta(\Sigma)$, and a city $\sigma_k \in \Sigma$ $(1 \leq k \leq \eta)$, we denote as $\gamma(\Sigma, \sigma_k)$ the minimum cost to visit all cities in the set Σ starting from the first city and terminating at city σ_k,

$$\gamma(\Sigma, \sigma_k) = \begin{cases} d_{1\sigma_k}, & \text{if } \eta(\Sigma) = 1, \\ \min\limits_{\sigma_\rho \in \Sigma_k} \{\gamma(\Sigma_k, \sigma_\rho) + d_{\sigma_\rho\sigma_k}\}, & \text{if } \eta(\Sigma) > 1 \end{cases}, \tag{26}$$

where $\Sigma_k = \Sigma \setminus \{\sigma_k\}$. The quantity $\gamma(\Sigma, \sigma_k)$ can be computed recursively until the cost of visiting all cities and terminating at σ_k is known for each $2 \leq k \leq n$. Finally, if we denote with $\mathcal{C}$ the minimum cost of a complete tour, which includes σ_n as the second-to-last city and returns to the first city, we will have

$$\mathcal{C} = \min_{\sigma_k \in 2,\ldots,n} \{\gamma(\{2,\ldots,n\}, \sigma_k) + d_{\sigma_k 1}\}. \tag{27}$$

Since the problem at hand does not require returning to the first city, a variant of the Held–Karp algorithm can be obtained by simply removing the cost of the last leg $d_{\sigma_k 1}$.

As an illustrative example, the behavior of the algorithm is shown through a simplified scenario involving four elements, namely 1, 2, 3 and 4. The goal is to find a permutation $P = (1, \sigma_2, \sigma_3, \sigma_4)$ that minimizes the quantity

$$d_{1\sigma_2} + d_{\sigma_2\sigma_3} + d_{\sigma_3\sigma_4}. \tag{28}$$

Initially, the set contains only one element. Therefore, we have

$$\gamma(\{2\}, 2) = d_{12}, \tag{29}$$
$$\gamma(\{3\}, 3) = d_{13}, \tag{30}$$
$$\gamma(\{4\}, 4) = d_{14}. \tag{31}$$

Then, the set is expanded to two elements,

$$\gamma(\{2,3\}, 2) = \gamma(\{3\}, 3) + d_{32}, \tag{32}$$
$$\gamma(\{2,3\}, 3) = \gamma(\{2\}, 2) + d_{23}, \tag{33}$$
$$\gamma(\{2,4\}, 2) = \gamma(\{4\}, 4) + d_{42}, \tag{34}$$
$$\gamma(\{2,4\}, 4) = \gamma(\{2\}, 2) + d_{24}, \tag{35}$$
$$\gamma(\{3,4\}, 3) = \gamma(\{4\}, 4) + d_{43}, \tag{36}$$
$$\gamma(\{3,4\}, 4) = \gamma(\{3\}, 3) + d_{34}. \tag{37}$$

Finally, the set contains four elements,

$$\gamma(\{2,3,4\}, 2) = \min\{\gamma(\{3,4\}, 3) + d_{32}, \gamma(\{3,4\}, 4) + d_{42}\}, \tag{38}$$
$$\gamma(\{2,3,4\}, 3) = \min\{\gamma(\{2,4\}, 2) + d_{23}, \gamma(\{2,4\}, 4) + d_{43}\}, \tag{39}$$
$$\gamma(\{2,3,4\}, 4) = \min\{\gamma(\{2,3\}, 2) + d_{24}, \gamma(\{2,3\}, 3) + d_{34}\}. \tag{40}$$

Ultimately, the minimum-cost tour is selected to be

$$\mathcal{C} = \min\{\gamma(\{2,3,4\}, 2) + d_{21}, \gamma(\{2,3,4\}, 3) + d_{31}, \gamma(\{2,3,4\}, 4) + d_{41}\}. \tag{41}$$

3.5. Admissibility and Consistency Proof

Equation (24) can be proven to implement an admissible heuristic: since each entry $\Delta V_{t_\xi}[i,j]$ collects the minimum cost associated with each orbit transfer (i,j), the minimum-cost tour along the matrix ΔV_{t_ξ} cannot overestimates the actual cost of the path.

Moreover, the heuristic in Equation (24) is consistent. Given a state χ with dimension ξ, then, for each generic successor $\chi' \in \Xi(\chi)$ with dimension $\xi + 1$,

$$h(\chi) = f_{HK}(\Delta V_{t_\xi}) = \min_{\chi' \in \Xi(\chi)} \left\{ \Delta V_{t_\xi}[s_\xi, s_{\xi+1}] + h_{s_{\xi+1}}(\chi) \right\} \leq \Delta V_{t_\xi}[s_\xi, s_{\xi+1}] + h_{s_{\xi+1}}(\chi), \tag{42}$$

where $h_{s_{\xi+1}}(\chi)$ is the minimum-cost tour along the matrix $\Delta V_{t_\xi}^{s_{\xi+1}}$ obtained removing the s_ξ-row and $s_{\xi+1}$-column. Now, since the matrix ΔV_{t_ξ} contains the minimum velocity changes to achieve the orbit transfers at times $> t_\xi$,

$$\Delta V_{t_\xi}[s_\xi, s_{\xi+1}] \leq \delta(\chi, \alpha, \chi'). \tag{43}$$

Moreover, for each (i, j),

$$\min_{\substack{t_\xi/\Delta\tau+l \leq k \leq N_t-l \\ 0 \leq m \leq M_\xi}} \{\Delta V_{mat}[i,j,k,m]\} \leq \min_{\substack{t_{\xi+1}/\Delta\tau+l \leq k \leq N_t-l \\ 0 \leq m \leq M_{\xi+1}}} \{\Delta V_{mat}[i,j,k,m]\}, \tag{44}$$

i.e., the minimum velocity change to achieve the orbit transfer increases whenever $t_{\xi+1} > t_\xi$. Therefore,

$$h_{s_{\xi+1}}(\chi) \leq h(\chi'). \tag{45}$$

In other words, the minimum-cost tour along the matrix $\Delta V_{t_\xi}^{s_\xi+1}$ cannot exceed the corresponding cost $h(\chi')$ computed along the matrix $\Delta V_{t_{\xi+1}}$, which is computed at time $t_{\xi+1}$. Consequently, Equation (42) through Equations (43) and (45) yields

$$h(\chi) \leq \delta(\chi, \alpha, \chi') + h(\chi'). \tag{46}$$

3.6. ILP Formulation

This section introduces the ILP formulation of the time-dependent TSP. Based on the approach described in [11], the same formulation is revised and extended to include the operational constraints. All variables are binary, while the objective function and the constraints are linear. First, the following variables are defined:

- x_{ijkm} indicates if the transfer from orbit i to orbit j, departing at time τ_k and arriving at time τ_{k+m}, is performed ($x_{ijkm} = 1$) or not ($x_{ijkm} = 0$);
- y_i indicates whether ($y_i = 1$) or not ($y_i = 0$) object i is the last one visited;
- z_i indicates whether ($z_i = 1$) or not ($z_i = 0$) object i is the first one visited.

Under these premises, the ILP problem can be written as follows:

$$\min_{x,y,z} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=0}^{N_t} \sum_{m=0}^{M_k} \Delta V_{mat}[i,j,k,m] x_{ijkm}, \tag{47}$$

subject to the following constraints:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=0}^{N_t} \sum_{m=0}^{M_k} x_{ijkm} = n - 1, \tag{48}$$

$$\sum_{j=1}^{n} \sum_{k=0}^{N_t} \sum_{m=0}^{M_k} x_{ijkm} = 1 - y_i, \quad i = 0, ..., n, \tag{49}$$

$$\sum_{i=1}^{n} \sum_{k=0}^{N_t} \sum_{m=0}^{M_k} x_{ijkm} = 1 - z_j, \quad j = 0, ..., n, \tag{50}$$

$$\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}(k-l)x_{ijkm}-\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}kx_{jikm}\leq \tag{51}$$

$$\leq\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}mx_{jikm}+z_i(N_t-nl),\quad i=1,...,n,$$

$$\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}(k-l)x_{ijkm}-\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}kx_{jikm}\geq \tag{52}$$

$$\geq\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}mx_{jikm}-y_i(N_t-l)-lz_i,\quad i=1,...,n,$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=0}^{N_t}\sum_{m=0}^{M_k}mx_{ijkm}\leq N_t-(n-1)l, \tag{53}$$

$$\sum_{i=1}^{n}y_i=1, \tag{54}$$

$$\sum_{i=1}^{n}z_i=1. \tag{55}$$

Specifically, the following apply:

- Equation (48) establishes that $(n-1)$ orbit transfers are performed;
- Equation (49) ensures that the visiting spacecraft departs from each visited object exactly once, except for the last object;
- Equation (50) ensures that the spacecraft arrives at each object exactly once, except for the first one;
- Equations (51) and (52) guarantee that every orbit transfer has duration $m\Delta\tau$ ($m=0,...,M_k$) while incorporating the operation constraints:
  - the term $(k-l)$ accounts for a waiting period T_{wait} after reaching each object (except for the first one);
  - the term $z_i(N_t-nl)$ ensures that the first object is left at most at time $N_t-(n-1)l$;
  - the term $y_i(N_t-l)$ ensures that the last object is reached at most at time N_t-l;
  - the term $-lz_i$ takes into account that phasing and OOS operations are assumed to be completed before the spacecraft leaves the first object, and thus departure is also allowed at time $k=0$;
- Equation (53) constraints the total mission duration by guaranteeing that the sum of each orbit transfer $m\Delta\tau$ and each phasing time $l\Delta\tau$ does not exceed the total time $N_t\Delta\tau$;
- Equations (54) and (55) state that only one object can be the last or first visited RSO, respectively.

4. Numerical Results

The performance of the algorithm is assessed on an application case presented by Cerf [21]. This scenario involves a set of objects traveling along different orbits, as reported in Table 1. The only meaningful elements are given by the altitude, inclination, and initial RAAN, as phasing maneuvers are not addressed. The cleaning program proposed by Cerf consists of three missions, each visiting 5 objects over a duration of 45 months, for a total of 15 RSOs. In [21], a three-stage approach is employed to retrieve the optimal sequence: (i) first, all possible elementary transfers between objects are computed for a mesh of discretized dates and stored in cost matrices; (ii) then, the optimal sequence is retrieved through an SA algorithm spanning all possible transfers and dates through interpolation

of the cost matrices; (iii) finally, the solution is refined through nonlinear programming involving continuous variables. A fixed waiting duration of 5 days is allocated to each RSO to perform operations once the object is reached, while the duration of each orbit transfer ranges from 20 to 200 days. Tables 2–4 collect the final results of the optimization process for the three missions.

Table 1. Set of target orbits.

Object Number	Altitude (km)	Inclination (deg)	Initial RAAN (deg)
1	700	97.0	0
3	720	97.6	180
4	730	97.9	270
5	740	98.2	18
6	750	98.5	108
7	760	98.8	198
8	770	97.1	288
9	780	97.4	36
11	800	98.0	216
12	810	98.3	306
13	820	98.6	54
14	830	98.9	144
15	840	97.2	234
16	850	97.5	324
17	860	97.8	72
18	870	98.1	162
19	880	98.4	252
20	890	98.7	342
21	900	99.0	360

Table 2. Mission 1: optimal sequence and associated velocity change [21].

Object Number	Date [day]	ΔV [m/s]
16	3.1	287.1
20	183.1	210.8
21	389.3	202.2
5	513.6	111.0
17	545.4	-
Total		811.1

Table 3. Mission 2: optimal sequence and associated velocity change [21].

Object Number	Date [day]	ΔV [m/s]
15	552.7	141.5
3	563.3	291.8
14	781.7	132.2
11	823.0	146.4
8	935.8	-
Total		711.9

Table 4. Mission 3: optimal sequence and associated velocity change [21].

Object Number	Date [day]	ΔV [m/s]
1	942.1	119.2
4	976.8	411.8
9	1143.4	183.5
7	1177.3	70.6
12	1365.8	-
Total		785.1

To compare the performance of the strategy at hand, the A* algorithm is tested on each mission scenario. Specifically, the optimal sequence is searched within the three clouds of 5 RSOs. The index $N_{t,i}$ related to the maximum duration of each mission $i = 1, 2, 3$ is obtained by the associated initial time $t_{dep,i}$ and final time $t_{arr,i}$, as outlined by Tables 2–4,

$$N_{t,i} = \sup\left\{z \in \mathbb{Z} : z \leq \frac{t_{arr,i} + T_w - t_{dep,i}}{\Delta\tau}\right\}, \quad i = 1, 2, 3, \tag{56}$$

where the waiting T_w is added to allow for the servicing operations when the final object of each mission is reached. To allow a fair comparison, different discretization grids are adopted, namely $\Delta\tau = 3, 5, 7, 10$ days. For each scenario, the time allowed for phasing, docking, and servicing operations is set to $T_w = \Delta\tau$ as well, meaning that $l = 1$. The maximum duration of each transfer is set to 200 days, meaning that the maximum duration index is set to

$$M_{max} = \sup\left\{z \in \mathbb{Z} : z \leq \frac{200 \text{ days}}{\Delta\tau}\right\}. \tag{57}$$

The resulting numerical setup is reported in Table 5, where $\mathcal{N}$ is the dimension of the search space (i.e., $N \times (N-1) \times N_{t,i} \times M_{max}$). The computation of the cost matrix is parallelized on different processors and performed on an Apple M2 Pro (32 GB of RAM, 10 cores: 4 high-efficiency + 6 high-performance), and the associated wall-clock time (WCT) is reported. To limit the search space of optimal orbit transfers, the altitude of the drift orbits is limited between 400 and 2000 km. Furthermore, the solutions of each minimization problem are accepted if the constraint on the RAAN is satisfied within an accuracy of 1 deg.

Table 5. Numerical setup of the A* search algorithm.

	$\Delta\tau = 3$ d	$\Delta\tau = 5$ d	$\Delta\tau = 7$ d	$\Delta\tau = 10$ d
Mission 1				
N_t	181	109	78	55
$M_{\max}$	66	40	28	20
$\mathcal{N}$	238,920	87,200	43,680	22,000
WCT [s]	475	189	112	60
Mission 2				
N_t	128	77	55	39
$M_{\max}$	66	40	28	20
$\mathcal{N}$	168,960	61,600	30,800	15,600
WCT [s]	318	118	61	34
Mission 3				
N_t	142	85	61	43
$M_{\max}$	66	40	28	20
$\mathcal{N}$	187,440	68,000	34,160	16,800
WCT [s]	340	155	78	45

Once all orbit transfers are computed and stored in the cost matrices, the three missions are optimized through the A* algorithm. Table 6 collects the results obtained under the different discretization grids, along with the associated wall-clock time.

Table 6. Comparison between overall velocity change and computational time across different discretization grids using A*.

	Mission 1		**Mission 2**		**Mission 3**	
	ΔV [m/s]	**WCT [s]**	**ΔV [m/s]**	**WCT [s]**	**ΔV [m/s]**	**WCT [s]**
$\Delta\tau = 3$ d	741.6	0.97	619.8	1.03	742.2	15.95
$\Delta\tau = 5$ d	744.6	0.76	625.4	0.63	747.5	2.99
$\Delta\tau = 7$ d	753.4	0.76	633.5	0.77	757.6	1.45
$\Delta\tau = 10$ d	754.7	0.79	635.4	0.75	759.0	0.96

Apparently, no significant differences emerge across the different discretization steps in terms of computational time for the first two missions. Instead, the third mission exhibits a moderate increase in the wall-clock time when the finest mesh grid is adopted. On the other hand, the velocity change generally exhibits a significant increase when larger time steps $\Delta\tau$ are selected, while the optimal sequence does not change. However, all computed ΔV values are lower than the values reported in Tables 2–4. Moreover, even if coarser time steps may yield suboptimal solutions, these can be further refined in a subsequent optimization stage employing continuous variables [21]. Therefore, larger $\Delta\tau$ values may be adopted within a preliminary optimization framework as long as the visiting order is not altered.

Table 7 collects the results obtained by solving the ILP problem. Even if the ILP formulation allows obtaining the same solutions across all mission scenarios, the comparison between Tables 6 and 7 reveals that the proposed A* approach enables a considerable reduction in the computational time, regardless of the discretization step. This is evident from inspection of Figure 2, which compares the WCT associated with both the A* and the ILP approach.

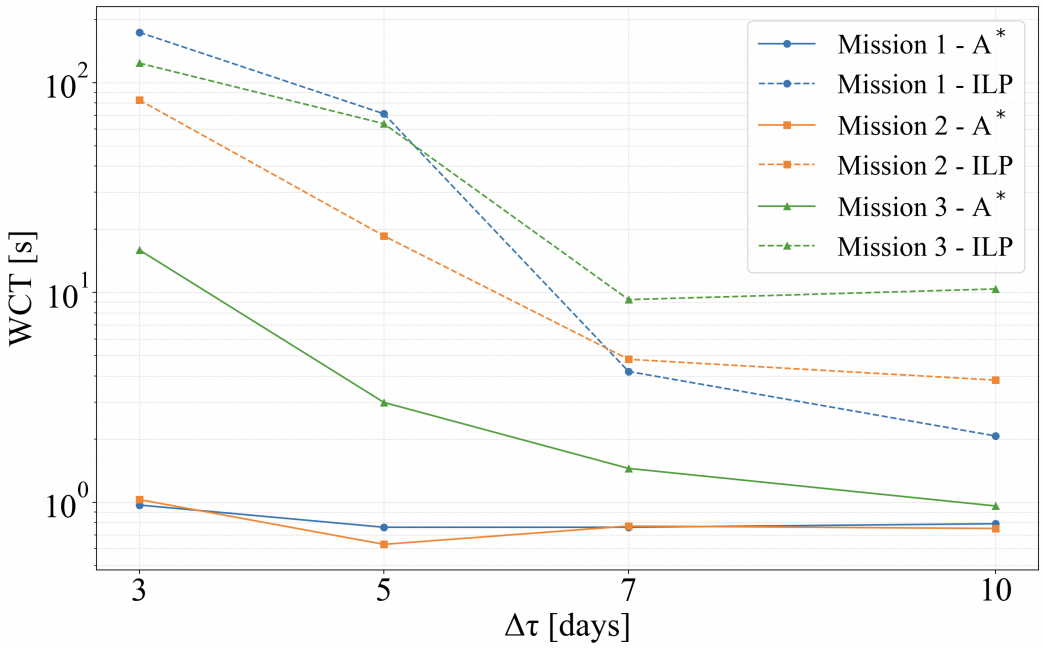


Figure 2. Comparison between the WCT associated with either A* or ILP across the three missions. A logarithmic scale is employed along the *y* axis.

Table 7. Results obtained through the ILP solver.

	Mission 1		**Mission 2**		**Mission 3**	
	ΔV [m/s]	**WCT [s]**	**ΔV [m/s]**	**WCT [s]**	**ΔV [m/s]**	**WCT [s]**
$\Delta\tau = 3$ d	741.6	173.63	619.8	82.67	742.2	123.85
$\Delta\tau = 5$ d	744.6	71.00	625.4	18.59	747.5	63.69
$\Delta\tau = 7$ d	753.4	4.20	633.5	4.80	757.6	9.24
$\Delta\tau = 10$ d	754.7	2.07	635.4	3.82	759.0	10.39

In order to better understand the behavior of the algorithm at hand, Tables 8–10 collect the results obtained through the A* search algorithm under the time step $\Delta\tau = 5$ days. For each mission, both the elapsed time from the beginning of the mission t and the time $\tilde{t}$ from the initial epoch are reported. Furthermore, the RAAN error $\Delta\Omega$ at the end of each transfer is reported to assess the results of the constrained optimization process within the inner layer. Evidently, this value is meaningless for direct transfers (highlighted in blue), as no optimization is required.

Table 8. Mission 1: optimal sequence through A* algorithm ($\Delta\tau = 5$ days).

Transfer	**Departure Epoch [day]**		**Arrival Epoch [day]**		**$\Delta\Omega$ [deg]**	**ΔV [m/s]**
	t_{dep}	$\tilde{t}_{dep}$	t_{arr}	$\tilde{t}_{arr}$		
16–20	0	3.1	185	188.1	6.8×10^{-12}	276.8
20–21	190	193.1	390	393.1	2.8×10^{-12}	203.9
21–5	395	398.1	535	538.1	0.0	181.5
5–17	540	543.1	540	543.1	/	82.3
Total						744.6

Table 9. Mission 2: optimal sequence through A* algorithm ($\Delta\tau = 5$ days).

Transfer	**Departure Epoch [day]**		**Arrival Epoch [day]**		**$\Delta\Omega$ [deg]**	**ΔV [m/s]**
	t_{dep}	$\tilde{t}_{dep}$	t_{arr}	$\tilde{t}_{arr}$		
15–3	0	552.7	5	557.7	3.2×10^{-1}	81.4
3–14	10	562.7	210	762.7	2.6×10^{-12}	272.5
14–11	215	767.7	265	817.7	6.3×10^{-12}	125.3
11–8	270	822.7	380	932.7	7.3×10^{-12}	146.2
Total						625.4

Table 10. Mission 3: optimal sequence through A* algorithm ($\Delta\tau = 5$ days).

Transfer	**Departure Epoch [day]**		**Arrival Epoch [day]**		**$\Delta\Omega$ [deg]**	**ΔV [m/s]**
	t_{dep}	$\tilde{t}_{dep}$	t_{arr}	$\tilde{t}_{arr}$		
1–4	0	942.1	75	1017.1	6.9×10^{-13}	118.8
4–9	80	1022.1	280	1222.1	1.2×10^{-14}	277.7
9–7	285	1227.1	285	1227.1	/	182.8
7–12	290	1232.1	420	1362.1	6.1×10^{-13}	168.2
Total						747.5

From comparison between Tables 2–4 and Tables 8–10, it is evident the two approaches yield the same solution in terms of the sequence of visited RSOs. However, the timing of the visited objects is different, yielding a significant reduction in the overall velocity change. For instance, two short-duration legs in the solution proposed by Cerf [21] (namely legs 5–17 and 9–7 in Tables 2–4) are instead replaced by direct transfers in the approach at

hand, i.e., without leveraging intermediate drift orbits. Remarkably, the velocity change associated with the second mission scenario (see Table 9) is significantly lower than the one reported in Table 3. The comparison between the two tables reveals that the largest propellant saving occurs during the first leg (i.e., the orbit transfer between objects 15 and 3). This transfer is associated with the largest RAAN error achieved through the constrained optimization process (namely 3.2×10^{-1} deg). Even if this is still consistent with the prescribed accuracy (i.e., 1 deg), alternative solutions may be obtained by adopting tighter constraints on the RAAN error, with a consequent increase in the overall ΔV. However, the adoption of the discretized approach enables the identification of an optimal sequence and timing, which can be subsequently refined though continuous variables.

Scalability Assessment

This section addresses the scalability of the proposed algorithm through larger sets of RSOs. As a matter of fact, even if the A* algorithm is cost-optimal and optimally efficient, the number of expanded nodes can be exponential with the dimension of the set, and thus the A* algorithm fails due to memory limitations [28].

To this aim, the A* and ILP algorithms are tested through a series of simulations with subsets of RSOs increasing in size. The results are compared in terms of computational time required for the preprocessing of the cost matrix ($T_{\Delta V_{mat}}$) for the A* algorithm (T_{A^*}) and for the ILP algorithm (T_{ILP}), as well as number of nodes in the open and closed sets. The simulations are run on an Intel(R) Core(TM) i9-10980XE CPU @3.00 GHz (192 GB of RAM, 18 cores). The number of RSOs sequentially increases from 3 to 10 to include all objects visited along Missions 1 and 2. For each simulation, the time step $\Delta\tau$ is set to 10 days, while the total mission duration is 940 days (i.e., $N_t = 94$). Finally, the waiting period is $T_{wait} = 10$ days, yielding $l = 1$.

As expected, the analysis of Table 11 and Figure 3 shows that the performance of the algorithm is limited when larger sets are considered. In particular, the WCT associated with the preprocessing time is not heavily affected by the number of RSOs, as the computation of the cost matrix can be distributed on parallel cores. Instead, both the computational time of the A* algorithm and the number of nodes significantly increase with the dimension of the set, so that the algorithm is unable to proceed due to memory limitations when sets with more than 10 RSOs are considered with the given numerical setup and hardware. This is an inherent limitation when optimality is sought. However, the A* approach generally outperforms the ILP solver in terms of computational time. Moreover, the ILP solver is not capable of concluding the optimization process even with 10 RSOs. Memory-bounded variants of the A* algorithm may be investigated to enable a more efficient memory usage [28]. Alternatively, different formulation of the heuristic function may allow better estimating the remaining cost of the mission and guiding node expansion more effectively.

Table 11. Results of the scalability study.

Number of RSOs	WCT [s]			Number of Nodes		ΔV [km/s]
	$T_{\Delta V_{mat}}$	T_{A^*}	T_{ILP}	Closed Set	Open Set	
3	24.78	1.11	0.56	7.00×10^0	4.10×10^2	0.4749
4	45.54	1.19	2.38	2.00×10^1	8.83×10^2	0.6303
5	74.29	1.25	15.18	1.48×10^2	6.09×10^3	0.7466
6	108.38	2.09	48.50	2.29×10^3	9.03×10^4	1.0124
7	150.40	6.27	551.41	1.13×10^4	4.58×10^5	1.0275
8	199.91	29.11	622.95	5.28×10^4	2.76×10^6	1.0546
9	256.10	1071.61	/	1.76×10^6	9.57×10^7	1.2911
10	325.61	4910.03	/	7.18×10^6	4.21×10^8	1.5598

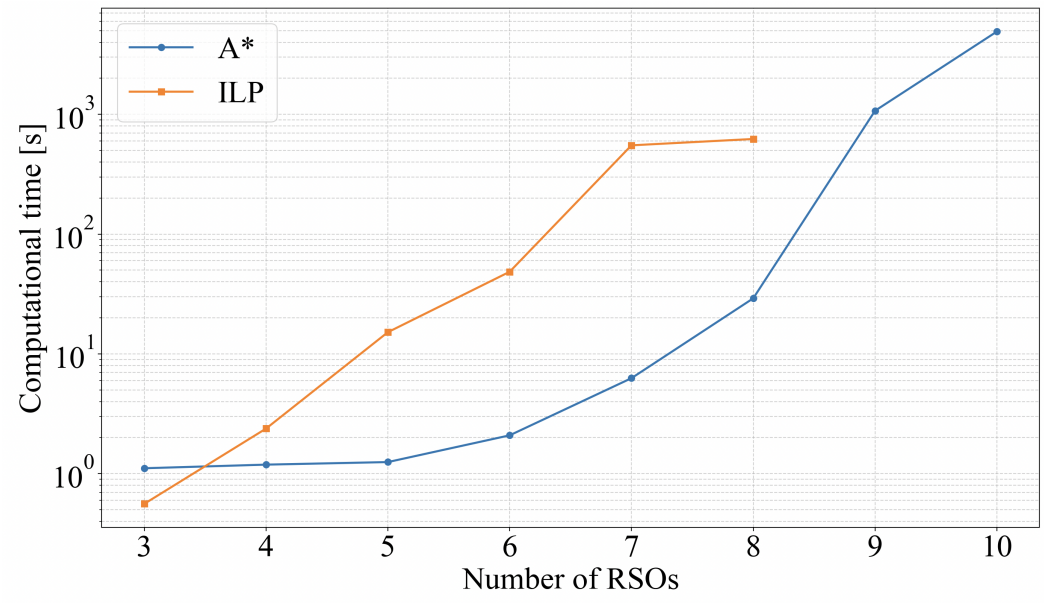


Figure 3. Computational time vs. number of RSOs.

5. Conclusions

Path planning for OOS missions provides key insights on how to tackle the complexity involved in large combinatorial problems, highlighting the trade-off between optimality and efficiency. This research addresses the development of an algorithm for OOS mission design, where a servicing spacecraft intends to perform multiple orbit transfers with minimum fuel consumption. The target RSOs are placed on non-coplanar orbits, subject to precession due to the J_2 harmonic of the gravitational field. A two-stage methodology is employed to decouple the computation of the optimal orbit transfers from the optimization of the sequence and timing of visited objects. First, a mesh of discretized epochs is created to compute the optimal transfers between all pairs of n orbits and times, and the resulting velocity changes are stored in cost matrices. Specifically, the orbit transfer strategy leverages the use of both direct transfers (when orbital planes are coincident) and intermediate drift orbits to increase the differential rate due to the Earth's oblateness. Then, an A* algorithm explores the solution space to obtain the optimal sequence and associated rendezvous times. The search strategy employs a heuristic function based on a modified version of the Held–Karp algorithm, which is proven to be admissible and consistent, ensuring convergence to the optimal solution. Operational constraints are also included by allowing a suitable waiting time to perform phasing and servicing activities. The numerical application investigates a case study taken from the previous literature, including three different missions in the LEO environment, aimed at visiting a set of 5 RSOs each. Several numerical setups are tested, associated with different discretization steps. In all cases, the A* algorithm proves to achieve a lower overall velocity change with respect to the reference solution through the use of both direct transfers and intermediate drift orbits. Moreover, an ILP formulation of the same problem is introduced to assess the performance of the strategy at hand. In all cases, the same optimal solutions are obtained, but with a significant increase in the computational time.

Even if the proposed algorithm provides an efficient approach to OOS mission design, its scalability to larger sets of objects remains limited. As a matter of fact, the scalability study shows that, when sets of more than 10 RSOs are considered, the A* algorithm is unable to proceed due to memory limitations. While memory-bounded variants of the A* algorithm may be developed, future research will also investigate the design of refined heuristic functions to drive the exploration of nodes. To this aim, two different

strategies are currently under investigation: (i) a heuristic estimation that leverages the optimal solution of the A* algorithm within smaller subsets of elements to bridge the gap between the predicted and actual cost of the remaining sequence, and (ii) a neural network-enhanced heuristic function trained with the optimal solutions of the A* algorithm. However, admissibility should be carefully assessed to preserve the optimality requirement.

The technique presented in this paper is rather general and does not refer to any specific servicing operation. Anyway, the proposed framework can be accordingly extended to include deorbiting (i.e., transfers toward disposal orbits) or station-keeping maneuvers. In these cases, the target orbits can be considered as additional elements within the set of visited objects, and the same methodology can be effectively employed.

Author Contributions: Conceptualization, E.M.L., F.C., L.F., and M.P.; methodology, E.M.L., L.F., and M.P.; software, E.M.L.; validation, E.M.L.; formal analysis, E.M.L. and L.F.; investigation, E.M.L.; resources, E.M.L. and M.P.; data curation, E.M.L.; writing—original draft preparation, E.M.L.; writing—review and editing, E.M.L., L.F., F.C., and M.P.; visualization, L.F., F.C., and M.P.; supervision, L.F., F.C., and M.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This paper contains all the data needed for reproducing the numerical results.

Acknowledgments: During the preparation of this study, the authors used GenAI tools (ChatGPT 5.0) for the purposes of generating a basic structure of an A* search algorithm and a Held–Karp algorithm. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADR	Active-Debris-Removal
ILP	Integer Linear Programming
LEO	Low-Earth-Orbit
OOS	On-Orbit Servicing
RAAN	Right Ascension of the Ascending Node
RSO	Resident Space Object
SA	Simulated Annealing
TSP	Traveling Salesman Problem
WCT	Wall-clock time

References

1. Luu, M.A.; Hastings, D.E. On-orbit servicing system architectures for proliferated low-earth-orbit constellations. *J. Spacecr. Rocket.* **2022**, *59*, 1946–1965.
2. Grumman, N. Mission Extension Vehicle (MEV). 2021. Available online: https://cdn.northropgrumman.com/-/media/wp-content/uploads/Mission-Extension-Vehicle-MEV-fact-sheet.pdf?v=1.0.0 (accessed on 1 June 2026).
3. ESA. ESA to Build First In-Orbit Servicing Mission with D-Orbit. 2024. Available online: https://www.esa.int/Space_Safety/ESA_to_build_first_in-orbit_servicing_mission_with_D-Orbit (accessed on 22 January 2026).
4. National Aeronautics and Space Administration. *On-Orbit Satellite Servicing Study Project Report*; Technical Report; Goddard Space Flight Center: Greenbelt, MD, USA , 2010.

5. Opromolla, R.; Grishko, D.; Auburn, J.; Bevilacqua, R.; Buinhas, L.; Cassady, J.; Jäger, M.; Jankovic, M.; Rodriguez, J.; Perino, M.A.; et al. Future in-orbit servicing operations in the space traffic management context. *Acta Astronaut.* **2024**, *220*, 469–477. [CrossRef]
6. Hou, L.; Misra, A. Traveling Salesman Problem of optimal debris removal sequence using non-population gradient search. *Acta Astronaut.* **2024**, *215*, 373–386.
7. Shen, H.; Tsiotras, P. Optimal scheduling for servicing multiple satellites in a circular constellation. In Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Monterey, CA, USA, 5–8 August 2002; p. 4907.
8. Jorgensen, M.K.; Sharf, I. Optimal planning for a multiple space debris removal mission using high-accuracy low-thrust transfers. *Acta Astronaut.* **2020**, *172*, 56–69.
9. Baranov, A.A.; Grishko, D.A. Review of path planning in prospective multi-target active debris removal missions in low earth orbits. *Prog. Aerosp. Sci.* **2024**, *145*, 100982. [CrossRef]
10. Barea, A.; Urrutxua, H.; Cadarso, L. Large-scale object selection and trajectory planning for multi-target space debris removal missions. *Acta Astronaut.* **2020**, *170*, 289–301.
11. Federici, L.; Zavoli, A.; Colasurdo, G. On the use of A* search for active debris removal mission planning. *J. Space Saf. Eng.* **2021**, *8*, 245–255. [CrossRef]
12. Medioni, L.; Gary, Y.; Monclin, M.; Oosterhof, C.; Pierre, G.; Semblanet, T.; Comte, P.; Nocentini, K. Trajectory optimization for multi-target Active Debris Removal missions. *Adv. Space Res.* **2023**, *72*, 2801–2823. [CrossRef]
13. Ciccotti, C.; Leonardi, E.M.; Pontani, M. Efficient On-Orbit Servicing via Routing Optimization and Low-Thrust Propulsion. In Proceedings of the 76th International Astronautical Congress (IAC), Sydney, Australia, 29 September–3 October 2025.
14. Federici, L.; Zavoli, A.; Colasurdo, G. Impulsive multi-rendezvous trajectory design and optimization. In Proceedings of the 8th European Conference for Aeronautics and Space Sciences (EUCASS), Madrid, Spain, 1–4 July 2019; pp. 1–4.
15. Wang, D.; Li, L.; Chen, L. An efficient genetic algorithm for active space debris removal planning. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 514–521.
16. Viavattene, G.; Devereux, E.; Snelling, D.; Payne, N.; Wokes, S.; Ceriotti, M. Design of multiple space debris removal missions using machine learning. *Acta Astronaut.* **2022**, *193*, 277–286. [CrossRef]
17. Yang, J.; Hu, Y.H.; Liu, Y.; Hou, X.; Pan, Q. On the application of reinforcement learning in multi-debris active removal mission planning. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 605–610.
18. Chen, S.; Jiang, F.; Li, H.; Baoyin, H. Optimization for multitarget, multispacecraft impulsive rendezvous considering J 2 perturbation. *J. Guid. Control. Dyn.* **2021**, *44*, 1811–1822.
19. Bourjolly, J.M.; Gurtuna, O.; Lyngvi, A. On-orbit servicing: A time-dependent, moving-target traveling salesman problem. *Int. Trans. Oper. Res.* **2006**, *13*, 461–481. [CrossRef]
20. Izzo, D.; Getzner, I.; Hennes, D.; Simões, L.F. Evolving solutions to TSP variants for active space debris removal. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 1207–1214.
21. Cerf, M. Multiple space debris collecting mission: Optimal mission planning. *J. Optim. Theory Appl.* **2015**, *167*, 195–218. [CrossRef]
22. Ciccotti, C.; Leonardi, E.M.; Pontani, M. On-orbit-servicing via routing optimization using high-thrust or low-thrust propulsion. *Acta Astronaut.* **2026**, *246*, 760–777.
23. Held, M.; Karp, R.M. A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **1962**, *10*, 196–210. [CrossRef]
24. Fortescue, P.; Swinerd, G.; Stark, J. *Spacecraft Systems Engineering*, 3rd ed.; John Wiley & Sons Ltd.: Chichester, UK, 2003.
25. Cerf, M. Multiple space debris collecting mission—debris selection and trajectory optimization. *J. Optim. Theory Appl.* **2013**, *156*, 761–796.
26. Curtis, H.D. *Orbital Mechanics for Engineering Students*, 3rd ed.; Butterworth-Heinemann: Oxford, UK, 2014.
27. Vallado, D.A. *Fundamentals of Astrodynamics and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2001; Volume 12.
28. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 4th ed.; Pearson Education: London, UK, 2021.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.