IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 March 2023, accepted 14 April 2023, date of publication 21 April 2023, date of current version 1 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3269305

RESEARCH ARTICLE

Evolutionary Optimization for Active Debris Removal Mission Planning

DANILO ZONA[1], ALESSANDRO ZAVOLI[2], LORENZO FEDERICI[3], AND GIULIO AVANZINI[1]
[1]Department of Engineering for Innovation, Università del Salento, 73100 Lecce, Italy
[2]Department of Mechanical and Aerospace Engineering, Sapienza University of Rome, 00184 Rome, Italy
[3]Department of Systems and Industrial Engineering, The University of Arizona, Tucson, AZ 85721, USA

Corresponding author: Alessandro Zavoli (alessandro.zavoli@uniroma1.it)

ABSTRACT Active debris removal missions require an accurate planning for maximizing mission payout, by reaching the maximum number of potential orbiting targets in a given region of space. Such a problem is known to be computationally demanding and the present paper provides a technique for preliminary mission planning based on a novel evolutionary optimization algorithm, which identifies the best sequence of debris to be captured and/or deorbited. A permutation-based encoding is introduced, which may handle multiple spacecraft trajectories. An original archipelago structure is also adopted for improving algorithm capabilities to explore the search space. As a further contribution, several crossover and mutation operators and migration schemes are tested in order to identify the best set of algorithm parameters for the considered class of optimization problems. The algorithm is numerically tested for a fictitious cloud of debris in the neighborhood of Sun-synchronous orbits, including cases with multiple chasers.

INDEX TERMS Active debris removal, evolutionary optimization algorithms, space mission design.

I. INTRODUCTION

In recent years, the problem of space debris is worrying the whole aerospace community as the sustainability of the extra-atmospheric environment is threatened by the increasing amount of ''space garbage'' now orbiting Earth. Several measures aimed at mitigating the creation of new debris are being undertaken, including guidelines on post-mission disposal (PMD) of new satellites. Unfortunately, the number of debris in orbit, already very high, is steadily increasing, empirically proving that PMD procedures are not sufficient to mitigate the problem, not even in the long run.

In such a scenario, the problem of space debris may eventually evolve dramatically, as predicted by the so-called Kessler syndrome [1], a cascade effect where, starting from a single fragmentation event due to an explosion or a collision, the fragments generated are going to collide with an increasing number of active and inactive orbiting objects. The number of fragments would exponentially increase and be spread by orbital perturbations over entire regions of space, possibly making them no longer accessible, thus harming or possibly preventing all space activities.

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Xu.

Some authors claim that Active Debris Removal (ADR) is the only possible solution to effectively mitigate the problem [2]. The goal of an ADR mission is to carry out deorbiting operations over each debris ('target') in a prescribed set, using one or more dedicated active spacecraft ('chaser'). The optimization of the trajectory flown by each chaser allows them to make the best use of the stowed propellant to deorbit as many targets as possible, making ADR missions economically affordable. Since each chaser needs to perform a series of rendezvous with different targets, they need to fly a so-called multi-rendezvous trajectory. Each arc of a multi-rendezvous trajectory represents a transfer between two successive debris of the target sequence. Mission optimization requires finding the optimal encounter sequence and the encounter epochs of the debris to deorbit, where the merit function is a measure of the payoff of the whole mission and a constraint is present on the maximum ΔV available to each chaser.

The multi-rendezvous trajectory optimization problem features two tightly coupled layers, namely: i) an outer layer, which concerns the search for the optimal encounter sequence and epochs; and ii) an inner-layer, in which each single target-to-target transfer needs to be optimized. This formulation leads to a Mixed-Integer NonLinear Programming (MINLP) problem, which involves both real variables and integers.

Real-valued parameters are related to the optimization of transfer legs, integer numbers refer to the search of the optimal encounter sequence. Since MINLP problems are known to be some of the hardest NP-hard problems, no solution method guarantees an optimal solution in a reasonable amount of time. For this reason, the most popular approach for the problem at hand is to disconnect the two layers and solve them separately. In this way, it is possible to obtain near-optimal solutions in an acceptable amount of computation time.

The idea at the basis of the so-called bi-level approach is to find an estimate of the single target-to-target transfer cost using a computationally inexpensive heuristic. Moreover, pre-determined encounter epochs are usually employed in such a way that it is possible to store all the costs in a tensor and speed up the search. It is then possible to focus on the solution of the outer-layer problem defining the encounter sequence and epochs while evaluating the cost of every transfer. The strategies used to estimate the transfer cost can all be traced back to one of two basic ideas: i) if the overall mission duration is long enough (free-time missions), the J_2 perturbation is exploited in order to obtain a free alignment of the orbital planes before starting the rendezvous maneuver; ii) if mission duration is limited, then the transfer problem can be posed as a single-target time-fixed rendezvous problem [3], [4], and one has to pay an extra ΔV for correcting the difference in the right ascension of the ascending node (RAAN) between the orbit planes of chaser and target at the beginning of the transfer. In the present case, long mission times (up to approximately two years) are considered, enabling the use of auxiliary drift orbits to reduce the transfer cost associated to the orbit plane change.

The solution of the outer-layer problem is significantly more difficult. Given the transfer costs, the problem of finding the encounter sequence and epochs can be read as a routing problem, similar to the Traveling Salesman Problem (TSP). The classic formulation of the TSP requires that an agent visits all the prescribed cities in a map, searching for the tour which minimizes the total distance traveled. The distance between two cities, which represents the cost of the transfer, is considered fixed in time, but if the agent is required to visit moving targets, a variant to the classical TSP is introduced, in which the distance between two targets on the map changes with time. This is the so-called Time-Dependant Traveling Salesman Problem (TDTSP). The analogy with the problem here considered is evident, if one assumes the debris as the targets to visit, whereas the time-varying cost of the transfer that the chaser pays to move from one target to the following one in the series is represented by an estimate of the fuel required for the corresponding orbit transfer maneuver. With this analogy in mind, it is possible to recast the outer-layer problem as a combinatorial problem with the objective of finding the tour with the minimum cost in term of ΔV or maximizing some performance index which measures the total mission payoff, for a given ΔV available.

Combinatorial problems are difficult to solve because when the dimension of the problem increases, the number of possible combinations can easily reach a billion and over. That's why enumeration algorithms have been exploited in very few works, even if they guarantee to find the optimal solution. Authors have thus turned their attention towards metaheuristics algorithms, which usually do not guarantee the optimality of the solution, but can find good solutions in a reasonable amount of computational time. Eventually, they can find the optimal solution if the problem is sufficiently easy or if the available computational time is sufficiently long. Among the many, Simulated Annealing (SA) [5], Ant Colony Optimization (ACO) [6], Beam Search [7] and Genetic Algorithms (GAs) [8], [9], [10] have been successfully exploited to solve the combinatorial ADR mission planning problem.

The aim of this work is to introduce a general solution method for ADR mission planning in which GAs are used to perform a global optimization in order to find the encounter sequence and the encounter epochs of the selected debris. The choice of GAs is dictated by their flexibility, which allows them to adapt to different formulations of the combinatorial problem thanks to a simple encoding. Moreover, their stochastic nature makes them particularly able to escape local optimal solutions. Our versions of GA are equipped with particular permutation-preserving operators specifically designed to handle combinatorial problems. Also, a distributed genetic algorithm (dGA) variant is presented, which allows for (parallel) distributed computing over a multiprocessor cluster. The distributed topology, in which the populations is divided into smaller subpopulations, was initially introduced for Particle Swarm Optimization (PSO) and Differential Evolution (DE) [11]. Later, it was extended to other metaheuristic algorithms and, in particular, to genetic algorithms. The efficiency and effectiveness of distributed genetic algorithms were demonstrated by Muhlenbein et al. [12] who applied a parallel distribution version of GA to different function optimization problems that are commonly employed to test optimization algorithms performances.

The proposed method makes use of cost tensors that contain all the pre-determined transfer costs obtained by means of a simple analytical heuristic to rapidly evaluate every single body-to-body transfer cost during the combinatorial problem solution [13]. The outer-layer combinatorial problem is then solved by exploiting a GA. In particular, the solution method allows us to plan time-constrained multi-chaser ADR missions with the objective of deorbiting all the selected debris moving on (nearly) sun-synchronous orbits with similar altitudes and inclinations [14].

The paper is organized as follows. First, the ADR mission problem is stated, highlighting the main features of a time-constrained rendezvous maneuver and of a multi-rendezvous trajectory. In Section III the optimization procedure is explained in detail. Then, the single-leg transfer cost estimate strategy is presented. The genetic algorithms adopted are presented in Section III-C. Numerical results for test cases based on both fictitious and real targets are reported

in Section IV-A. A section of concluding remarks completes the paper.

II. PROBLEM STATEMENT

The chaser is a specially designed active spacecraft, with the capability of deorbiting the debris that it visits. In a classical ADR mission, the chaser is required to reach and deorbit a certain number of targets, represented by passive objects that lay on orbits threatening other active spacecraft. Usually the debris are clustered in clouds of many objects (tens or even hundreds). Clearly, the choice of which debris to remove is not unequivocally defined.

Removing as many fragments as possible can be considered as the simplest objective of an ADR mission, but it is also possible to rank debris fragments on the basis of their potential threat to other orbiting objects, their mass or their dimensions. A score accounting for all those characteristics can be assigned to each debris. However, in our case, the choice of what debris to remove is left to the user. The number of debris that a chaser can visit and deorbit, in fact, is limited to a few by both the stewed propellant and the deorbiting kits than it can carry. Consequently, if the number of debris fragments in the clouds is too large, a single chaser is not sufficient to deorbit all of them.

In a more complex scenario, one can think of using multiple chasers that act either simultaneously or in a sequence, in order to clean the whole debris cloud. In this framework, it is possible to search for the optimal plan which makes use of several coordinated missions, such that the overall ΔV is minimized. This strategy usually provides better solutions with respect to single missions solved independently from each other, yet it introduces another layer of complexity represented by the optimal split of the targets among the chasers. The optimization of a multi-chaser ADR mission thus consists in finding not only the optimal split of the targets among the chasers, but also the optimal encounter sequence and epochs in which each chaser must visit the selected targets.

Obviously, in order to find the optimal solution, each transfer problem needs also to be optimized. The goal is to minimize the overall ΔV that the chasers need to fly the multi-rendezvous trajectories. More formally, let us consider a multi-chaser ADR mission, where N_c chaser spacecraft must visit and deorbit a complete set of N_s space debris, numbered from 1 to N_s. Let $\boldsymbol{p}^{(i)} = \{p_1^{(i)}, p_2^{(i)}, \ldots, p_{N_s^{(i)}}^{(i)}\}$ be a sequence of $N_s^{(i)} \leq N_s$ debris visited by the i-th chaser, and $\boldsymbol{t}^{(i)} = \{t_1^{(i)}, t_2^{(i)}, \ldots, t_{N_s^{(i)}}^{(i)}\}$ the vector of the corresponding encounter epochs.

For the i-th chaser, the cost associated with the k-th leg of its multi-rendezvous trajectory, that is, the cost for moving from debris $p_k^{(i)}$ to debris $p_{k+1}^{(i)}$ departing at time $t_k^{(i)}$ and arriving a time $t_{k+1}^{(i)}$, can be written as $\Delta V_k^{(i)} = f\left(p_k^{(i)}, p_{k+1}^{(i)}, t_k^{(i)}, t_{k+1}^{(i)}, \boldsymbol{\Theta}_k^{(i)}\right)$, where the vector $\boldsymbol{\Theta}_k^{(i)}$ lists all parameters needed to describe the k–th rendezvous leg of the i–th chaser (e.g., position and components of the velocity impulses in case of an impulsive transfer).

The total ΔV required by the i-th chaser is equal to

$$\Delta V^{(i)} = \sum_{k=1}^{N_s^{(i)}-1} \Delta V_k^{(i)}. \tag{1}$$

As done in [14], we assumed that each chaser starts the tour directly from the first target visited, i.e., it is injected directly in rendezvous conditions with the first target and not on a parking orbit. This also implies that the cost of visiting the first target is zero for each chaser.

The introduction of constraints for the coordination of the chasers becomes necessary when studying missions with multiple chasers. Depending on mission requirements, the chasers may be asked to work either simultaneously or in non-overlapping time windows. Considering that the flying time-window of the i-th chaser starts at epoch $t_1^{(i)}$ and ends at epoch $t_{N_s^{(i)}}^{(i)}$, the condition of non-overlapping operative time-windows can be enforced by introducing the constraint:

$$t_1^{(i+1)} > t_{N_s^{(i)}}^{(i)}, \qquad \forall i = 1, \ldots, N_c - 1 \tag{2}$$

Moreover, since we are considering a time-constrained mission, the final rendezvous time must be lower than the maximum mission time T_{max}:

$$t_{N_s^{(N_c)}}^{(N_c)} \leq T_{max} \tag{3}$$

When a sufficiently large number of chasers is considered, the optimal mission plan may lead to an uneven split of the mission cost among the chasers, which might be a concern from an operational point of view. To avoid this issue, an additional constraint may be introduced, limiting the maximum velocity increment that each chaser can perform, that is:

$$\Delta V^{(i)} \leq \Delta V_{max}^{(i)}, \qquad \forall i = 1, \ldots, N_c \tag{4}$$

where $\Delta V_{max}^{(i)}$ is the maximum ΔV that can be performed by the i-th chaser. According to the mission conditions, the flexibility of the proposed formulation allows for the implementation of more complex and articulated fuel-consumption constraints.

The ADR mission planning problem can be formulated as

$$\mathcal{P}: \begin{cases} \min\limits_{\boldsymbol{x}} \sum\limits_{i=1}^{N_c} \Delta V^{(i)} \\ \text{s.t.}: \; p_k^{(i)} \in \{1, \ldots, N_s\}, & \forall k = 1, \ldots, N_s^{(i)}, \\ \qquad \forall i = 1, \ldots, N_c \\ p_k^{(i)} \neq p_j^{(i)}, & \forall k \neq j = 1, \ldots, N_s^{(i)}, \\ \qquad \forall i = 1, \ldots, N_c \\ \boldsymbol{p}^{(i)} \cap \boldsymbol{p}^{(j)} = \emptyset, & \forall i \neq j = 1, \ldots, N_c, \\ \qquad \forall i = 1, \ldots, N_c \\ \sum\limits_{i=1}^{N_c} N_s^{(i)} = N_s \end{cases} \tag{5}$$

where the vector of design variables $\boldsymbol{x}$ encompasses the encounter sequences $\boldsymbol{p}^{(i)}$ and epochs $\boldsymbol{t}^{(i)}$ of the targets visited by each chaser and the parameters included in the vectors $\boldsymbol{\Theta}^{(i)} = \left(\boldsymbol{\Theta}_1^{(i)T}, \boldsymbol{\Theta}_2^{(i)T}, \ldots, \boldsymbol{\Theta}_{N_s^{(i)}}^{(i)T}\right)^T$,

$$\boldsymbol{x} = \bigcup_{i=1}^{N_c} \{\boldsymbol{p}^{(i)}, \boldsymbol{t}^{(i)}, \boldsymbol{\Theta}^{(i)}\}. \tag{6}$$

III. SOLUTION METHOD

In the previous section we formulated the original ADR mission planning problem as a MINLP problem, which is usually impossible to solve in practice. In many cases, also achieving an acceptable solution in a limited amount of computation time can be difficult. We now introduce the solution method that we specifically developed to facilitate the original problem, solving it in a reasonable time. The approach consists in splitting the problem in (possibly easier) sub-problems that are solved separately. The solutions of these sub-problems are then merged to obtain a (hopefully) good solution of the original problem.

As discussed in the Introduction, the search for the optimal encounter sequence and epochs is dealt with at an outer level, whereas the optimization of each single target-to-target transfer is tackled at an inner level. The two layers are interconnected and should be solved simultaneously because they depend on each other.

In order to break this relation, we introduced a strategy that allows for the cost estimate of each transfer. Thanks to this heuristic and to the introduction of discretized encounter epochs we were able to pre-compute all the transfer costs, with no knowledge of the encounter sequence and epochs. This means that it is possible to evaluate the cost of the transfer from any debris to any other one, for any starting and arriving epochs. The problem thus reduces to find the optimal encounter sequence and epochs for all the chasers in order to minimize the overall ΔV. This is a very common routing problem, as discussed in the Introduction.

In the next sections, details on the heuristic used for the single-leg cost estimation are presented, and the formulation of the decoupled outer-layer level problem is given, showing that it can be posed as a (pure) combinatorial problem.

A. SINGLE LEG COST ESTIMATION

The inner layer characterizing the ADR mission planning problem concerns the optimization of each body-to-body single transfer. However, in order to split and ease the problem at hand, we used a simple analytical heuristic that allows for a fast estimate of the transfer cost.

The considered heuristic was proposed for the first time by Shen et al. who used it to estimate the cost of the transfers for the problem proposed in the 9-th edition of the Global Trajectory Optimization Competition (GTOC). It is based on the analytical ΔV estimate by Edelbaum, taking into account the J_2 perturbation which causes a precession of the ascending node. This effect is exploited to reduce the transfer cost, as it eventually brings the orbital planes of the debris closer.

As reported in section IV-A, objects moving on nearly Sun-Synchronous Orbits (SSOs) are considered, which generally lay on different orbital planes. The interest in SSOs is related to the fact that this region of space is one of the most densely populated of both operational and decommissioned space vehicles and debris, thus making the study of how to reduce the chance of a collision with fragments of debris particularly relevant. Transfers between two nearly Sun-synchronous orbits generally benefit from a strategy that allows for achieving the necessary orbit plane rotation by means of the J_2 perturbation, provided a reduction of the fuel required to perform the transfer is achieved, at the expense of a (possibly long) waiting time necessary for the required orbit plane rotation. Details of the heuristic adopted in this study are reported in appendix .

In order to speed up the optimization process, predetermined encounter epochs were assumed, which means that starting and arrival times of the transfer can occur only at prescribed discrete epochs, equally spaced in time. The transfer costs were computed by means of the heuristic mentioned above, and saved in a 4D cost tensor of dimensions $N \times N \times N_T \times N_T$, being N the total number of targets and N_T the number of sampled starting/arrival epochs. Transfer with a small duration (in the order of 30 days or faster) can be neglected, because their cost is in general too high. In such a case, the transfer cost is put to an arbitrarily high value, in order to penalize the corresponding solution. Also, transfer cost monotonically decreases as a function of transfer duration. In this case, it is possible to drop transfers with a very long duration (in the order of 200 days or longer), putting their cost equal to the last calculated transfer. The cost associated to long transfers is thus penalized, in order to automatically prune such solutions during the optimization process.

B. COMBINATORIAL PROBLEM

In the previous section, we presented a strategy for the fast evaluation of transfer costs, in order to ease the original problem and speed up the whole optimization process. The present work is focused on the solution of the outer-layer problem, which consists in finding how to split the targets among the chasers (if more chasers are considered) and the order in which targets should be visited, in order to minimize the overall mission cost. Details of the transfer were neglected, and the parameters $\boldsymbol{\Theta}^{(i)}$ are no longer required for the preliminary mission design.

As discussed above, we considered encounter epochs t_k discretized over a uniform time-grid

$$0 = \tau_0 < \tau_1 < \ldots < \tau_k < \ldots < \tau_{N_T} = T_{max}, \tag{7}$$

where N_T is the number of the sampled encounter epochs, and τ_0 is the departure epoch. The transfer time is thus an integer multiple of the time unit $\Delta T = T_{max}/N_T$. Thus: $\tau_h = h\Delta T, \ h = 0, \ldots, N_T$.

Under these assumptions, we were able to cast the mixed integer optimization problem described by system (5) as a pure combinatorial optimization problem. To this end, we introduced a novel permutation-based encoding, which may handle multiple spacecraft trajectories. Let $\mathbf{\Pi} = \{\Pi_1, \Pi_2, \ldots, \Pi_{N_e}\} \in \mathcal{P}^{N_e}$ be an augmented-size permutation of $N_e = N_c \times N_T$ elements.

The augmented-size permutation can be decoded so as to obtain the encounter sequence of each chaser and the corresponding encounter epochs. To obtain this information, the permutation is first divided into N_c 'stripes' of equal length, each one of N_T elements. Each stripe is then inspected independently: elements greater than N_s are considered as blanks, revealing the sequence of debris to encounter $\boldsymbol{p}^{(i)}$, whereas the position of the non-blank elements reveals the corresponding encounter epochs $\boldsymbol{t}^{(i)}$. More formally, for the i-th chaser one may define $\boldsymbol{p}^{(i)} = \{\Pi_h \,|\, \Pi_h \leq N_s,\ \forall h \in [(i-1)N_T + 1, iN_T]\}$ and $\boldsymbol{t}^{(i)} = \{\tau_h \,|\, \Pi_h \leq N_s,\ \forall h \in [(i-1)N_T + 1, iN_T]\}$.

Figure 1 presents an example of the proposed encoding for a multi-chaser mission with $N_c = 2$ chasers, $N_s = 8$ targets, and a time grid with $N_T = 8$ divisions (or encounter opportunities). Decoding the augmented-size permutation $\mathbf{\Pi}$ reveals that the first chaser visits the targets $\boldsymbol{p}^{(1)} = \{3, 4, 7, 5\}$ at the epochs $\boldsymbol{t}^{(1)} = \{\tau_1, \tau_2, \tau_6, \tau_7\}$, whereas the second chaser visit the targets $\boldsymbol{p}^{(2)} = \{2, 8, 6, 1\}$ at epochs $\boldsymbol{t}^{(2)} = \{\tau_1, \tau_5, \tau_7, \tau_{10}\}$. Notice that the time grid is the same for both chasers.

The resulting combinatorial optimization problem is then formulated as follows:

$$\mathcal{C}: \begin{cases} \min\limits_{\mathbf{\Pi} \in \mathcal{P}^{N_e}} \sum\limits_{i=1}^{N_c} \sum\limits_{k=1}^{N_s^{(i)}-1} \Delta V(p_k^{(i)}, p_{k+1}^{(i)}, t_k^{(i)}, t_{k+1}^{(i)}) \\ \text{s.t. :} \\ \boldsymbol{p}^{(i)} = \{\Pi_h \,|\, \Pi_h \leq N_s,\ \forall h \in [(i-1)N_T + 1, iN_T]\}, \\ \qquad \forall i = 1, \ldots, N_c \\ \boldsymbol{t}^{(i)} = \{\tau_h \,|\, \Pi_h \leq N_s,\ \forall h \in [(i-1)N_T + 1, iN_T]\}, \\ \qquad \forall i = 1, \ldots, N_c \end{cases} \tag{8}$$

C. GENETIC ALGORITHMS

Genetic Algorithms (GAs) are population-based Evolutionary Algorithms (EAs), inspired by the principles of biological evolution. Each individual of the population represents a potential solution to the optimization problem, that, for the adopted encoding, is a permutation of N_e positive integer numbers.

In a basic version of a GA, an initial population of N_P individuals is randomly generated, in order to explore as exhaustively as possible the whole search space. At each generation, individuals are selected for reproduction using a tournament rule, where 'better' individuals (the ones associated with a lower overall mission ΔV, in the present case) are favored. Yet, a small probability of selecting one of the low-performing elements of the population exists, for the sake of maintaining a suitable level of diversity among the individuals. The selected individuals go through a crossover process, with the aim of creating new, and hopefully better, individuals. An elitism mechanism is often adopted, and the best $N_P^{elite} < N_P$ individuals in the populations are directly promoted to the new generation, thus avoiding losing the best solutions found so far, due to some random effects.

When considering a permutation-based encoding, specific permutation-saving crossover operators must be used, to enhance the capabilities of GA's. Among the many proposed in the literature, we chose to implement non-wrapping order crossover (NWOX), partially-matched crossover (PMX), cycle crossover (CX), and uniform PMX (uPMX), as they proved to be the most promising approaches for our class of problems. As an example, the steps taken by the NWOX operator to generate offspring (C1 and C2) starting from parents (P1 and P2) are shown in Fig. 2 and described in some detail. For more details and information about other crossover operators one can refer to [15].

The NWOX operator begins by creating the two children as copies of the parents. Two cut points are randomly selected along the children's permutations (P1 and P2). Suppose, for simplicity, that for both children the two cut points are one between the third and the fourth elements and the other one between the sixth and the seventh elements (as in Fig. 2). Child one (C1) is searched for the elements of C2 between the two cut points, and they are replaced with holes (Fig. 2, step b). Subsequently, a sliding motion is used to move all holes into the region between the cut points. In such a way all non-empty elements of the child are slid leftward or rightwards until they are grouped together in two full strings, which maintain the original order of the genes (Fig. 2, step c). Finally, the operator fills the sub-strings between the cut points of C1 with the genes previously removed, keeping the order they had in P2, and vice versa (Fig. 2, step d).

After the new individuals have been generated, the population goes through a mutation process. Also in this case the mutation operator must preserve the permutation so our implementation makes use of a mutation operator chosen among the following ones: insert, which randomly puts an element in a random position of the permutation; swap, which randomly swaps two elements of the permutation; reverse, which overturns the sub-string of the permutation between two random indices; scramble, which randomizes the positions of the elements in a sub-string of the permutation between two random indices. Eventually, we considered also crossover and mutation operators that we called 'random'. In this strategy, one of the operators is uniformly randomly chosen between the ones presented above. This appears as a good strategy, as it affects the diversity in the population while allowing for a more balanced exploitation-exploration ratio, as we will see in the next section.

Another way of controlling population diversity and avoiding premature convergence towards a local optimum is the activation of an epidemic mechanism. If no progress is done within a fixed number of generations (n_G^{epid}) then a large

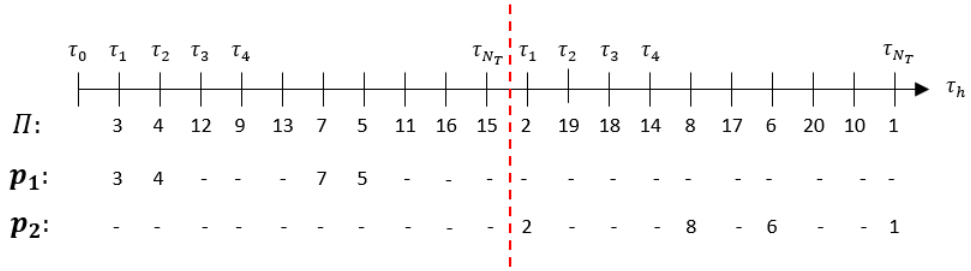


FIGURE 1. Encoding adopted in multiple-chasers missions.

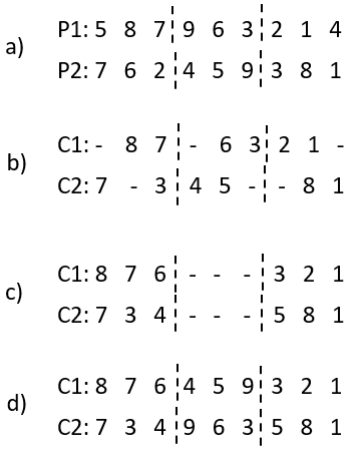


FIGURE 2. Non-wrapping order crossover.

part (ρ_{epid}) of the population is randomly reinitialized over the entire search space. The maximum number of epidemic events that may occur in a single run (n_{max}^{epid}) is fixed, in order not to compromise the overall efficiency of the search.

A Local Search (LS) procedure could also be used to occasionally improve the solution. When LS is active, at the end of generation $N_{gen_0}^{LS}$, and every N_{gen}^{LS} generations after that, a small fraction of the population, composed by n_p^{LS} individuals, undergoes a greedy search process, where the adjective "greedy" implies that in such a case the best solutions are either always chosen, or selected with a very high probability. The neighborhood of the starting individuals is thus searched with the aim of attaining a solution with a better fitness.

In this work, a complete, randomized, 2-Opt local search [16] is performed, using as a starting guess the solution corresponding to each of the selected individuals. This procedure considers all possible combinations obtained by reversing the block of the augmented-size permutation between two indices, and always accepts a candidate permutation resulting into individuals with better fitness than the starting ones. The procedure is repeated until no improvement is obtained for any of the possible combinations. While being very effective, the LS procedures significantly increase the computational cost. Hence a fine-tuning of the LS frequency N_{gen}^{LS} is important from a practical standpoint.

Adding constraints in an optimization process dealt with an EA is not trivial, as they were initially developed for unconstrained optimization problems. *Ad hoc* strategies are thus needed to overcome this issue. Penalty functions are the most common way of incorporating constraints into EAs [17]. In our case, constraints are handled using an augmented value of the objective function depending on constraint violation (*penalty* rule), or favoring *feasible* individuals over the unfeasible ones (*feasibility* rule). As a variant of the feasibility rule, a ε-constraint handling rule is used [18], where individuals are considered to be feasible if the overall constraint violation is lower than a prescribed ε level, which decreases exponentially over time from a starting value ε_0 to a limit value ε_∞, that is

$$\varepsilon = \begin{cases} \varepsilon_0 & \text{if} \quad n_G \leq n_{G,0} \\ \varepsilon_0 \left[\dfrac{\varepsilon_\infty}{\varepsilon_0}\right]^{\frac{n_G - n_{G,0}}{n_{G,\infty} - n_{G,0}}} & \text{if} \quad n_{G,0} < n_G < n_{G,\infty} \\ \varepsilon_\infty & \text{if} \quad n_G \geq n_{G,\infty} \end{cases} \tag{9}$$

where $n_{G,0}$ and $n_{G.\infty}$ define the velocity of the ε decrease.

Finally, a distributed genetic algorithm was implemented. Distributed topology exploits an archipelago-like structure: the population is dived into smaller sub-populations, also known as *demes*. The island of the archipelago are isolated from each other and each evolves independently and simultaneously. They interact only with every fixed number of iterations, with an exchange of selected agents from one island to another. The characteristics of the migration (i.e., the sending and accepting islands) are defined by the prescribed migration policy. Consequently, in distributed topology, exploration is promoted while the islands are evolving independently, whereas exploitation is promoted during the migrations of agents between the demes. The interaction between evolution and migration is able to maintain a favorable diversity among the populations, which helps delaying premature convergence and granting a good balance between exploration and exploitation.

According to the migration policy, every N_{G^m} fixed iterations, N_{P^m} best individuals of one island migrate and replace the N_{P^m} worst individuals on another island. Each island sends and accepts agents only once. The scheme of the migration is pointed out by the interconnections between the islands. The islands are organized in a 2-dimensional

grid, putting the ones with the same crossover operator on the same line and the ones with the same mutation operator on the same column. On one row, from left to right, one has respectively random, NWOX, PMX and CX crossover operators, eventually repeated. Through a column (from up to down) one has random, Revers, Insert, and Swap mutation operators, eventually repeated. With this structure in mind, four different interconnections were considered, from which as many migration policies derive:

- Ring (column) migration, in which agents go from an island to the one right below, except for the last one that sends agents to the first island;
- Ring (row) migration, where agents travel rightwards to the closest island and from the last island to the first;
- In Full migration agents are put together, scrambled, and then distributed to the islands in a random way;
- Random migration, where starting and arrival islands are chosen randomly, paying attention to selecting each island only once.

One can notice that in the last two migration scheme, the structure of the archipelago and the position of the islands is not important, as they are based on random factors, whereas they could play a role in the first two schemes. However, as we will see at the end of the results section, the random scheme proved to be the best one by far, so the role of the structure of the archipelago and of the position of the islands was not investigated.

The termination criteria are mainly based on the generation number N_G, that is, on the computational budget allocated for one run of the optimization algorithm. The optimal value for this parameter strongly depends on the actual problem under investigation. In our case, the number of fitness evaluations (FES) is fixed and the number of generations is evaluated as $N_G = FES/N_P$. This provides a direct link between the termination criterion with the available computational budget.

As a final consideration, the performance of a genetic algorithm clearly depends on the choice of the selection, mutation, and crossover operators, but also on several 'hyper-parameters', such as population size (N_P), number of generations (N_G), 'elite' population size (N_P^{elite}), percentage of population 'killed' by the epidemic (ρ^{epid}), the maximum number of epidemics in a single run (n_{max}^{epid}), migration agents (N_P^m), migration frequency (N_G^m), crossover probability (p_c), that represents the percentage of the parents replaced by offspring, mutation rate (p_m), that is the probability that one individual will undergo a random mutation, and other operator-specific parameters. Proper tuning of these hyper-parameters, usually performed on a simplified (and possibly down-scaled) problem, is required for optimizing the solution capability of the GA.

Algorithm 1 Pseudo-Code of the Island-Structured Genetic Algorithm

```
Initialize first population p of N_P elements
while termination conditions are not met do
    for each island do
        Evaluate fitness
        Save N_P^elite elite individuals
        Perform selection
        Perform crossover
        Perform mutation
        if Local search is active & local search conditions
are met then
            Perform local search on N_P^LS individuals
        end if
        Replace old population (p ← p_new)
        Insert elite individuals in p
        if migration conditions are met then
            Perform migration
        end if
        if Epidemic conditions are met then
            Activate epidemic mechanism
            Replace 'dead' individuals
        end if
    end for
    Increase generation number by 1
end while
```

IV. RESULTS

In this section the test case considered is introduced first, then the results of simulations carried out on this test case are reported and discussed. The main goal of the tests is to analyze the performances of the proposed algorithm, in order to tune the hyperparameters and, if possible, find the operators that work suitably well for the whole class of problems at hand, for any mission that one may be faced with.

A. TEST CASE

The proposed solution method was tested on a test case widely used in the literature, which consists of a cloud of 21 fictitious debris fragments [14]. Each of them moves on a circular LEO with an inclination between 97 deg and 99 deg, to simulate a nearly SSO. It is known that in such a region of near-Earth space the concentration of debris is very high, because it is one of the most populated areas for its practical use in Earth observation missions.

The orbital parameters of the 21 fragments of the fictitious debris cloud are reported in Table. 1, which shows the altitude, inclination, initial RAAN, and precession rate of each target. Since we considered Keplerian dynamics, except for the J_2 effect over the ascending node, these parameters are sufficient to fully describe their motion.

The orbits have different orbital parameters, hence they are characterized by different node precession rates. This in turn causes the orbital planes to achieve a minimum angular

TABLE 1. Orbital elements of the targets.

Debris Number	Altitude (km)	Inclination (deg)	Initial Ω (deg)	Precession rate (deg/day)
Debris 1	700	97.0	0	0.8429
Debris 2	710	97.3	90	0.8745
Debris 3	720	97.6	180	0.9058
Debris 4	730	97.9	270	0.9367
Debris 5	740	98.2	018	0.9672
Debris 6	750	98.5	108	0.9975
Debris 7	760	98.8	198	1.0273
Debris 8	770	97.1	288	0.8260
Debris 9	780	97.4	36	0.8565
Debris 10	790	97.7	126	0.8866
Debris 11	800	98.0	216	0.9165
Debris 12	810	98.3	306	0.9460
Debris 13	820	98.6	54	0.9752
Debris 14	830	98.9	144	1.0040
Debris 15	840	97.2	234	0.8094
Debris 16	850	97.5	324	0.8389
Debris 17	860	97.8	72	0.8681
Debris 18	870	98.1	162	0.8969
Debris 19	880	98.4	252	0.9254
Debris 20	890	98.7	342	0.9536
Debris 21	900	99.0	360	0.9815

distance, after a sufficient waiting time, which results in a cheaper transfer cost. These considerations motivate the choice of a heuristic for the transfer cost estimation which exploits the J_2 perturbation effect.

B. ALGORITHM PRELIMINARY TUNING

Missions with up to 4 chasers acting simultaneously and time-grids with different time-steps were considered, namely $\Delta T = 10,\ 20,\ 30$, and 60 days. The number of possible encounter epochs also depends on mission duration, T_{max}. For $T_{max} = 720$ days, one obtains $N_T = 72, 36, 24$, or 12, for the ΔT considered. Hereafter, the notation $N_c@N_s \times N_T$ is used for a mission that employs N_c chasers to remove N_s targets with a time-grid of N_T encounter epochs.

The computational time varies significantly with the mission under investigation. As expected, problem complexity, and consequently computational time required to obtain a good solution, increases as the number of chasers and/or encounter epochs increases. As an example, solving the 14-targets mission with 3-chasers and $N_T = 12$ requires about 10 minutes on a computer with an Intel Core™ i7-1065G7 CPU @ 3.90 GHz. When the same problem is solved for $N_T = 72$, time increases up to 150 minutes, that is, increasing the number of considered encounter epochs by a factor of 6 causes an increase in computation cost by a factor of 15. Parallel computation can be effectively used to keep the run-time sufficiently low, exploiting the ease of parallelization intrinsic to GA algorithms.

A preliminary tuning was carried out in order to identify an acceptable configuration of hyperparameters and operators of the GA. During this phase, it was evident that each run asymptotically reached a (possibly suboptimal) population distribution. Hence, rather than increasing the number of generations up to very large numbers, it is more convenient to perform multiple independent runs of the same mission, increasing the chances that the optimal solution is found by one or more of the runs, thanks to the random initialization of the starting population. This aspect will be further discussed below.

TABLE 2. Results for the crossover operator analysis, mission 3@15 × 24, random mutation operator.

	Crossover Operator				
N_P	Random	NWOX	PMX	uPMX	CX
64	2.2154	2.3289	2.3015	2.5295	2.4311
128	2.1464	2.2448	2.2396	2.3496	2.4722
256	**2.1046**	2.1678	2.220	**2.2689**	2.3902
512	2.2066	**2.1256**	2.2236	2.5892	2.3434
1024	2.2483	2.2094	**2.1855**	2.9266	**2.2442**

TABLE 3. Results for the mutation operator analysis, mission 3@15 × 24, random crossover operator.

	Mutation Operator				
N_P	Random	Reverse	Insert	Swap	Scramble
64	2.2154	2.3302	2.6665	2.4961	2.6594
128	2.1464	2.2153	2.4050	2.2681	2.3691
256	**2.1046**	**2.1353**	**2.270**	**2.1817**	**2.2835**
512	2.2066	2.1841	2.2874	2.2198	2.3066
1024	2.2483	2.1989	2.3122	2.2906	2.3888

Performances of the algorithm were analyzed as a function of the crossover and mutation operators, together with population size. After fixing the mutation operator, the same mission was solved for different population sizes and crossover operators. The same strategy was used to assess the effectiveness of the mutation operators, prescribing the crossover operator. The first tests were conducted on a mission involving 3 chasers to clean 15 targets (opportunely chosen to match results reported by Cerf [14]) with 24 potential encounter epochs with a time step of 1 month, so the overall mission duration is 720 days.

The results of the preliminary study on hyperparameters are reported in Tables 2 and 3. Each row of the tables reports the mean fitness over a hundred tests with a fixed population size, CR, and MT operators. In the analysis of the crossover operator, the random mutation operator was considered, whereas when the mutation operator is analyzed, the random crossover operator was selected. In each column the best value is highlighted in bold, in order to point out the best configuration, for a prescribed total number of FES. The best results are always obtained for population sizes $N_P = 256$ or 512; smaller populations result in a premature convergence of the algorithm to suboptimal solutions, whereas bigger ones do not converge fast enough and require more FES (hence, a heavier computational burden) to obtain better results.

As far as the crossover and mutation operators are concerned, the random operators outperform all the other ones for smaller population sizes, up to $N_P = 256$. For larger population sizes, the NWOX and PMX crossover operators and the reverse mutation operator provided better solutions.

TABLE 4. Results for the crossover operator analysis, mission 4@21 × 36, random mutation operator.

	Crossover Operator				
N_P	Random	NWOX	PMX	uPMX	CX
64	**3.4334**	3.5755	3.5653	3.5396	3.5996
128	3.5202	3.4391	3.5534	3.6075	3.5431
256	3.4401	**3.3397**	3.4909	**3.4225**	3.5561
512	3.6714	3.4819	3.3438	4.9926	**3.4074**
1024	3.9412	3.9568	**3.3430**	6.6656	3.4456

TABLE 5. Results for the mutation operator analysis, mission 4@21 × 36, random crossover operator.

	Mutation Operator				
N_P	Random	Reverse	Insert	Swap	Scramble
64	**3.4334**	3.7510	4.2122	3.5290	4.1963
128	3.5202	3.4887	3.7012	3.5258	3.6752
256	3.4401	**3.4881**	**3.5759**	**3.4513**	**3.5955**
512	3.6714	3.5616	3.7248	3.6221	3.7926
1024	3.9412	3.7997	3.9131	3.8047	3.9327

The best configuration for crossover/mutation operators and population size is thus obtained for random/random with $N_P = 256$. Other configurations that performed well are NWOX/random with either $N_P = 512$ or $N_P = 256$, and PMX/random with $N_P = 512$.

The same analysis is then performed for a different, and more complex, mission scenario. Tables 4 and 5 show the results obtained for the mission 4@21 × 36, with a maximum duration of 720 days. Here the best results are obtained for the NWOX/random configuration, with $N_P = 256$. Other well-performing configurations are the PMX/random. Thus, a robust best configuration cannot be identified, as some configurations perform well on a problem, whereas another one provides better solutions on a different problem. If an absolute best configuration for the algorithm is not available, however, some bad configurations, which perform badly in both problems and that should be avoided, are definitely highlighted. As an example, the uPMX crossover operator and the insert mutation operator perform more or less poorly in every configuration.

As a second step, the combined effect of maximum number of generations and population size on algorithm performance is investigate for our version of a classical GA. The comparison is performed in terms of mean fitness and best solution found. Random operators were considered for both mutation and crossover, with a reference population size of 512 individuals. The mission 3@15 × 24 was chosen for the analysis, which is a sufficiently simple problem solved in a relatively small amount of CPU time, but still representative of the group of problems that the proposed solution method is required to tackle. Also, in this case a hundred of runs are carried out for each test. The results of this analysis are presented in Fig. 3 and Fig. 4, which reports respectively the best solution found among all the runs and the mean of the best solutions found in each run, as a function of the number of FES, for different population sizes. In both plots, the x-axis

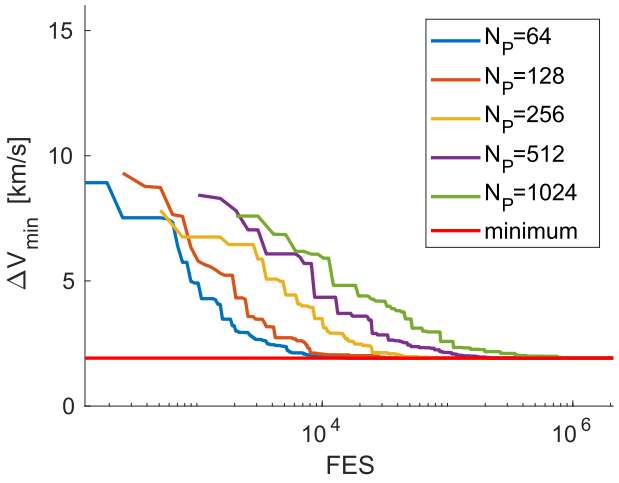


FIGURE 3. Best solution found versus number of FES for different population sizes.

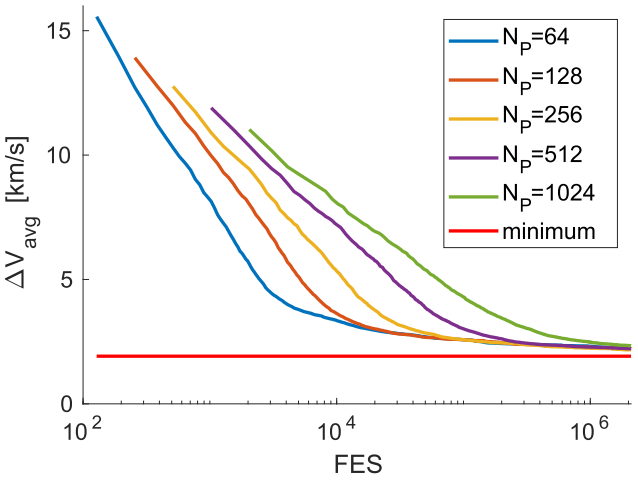


FIGURE 4. Mean of the best solution found in each run versus number of FES for different population sizes.

is represented in a logarithmic scale and the red line points out the optimal solution found for the considered mission.

In Fig. 3, one clearly sees that the minimum of the fitness function is always reached after a certain number of FES, for every size of the population. Quite obviously, the plot starts at a higher value of FES, as the size of the population increase, because the minimum number of FES is equal to the size of the population itself. However, if one has to choose the size of the population just from this plot, then the choice would clearly be $N_P = 64$, because it reaches the global minimum with a smaller number of FES, which implies a smaller computational cost. If we look at Fig. 4 this behaviour is confirmed also by the mean of the fitness over all the tests. In fact, it is possible to see that for smaller population sizes a faster convergence is achieved, represented by the slope of the plots, which is higher for faster convergence rate. However, the value to which the algorithm converges is not the same, and it is lower for a population size of 256 individuals, which

TABLE 6. GA hyper-parameters.

Parameter	Symbol	Value
Selection rule		Tournament ($k = 2$)
Crossover rule		NWOX
Mutation rule		Random
Population size	n_P	256
No. of generations	n_G	25000
Crossover probability	p_c	0.9
Mutation probability	p_m	0.1
Elite size	n_P^{elite}	12
No. of epidemics	N_{max}^{epid}	10
Epidemics frequency	N_G^{epid}	200
Epidemic mortality	ρ^{epid}	100%
LS population size	n_P^{LS}	50
LS starting generation	$N_{gen_0}^{LS}$	500
LS frequency	N_{gen}^{LS}	500

thus seems to grant the best compromise between exploitation and exploration of the search space.

Looking at the result found so far, we decided to use the NWOX crossover operator and the random mutation operator, with a population size of 256 individuals. This configuration performs well in both considered missions, being better in the more complex one. The choice was dictated considering that for simple missions even a suboptimal configuration can achieve good results (i.e., it usually finds the best solution over 100 runs), whereas for complex test cases only a good configuration finds the globally optimal solution. Table 6 reports the configuration used in the successive analysis which we now present.

C. MISSION DURATION AND TIME DISCRETIZAION

The effect of total mission duration (T_{max}) and time-grid discretization (ΔT) on overall mission cost was then investigated using the 10-targets missions as a test case. Missions involving either 2 or 3 chasers were considered, with a duration between 300 and 720 days, for four possible time-grid discretization, that is, $\Delta T = 10$, 20, 30 or 60 days. Figure 5 presents the results of this parametric analysis. As expected, mission cost decreases as the duration of the mission increases. Mission cost is mainly due to the ΔV required for changing the RAANs. A longer mission time allows for a better use of nodal precession, which causes chaser orbital plane to approach that of the selected target at no fuel cost.

For what concern the effect of time discretization on solution quality, it is evident that a finer time-grid allows for improved results in terms of total ΔV, whatever the mission duration. However, differences between the finest and the coarser time-grids become smaller as the duration of the mission increases. This result suggests that the use of denser time grids is often unnecessary, especially in the case of missions of duration exceeding 18 months. In such a case, only minor ΔV improvements are obtained, hardly relevant in practice, at the cost of a significant increase in computational cost, due to the increase in the size of the combinatorial problem. A time unit ΔT of approximately 20 days is sufficient and it is used in the following investigations.

D. ΔV BALANCING

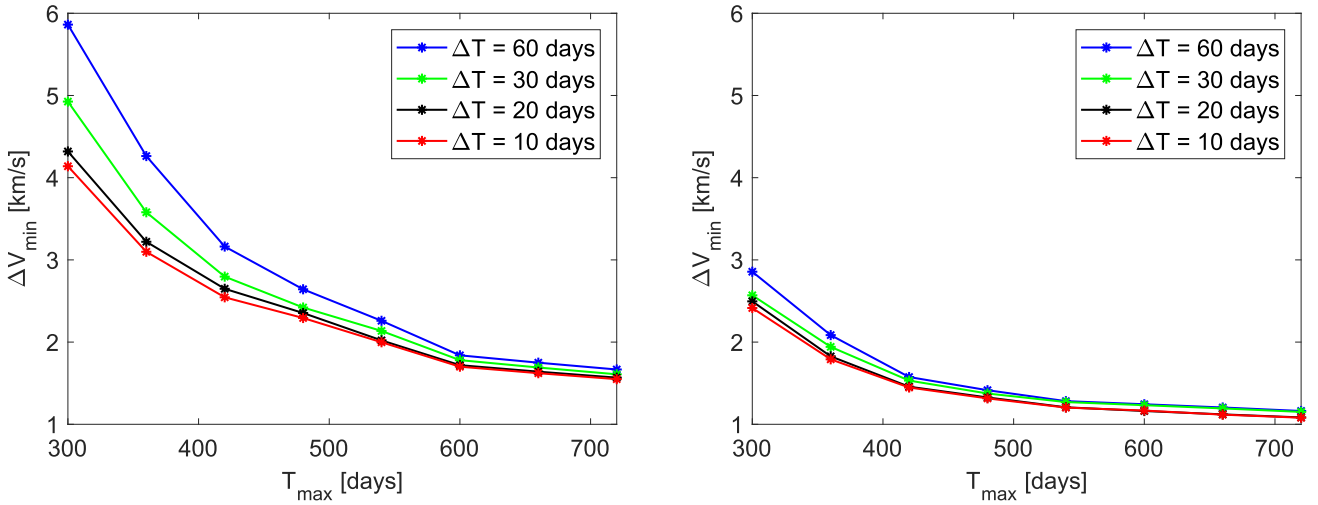
The distribution of mission cost among the chasers is the subject of a specific analysis. Figure 6 presents a bar plot of the ΔV spent by each chaser for the 2@10 and 3@10 missions, as a function of overall mission duration, T_{max}. This figure suggests that in the presence of two chasers the allocation of the propulsive effort among active spacecraft is quite balanced (Fig. 6a). Conversely, when three or more chasers are used, the solution may present relevant variations of mission cost share associated to each spacecraft (Fig. 6b). As an example, for $T_{max} = 660$ days, chaser 3 collects only two debris, with a low ΔV (only 170.06 m/s), whereas chaser 1 collects 4 fragments, with ΔV=409.20 m/s.

The mission with 10 targets, $T_{max} = 720$ days, and $\Delta T = 20$ days was further investigated, for better understanding how the encounter sequence is modified by a different number of chasers. Figure 7 presents the sequences of targets reached by each chaser in a radius versus inclination plot, for missions with 2 and 3 chasers, respectively. Encircled targets (e.g., debris 16 and 11 in Fig. 7a) indicate the first target visited by each chaser, whereas the arrows point out the rendezvous sequence. In the considered scenarios, the blue sequence for chaser 1 differs only for the starting debris, which is n. 16 in the two-chaser mission, but it is visited by chaser 3 in the 3@10 mission (Fig. 7b). The sequence for chaser 2 (represented in orange in Fig. 7a), is split into two sequences in Fig. 7b, when a third chaser is introduced. The sub-sequence 15-3-14 (also in orange in Fig. 7b) is maintained but traversed in the opposite direction.

If the maximum ΔV available to each chaser is limited to $\Delta V_C = 500$ m/s, the encounter sequence changes. Figure 8, shows the removal debris maps for the constrained missions, with $T_{max} = 720$ days and $\Delta T = 20$ days. Chaser 1 collects the same debris as in the unconstrained case, whereas the other two sequences are changed. Target 11, previously visited by the third chaser, is now visited by the chaser 2. The relocation of target 11 allows for a reduction in the propulsive effort of chaser 3, so that the ΔV of each chaser is below the enforced threshold. The overall mission cost increases (+6%), as chaser 2 is tasked with removing this debris, deviating from its optimal mission path. Table 7 presents the ΔVs of constrained and unconstrained solutions.

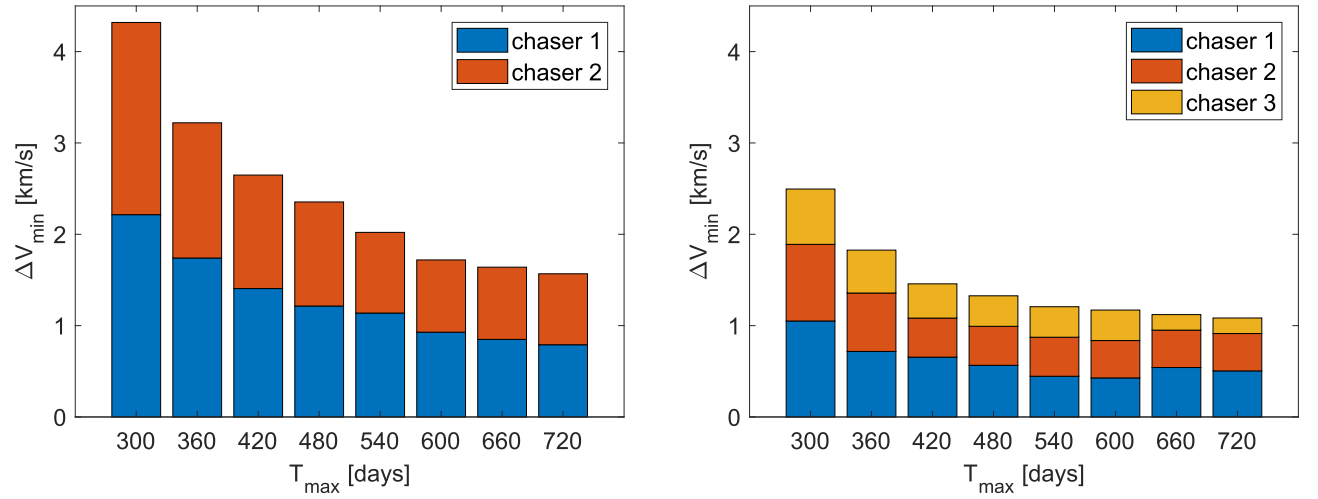
E. SIMULATIONS AND DISCUSSION

The capability of the proposed approach to handle non-overlapping time windows was investigated using as a first case study the 15-targets mission with 3 chasers. The overall mission duration was set as $T_{max} = 1360$ days with grid time-step $\Delta T = 20$ days. The best solution is reported in Table 8. Note that, in this case, the results are in full agreement with those reported in [14], confirming the validity of the solution method proposed in the present paper.



(a) Mission 2@10. (b) Mission 3@10.

FIGURE 5. Mission cost as a function of flight time, for several time-grid discretizations.



(a) 2@10 mission. (b) 3@10 mission.

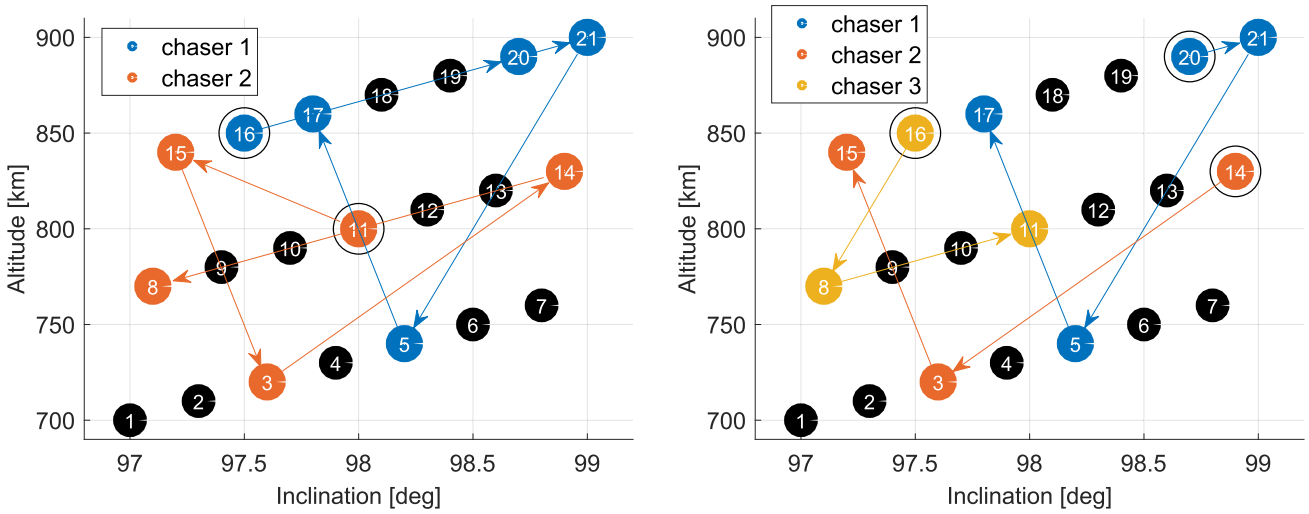
FIGURE 6. Bar plot of the chasers' fuel consumption as a function of the mission duration T_{max} ($\Delta T = 20$ days).

TABLE 7. Comparison between unconstrained and constrained 3@10 missions ($T_{max} = 720$ days, $\Delta T = 20$ days).

Chaser	ΔV [m/s]	
	Unconstrained	Constrained
#1	409.20	409.20
#2	170.06	406.36
#3	504.30	333.92
Tot	1083.56	1149.48

The complete removal of all 21 debris using 4 chasers was finally considered, as the most complex mission scenario available, for various mission duration, using a standard GA. Figure 9a shows the bar plot of the ΔV spent by each chaser as a function of mission duration. In the absence of a constraint that enforces a uniform distribution of the ΔV among the chasers, the optimal solution presents relevant differences in the fuel spent by different chasers for all mission times, with one chaser always removing fewer debris than the others. Figure 9b presents the debris removal map of the best solution found for $T_{max} = 720$ day.

The same problem is also tackled using the novel Island-GA. The hyperparameters of the GA implemented for evolving the population of the different islands are selected using all the possible combinations of crossover and mutation operators, leaving out only the uPMX and insert operators, which proved to be less efficient during the preliminary tuning test phase. Consequently, we have 4 crossover and 4 mutation operators, with 16 possible combinations. The algorithm thus



(a) Mission 2@10. (b) Mission 3@10.

FIGURE 7. Debris map of the visited targets, in a radius vs. inclination graph (T_{max} = 720 days, ΔT = 20 days).

TABLE 8. 15-targets mission with chasers that operate in non-overlapping time windows.

Chaser #1			Chaser #2			Chaser #3		
ID	t [days]	ΔV[m/s]	ID	t [days]	ΔV[m/s]	ID	t [days]	ΔV[m/s]
16	0	-	15	520	-	1	840	-
20	160	338.74	3	560	67.76	4	960	60.97
21	340	235.85	14	700	364.08	9	1120	432.09
5	440	241.49	11	760	210.59	7	1300	91.83
17	500	163.48	8	820	60.63	12	1340	41.68
Tot		979.56	Tot		703.07	Tot		626.58

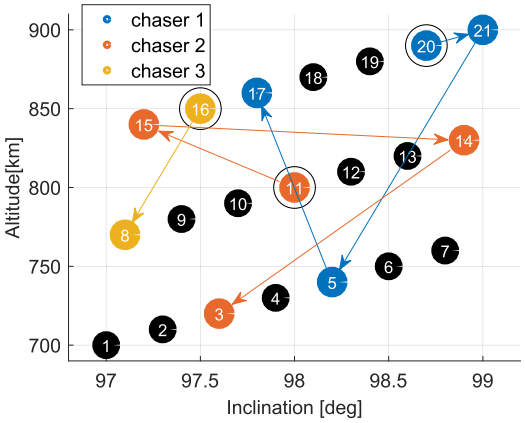


FIGURE 8. Debris removal map for the 3@10 mission with T_{max} = 720 days, ΔT = 20 days, ΔV_C = 500 m/s).

TABLE 9. Results for the migration strategy analysis, mission 3@15 × 24.

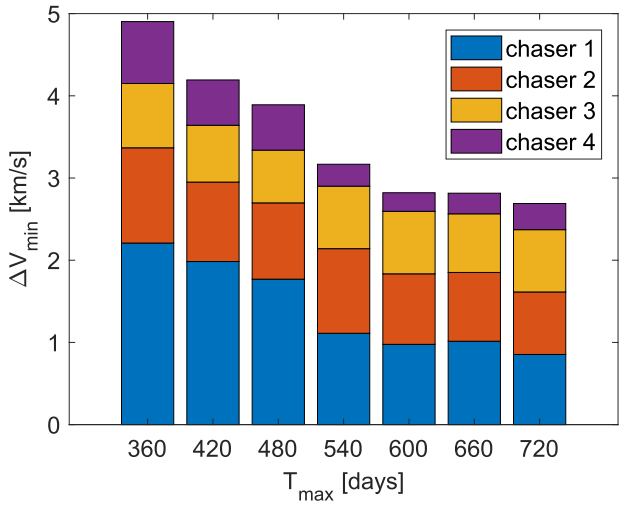
	Migration Strategy			
N_P	Ring (column)	Ring (row)	Full	Random
64	2,0042	2,0049	2,099	1,9694
128	**1,9671**	**1,953**	2,042	**1,9481**
256	2,008	1,9887	**1,9932**	1,9764
512	2,0304	2,01	2,0041	2,013
1024	2,0531	2,0243	2,1178	2,0256

TABLE 10. Results for the migration strategy analysis, mission 4@21 × 36.

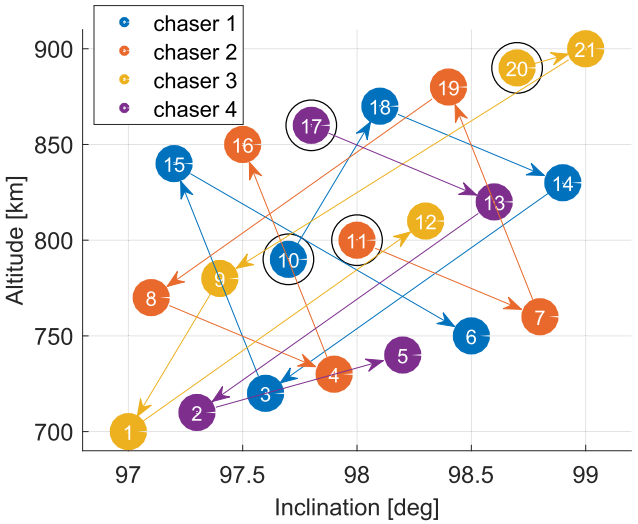
	Migration Strategy			
N_P	Ring (column)	Ring (row)	Full	Random
64	3,1613	3,1204	3,8805	3,0793
128	**3,0598**	**3,0418**	3,2335	**2,9878**
256	3,2384	3,111	3,216	3,149
512	3,2274	3,1623	**3,206**	3,2186
1024	3,3003	3,2623	3,3878	3,2371

exploits a structure of $N_I = 16$ islands, each of them with a population of $N_P^I = N_P/N_I$ individuals.

Tables 9 and 10 summarize the results, where the mean of the best solution found in each of the 100 test is reported for each value of the overall population size and for different migration schemes. The results are promising, because the new algorithm outperforms all configurations of the classical GA tested so far. Moreover, the performances of the Island-GA are apparently less dependent on the particular problem. A configuration is identified, that performs better than the other ones in both missions, employing the



(a) Chasers' fuel consumption.



(b) Debris map ($T_{max} = 720$ days).

FIGURE 9. Results of the mission 4@21.

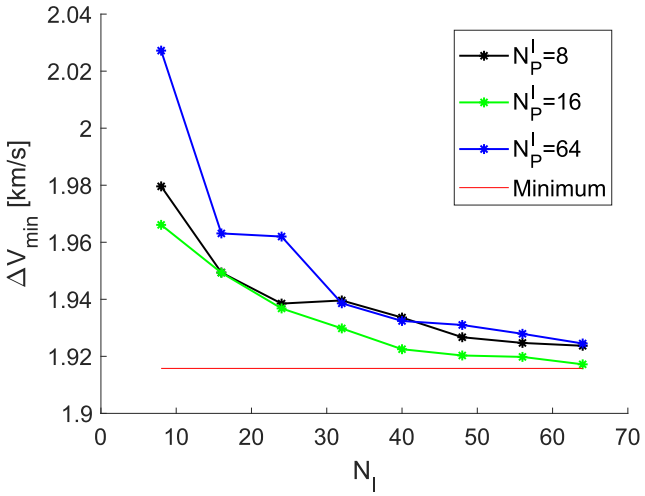


FIGURE 10. Best fitness found in 100 runs versus the number of islands forming the archipelago.

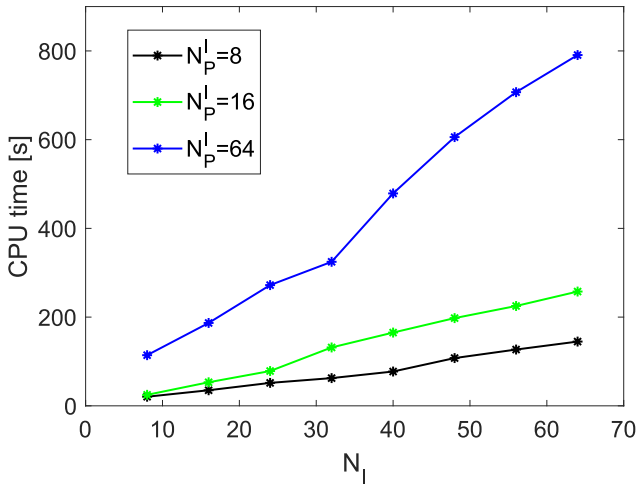


FIGURE 11. Permutation-preserving crossover operators.

random migration scheme with an overall population of 128 individuals.

In order to fully understand how the performances of the Island-GA change according to the number of islands, we report two figures that represent the plot of the mean ΔV versus the number of islands, for a fixed size of the island's population (Fig. 10) and the CPU time required by the algorithm to solve these problems (Fig. 11), in a number of islands vs computational time plot. It is evident that the configuration with 16 individuals per island is the best, performing better than the others, especially as the number of islands increases. A higher value of N_I allows for better performances, in terms of quality of the optimal solution found, at the cost of an increased computational time, as reported in Fig. 11. This means that a compromise is needed between quality of the results and computational time. Configurations that employ 20 or 30 islands with 16 individuals per island represent a good choice because good performances are obtained within an acceptable computational time. Configurations with a higher number of islands improve performances only by a few percent, whereas computational time increases at a higher rate.

V. CONCLUSION

The use of various implementations of classic genetic algorithms (GA) and a novel, archipelago GA is analyzed, as a means for mission planning, in the framework of active debris removal by means of a fleet of chasers working simultaneously over a given cloud of debris. The analysis showed that random crossover and mutation operators perform better than

other ones, for small and medium sizes of the population of solutions.

The results obtained for a conventional implementation of a GA are in line with those available in the literature for the same test case, thus proving the validity of the approach. The analysis proved that it is possible to enforce *ad hoc* constraints in order to solve practical issues, such as a uniform distribution of fuel required by different chasers during a mission. Finally, the novel archipelago structures allow obtaining even better solutions, less dependent on the particular problem solved, at a comparable computation cost.

APPENDIX. SINGLE RENDEZVOUS COST ESTIMATE

The heuristic for estimating the cost of each rendezvous follows the work of Shen and Casalino [13], that expands the well-known Edelbaum approach [19], used to evaluate the approximate ΔV cost for low-thrust multi-revolution orbit transfers between almost-circular orbits with different semi-major axis and inclination in Keplerian dynamics, accounting for the presence of J_2 Earth perturbation. Relevant equations are here reported for the sake of completeness. Please refer to Ref. 13 for further details.

The Precession of the nodal line due to the effect of J_2 Earth's oblateness is evaluated as

$$\frac{d\Omega}{dt} = -\frac{3}{2}\sqrt{\frac{\mu}{a^3}}\frac{J_2 \cos i}{(1-e^2)^2}\left(\frac{r_E}{a}\right)^2. \tag{10}$$

Let us consider a transfer leg between the k-th and $(k+1)$-th targets, which departs at time t_k and arrives at time t_{k+1}. The approximate ΔV costs to match node angle, semi-major axis and inclination, as proposed by Edelbaum [19], are

$$x = \Delta V_\Omega = (\Omega_{k+1}(t_{k+1}) - \Omega_k(t_{k+1}))v_0 \sin i_0, \tag{11}$$

$$y = \Delta V_a = \frac{a_{k+1} - a_k}{2a_0} v_0, \tag{12}$$

$$z = \Delta V_i = (i_{k+1} - i_k)v_0, \tag{13}$$

where $a_0 = (a_{k+1} + a_k)/2$, $i_0 = (i_{k+1} + i_k)/2$, and $v_0 = \sqrt{\mu/a_0}$.

A two-impulse maneuver with combined changes of energy, inclination and RAAN is considered. The coefficients that regulate the split of x, y, and z between the two impulses are

$$S_z = -\frac{my - 2x + nz}{x(m^2+n^2+4)}, \tag{14}$$

$$S_y = \frac{yn^2 - mzn + 4y + 2mx}{2y(m^2+n^2+4)}, \tag{15}$$

$$S_z = \frac{zm^2 - nym + 4z + 2nx}{2z(m^2+n^2+4)}, \tag{16}$$

with the auxiliary quantities m, n, and $\dot{\Omega}_0$ defined as

$$m = -(7\dot{\Omega}_0 \sin i_0)(t_f - t_i), \tag{17}$$

$$n = -(\dot{\Omega}_0 \sin i_0 \tan i_0)(t_f - t_i), \tag{18}$$

$$\dot{\Omega}_0 = (\dot{\Omega}_2 + \dot{\Omega}_1)/2. \tag{19}$$

The first impulse can thus be written as

$$\Delta V_1 = \sqrt{(S_x x)^2 + (S_y y)^2 + (S_z z)^2}. \tag{20}$$

After the first impulse, the chaser's RAAN varies according to a precession rate $\dot{\Omega}_1$ that is related to the effect of J_2. Therefore, the final difference of node angles needs to be adjusted according to the new precession. This results in a decrease of the ΔV needed to match the node angle x

$$\Delta x = mS_y y + nS_z z. \tag{21}$$

The second impulse is

$$\Delta V_2 = \sqrt{(x - S_x x - \Delta x)^2 + (y - S_y y)^2 + (z - S_z z)^2}. \tag{22}$$

The overall cost of the transfer can be approximated with

$$\Delta V = \Delta V_1 + \Delta V_2. \tag{23}$$

In practice, an analytical solution that gives the (sub-)optimal split coefficients S_x, S_y, and S_z is found by taking the first derivative of ΔV^2 with respect to the coefficients themselves and neglecting the cross product term, that is, $\Delta V^2 = (\Delta V_1 + \Delta V_2)^2 \approx \Delta V_1^2 + \Delta V_2^2$. It is worth noting that there is no constraint on the value of the coefficients S_x, S_y, and S_z. This means that, if convenient in terms of transfer cost, the semi-major axis and inclination changes may be larger than the original difference, in order to take advantage of the node precession caused by J_2.

As a second remark, the case where the transfer time is large enough to allow for the difference in RAAN to be recovered without cost must be handled separately. In that case, the chaser waits on the departure orbit for the orbital planes to be 'naturally' aligned and a combined change of energy and inclination takes then place. The transfer cost is then approximated as

$$\Delta V = 0.5\, v_0 \sqrt{\left(\frac{\Delta a}{a_0}\right)^2 + \Delta i^2}. \tag{24}$$

This heuristic allows for a rapid evaluation of the (estimated) rendezvous costs and can be thus used in place of Eq. (1), significantly reducing the overall computational effort.

REFERENCES

[1] D. J. Kessler and B. G. Cour-Palais, "Collision frequency of artificial satellites: The creation of a debris belt," *J. Geophys. Res.*, vol. 83, no. A6, pp. 2637–2646, Jun. 1978, doi: 10.1029/JA083iA06p02637.

[2] J.-C. Liou and N. L. Johnson, "Instability of the present LEO satellite populations," *Adv. Space Res.*, vol. 41, no. 7, pp. 1046–1053, 2008, doi: 10.1016/j.asr.2007.04.081.

[3] B. Pan, P. Lu, and Z. Chen, "Coast arcs in optimal multiburn orbital transfers," *J. Guid., Control, Dyn.*, vol. 35, no. 2, pp. 451–461, Mar. 2012, doi: 10.2514/1.54655.

[4] X. Yue, Y. Yang, and Z. Geng, "Continuous low-thrust time-optimal orbital maneuver," in *Proc. 48th IEEE Conf. Decis. Control (CDC) 28th Chin. Control Conf.*, Dec. 2009, pp. 1457–1462, doi: 10.1109/CDC.2009.5399622.

[5] L. Federici, A. Zavoli, and G. Colasurdo, "A time-dependent TSP formulation for the design of an active debris removal mission using simulated annealing," in *Proc. AAS/AIAA Astrodyn. Specialist Conf.*, Portland, ME, USA, Aug. 2019, pp. 1–20.

[6] H.-X. Shen, T.-J. Zhang, L. Casalino, and D. Pastrone, "Optimization of active debris removal missions with multiple targets," *J. Spacecraft Rockets*, vol. 55, no. 1, pp. 181–189, Jan. 2018, doi: 10.2514/1.A33883.
[7] D. Izzo, I. Getzner, D. Hennes, and L. F. Simões, "Evolving solutions to TSP variants for active space debris removal," in *Proc. Annu. Conf. Genetic Evol. Comput.*, Jul. 2015, pp. 1207–1214, doi: 10.1145/2739480.2754727.
[8] Y. Liu, J. Yang, Y. Wang, Q. Pan, and J. Yuan, "Multi-objective optimal preliminary planning of multi-debris active removal mission in LEO," *Sci. China Inf. Sci.*, vol. 60, no. 7, pp. 1–10, Jul. 2017, doi: 10.1007/s11432-016-0566-7.
[9] J. Murakami and S. Hokamoto, "Approach for optimal multi-rendezvous trajectory design for active debris removal," presented at the 61th Int. Astronaut. Congr., 2010.
[10] S. Hokamoto, "Genetic-algorithm-based rendezvous trajectory design for multiple active debris removal," presented at the 28th Int. Symp. Space Tech. Sci., Jun. 2011.
[11] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm Evol. Comput.*, vol. 39, no. 4, pp. 24–35, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650217308805
[12] H. Mühlenbein, M. Schomisch, and J. Born, "The parallel genetic algorithm as function optimizer," *Parallel Comput.*, vol. 17, nos. 6–7, pp. 619–632, Sep. 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167819105800523
[13] H.-X. Shen and L. Casalino, "Simple ΔV approximation for optimization of debris-to-debris transfers," *J. Spacecraft Rockets*, vol. 58, no. 2, pp. 575–580, Mar. 2021, doi: 10.2514/1.A34831.
[14] M. Cerf, "Multiple space debris collecting mission: Optimal mission planning," *J. Optim. Theory Appl.*, vol. 167, no. 1, pp. 195–218, Oct. 2015, doi: 10.1007/s10957-015-0705-0.
[15] V. A. Cicirello, "Non-wrapping order crossover: An order preserving crossover operator that respects absolute position," in *Proc. 8th Annu. Conf. Genetic Evol. Comput.*, vol. 2, Jul. 2006, pp. 1125–1132.
[16] M. G. A. Verhoeven, E. H. L. Aarts, and P. C. J. Swinkels, "A parallel 2-opt algorithm for the traveling salesman problem," *Future Gener. Comput. Syst.*, vol. 11, no. 2, pp. 175–182, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167739X9400059N
[17] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, pp. 1245–1287, Jan. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782501003231
[18] L. Federici, B. Benedikter, and A. Zavoli, "EOS: A parallel, self-adaptive, multi-population evolutionary algorithm for constrained global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–10.
[19] T. N. Edelbaum, "Propulsion requirements for controllable satellites," *ARS J.*, vol. 31, no. 8, pp. 1079–1089, Aug. 1961, doi: 10.2514/8.5723.

DANILO ZONA received the B.Sc. degree in aerospace engineering from the University of Naples Federico II and the M.Sc. degree in astronautical engineering from the Sapienza University of Rome. He is currently pursuing the Ph.D. degree in engineering of complex systems with the Department of Engineering for Innovation, Università del Salento. He is working on solution methods for optimal planning of active debris removal missions by using evolutionary algorithms. His research activities are also focused on the study of heuristics to estimate transfer costs with the aim of speeding up the trajectory design process. His main research interests include space trajectory optimization, fundamental problems in astrodynamics, and space mission analysis and design.

ALESSANDRO ZAVOLI was born in Rimini, Italy, in 1985. He received the B.S. degree in aerospace engineering, the M.S. degree in space engineering, and the Ph.D. degree in aeronautical and space technologies from the Sapienza University of Rome, in 2013. He is currently a Researcher with the Department of Mechanical and Aerospace Engineering, Sapienza University of Rome. He participated in several editions of the Global Trajectory Optimization Competition, most notably the 6th edition, in 2012, where his team ranked first. Since 2014, his research encompasses spacecraft attitude control. His study concerns the analysis and synthesis of nonlinear controls for underactuated systems, such as magnetically actuated spacecraft. He is currently involved in support and cross-check activities for ESA-ESRIN in the framework of the VEGA launch vehicle program, investigating modern optimization methods, and robust and adaptive control laws for atmospheric flight. His principal research interests include spacecraft trajectory optimization, where he focused on the application of indirect methods for interplanetary mission design and ascent trajectory optimization.

LORENZO FEDERICI received the bachelor's degree in aerospace engineering, the master's degree in space and astronautical engineering, and the Ph.D. degree in aeronautical and space engineering from the Sapienza University of Rome, in 2016, 2018, and 2022, respectively. From March 2021 to August 2021, he was a Visiting Scholar with the Space Systems Engineering Laboratory, The University of Arizona, Tucson, AZ, USA, where he came back as a Postdoctoral Scholar, in November 2022. He is currently a Postdoctoral Research Associate with the Department of Systems and Industrial Engineering, The University of Arizona. He is also involved in the management and development of the space domain awareness software infrastructure with the Space4 Center, The University of Arizona, for the identification, classification, and characterization of artificial objects in LEO, GEO, and cislunar regimes through both optical and radio-frequency observations. His research interests include evolutionary optimization algorithms, graph search methods, and deep learning techniques applied to robust space trajectory design and optimization and autonomous spacecraft guidance, navigation, and control.

GIULIO AVANZINI received the M.Sc. degree in aeronautical engineering and the Ph.D. degree in theoretical and applied mechanics from the Sapienza University of Rome. He was a Research Staff with the Italian Ship Model Basin, in 1997, and then an Assistant Professor with Politecnico di Torino, from 1998 to 2011. He was also a Visiting Professor with the University of Glasgow and the University of Illinois at Urbana–Champaign. He is currently a Professor in flight mechanics with Università del Salento. His research interests include fundamental problems in astrodynamics; space mission analysis and design; spacecraft attitude dynamics and control, especially in underactuated conditions; satellite formation flying (with a focus on tethered formations); nonlinear methods for the analysis of aircraft and spacecraft dynamics; direct and inverse simulation of aircraft and rotorcraft motion; autonomous flight; and performance and sizing of electric and hybrid-electric aircraft and rotorcraft.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza" 2' within the CRUI CARE Agreement