applied sciences

MDPI

Article

An Iterative Optimization Algorithm for Planning Spacecraft Pathways Through Asteroids

Valentino Santucci

Department of International Social and Human Sciences, University for Foreigners of Perugia, 06123 Perugia, Italy; valentino.santucci@unistrapg.it

Abstract: In this article, we explore the use of meta-heuristic algorithms for costly black-box permutation optimization problems. These combinatorial problems are defined by solution spaces that consist of permutations of elements, with an objective function that lacks a closed mathematical representation and is expensive to evaluate. The focus of our investigation is the Asteroid Routing Problem (ARP), which seeks to determine the optimal sequence of asteroids to be visited by a spacecraft while minimizing energy consumption and travel time. Specifically, we assess the performance of a simple algorithm called FAT-RLS, which primarily relies on a randomized local search approach, enhanced with a tabu structure and a mechanism to adjust the perturbation strength. We conducted a series of experiments on well-established instances of the ARP to compare the effectiveness of FAT-RLS against two recognized meta-heuristics designed for this problem, namely, UMM and CEGO. Experiments were conducted in both uninformed and informed settings, where the meta-heuristics are initialized with a specifically designed constructive algorithm for the ARP. The results demonstrate that FAT-RLS is consistently superior to UMM, while there is no conclusive evidence for the comparison with CEGO, though the FAT-RLS results seem slightly better.

Keywords: combinatorial optimization; Asteroid Routing Problem; randomized local search

Citation: Santucci, V. An Iterative Optimization Algorithm for Planning Spacecraft Pathways Through Asteroids. *Appl. Sci.* **2024**, *14*, 10987. https://doi.org/10.3390/app142310987

Received: 25 October 2024
Revised: 5 November 2024
Accepted: 13 November 2024
Published: 26 November 2024

1. Introduction and Related Work

Costly black-box optimization poses significant challenges across various fields, including engineering and finance, as it involves problems where the objective function does not have an explicit formal representation and demands considerable computational resources in terms of time, memory, or monetary cost. In these cases, algorithms can only learn knowledge about the problem by performing multiple evaluations of the objective function. However, because typically these evaluations are expensive, the algorithms often face strict computational budget limitations, thus restricting the number of allowable evaluations.

For continuous variables, Bayesian optimization methods are frequently employed, as discussed in [1,2]. These techniques build a surrogate model of the objective function, usually using a Gaussian process or a Kriging model, in an iterative manner. By performing a large number of computationally cheap evaluations of the surrogate function, they identify potential candidate solutions, which are then tested against the actual objective function. Therefore, the true evaluations have a twofold purpose. On one hand, they allow the discovery of solutions of the problem at hand, and on the other hand they are used to improve the surrogate model for the next iteration. This strategy minimizes the need for expensive evaluations and enables the use of traditional optimization techniques that rely on the surrogate model for guidance.

In this study, we deal with a category of combinatorial optimization problems known as permutation problems, where solutions consist of permutations of elements from a specified set [3,4]. Specifically, we examine a recently introduced black-box permutation optimization problem, the Asteroid Routing Problem (ARP), presented by López-Ibáñez et al. in [5].

The ARP is inspired by the *11th Global Trajectory Optimisation Competition* (for more details, see the official competition webpage athttps://sophia.estec.esa.int/gtoc_portal/?page_id=782, accessed on 10 November 2024) and involves a spacecraft that, after being launched from Earth, has to visit a given set of asteroids while minimizing both energy consumption and the total time required. Although this study does not focus on the astrophysical details of the problem, it is important to highlight that the ARP's primary motivation is linked to the growing demand for technological devices, such as mobile phones and computers, which is causing a rapid depletion of Earth's mineral resources like silicon, boronite, and quartz. As a result, asteroid or near-Earth object mining is emerging as a potential solution to address the scarcity of such mineral resources.

Interestingly, the ARP is just one of many computational optimization problems related to space engineering. Closely related problems are active debris removal, which has been described in [6], and the design of rendezvous tours through the Sun–Jupiter asteroid, depicted in [7]. Other variants of the well-known Traveling Salesman Problem have been proposed for minimizing the energy and time required for on-orbit servicing [8], and for optimizing debris selection and trajectories in debris collecting missions [9].

From a computational point-of-view, the ARP is a permutation optimization problem where evaluating the objective function necessitates executing a computationally expensive procedure. This makes the ARP an interesting benchmark for costly black-box permutation optimization. Other related benchmark problems have been proposed in [10] and as competition challenges by the European Space Agency (ESA), as reported, e.g., in [11–14]. Moving out from the field of space engineering, other related problems are fourth-party logistics [15] and the schedule risk control problem of IT outsourcing projects [16].

Given the combinatorial nature of the ARP, traditional Gaussian-based Bayesian optimization techniques are not applicable. In [5], two specialized algorithms were used to tackle the ARP: (i) CEGO (Combinatorial Efficient Global Optimization)a distance-based Bayesian optimization method tailored for combinatorial problems; and (ii) UMM (Unbalanced Mallows Model) an estimation distribution algorithm specifically designed for costly permutation optimization problems.

Though CEGO and UMM are, to the best of our knowledge, the only algorithms that have been applied to the ARP, it is interesting to mention other algorithmic approaches for related space trajectory optimization problems, such as the Ant Colony Optimization variant proposed in [10], the tree search methods considered in [12,17], a reinforcement learning approach introduced in [18], the multi-agent genetic algorithm used in [19], and the integer linear programming approach considered in [20].

Santucci et al. [21] recently proposed a simple yet effective approach to permutation optimization problems, termed FAT-RLS, that outperformed both UMM and CEGO when a low budget for evaluation was considered. However, FAT-RLS was applied to standard permutation problems such as the Linear Ordering Problem and the Permutation Flowshop Problem. Therefore, the goal of this work is to investigate the application of FAT-RLS to a real costly permutation optimization problem, such as the ARP.

The remainder of the article is structured as follows. Section 2 introduces permutation problems along with other preliminary concepts. The computational details of the ARP are provided in Section 3, followed by a description of the FAT-RLS algorithm in Section 4. Section 5 offers a brief overview of the competitor algorithms used in the experiments, while Section 6 discusses the conducted experiments. Finally, conclusions and potential future research directions are presented in Section 7.

2. Permutation Optimization Problems

Permutations are flexible algebraic structures that have applications across numerous domains due to their ability to model various concepts, such as orderings, rankings, one-to-one mappings between sets, and tours or cycles within a given set of locations.

Permutations, typically referred to using Greek letters like σ or π, can be formally defined as bijective functions mapping the set of the first n natural numbers, i.e., $[n] = \{1, 2, \ldots, n\}$,

onto itself. While multiple notations for permutations exist, the widely used *single-line notation* expresses a permutation σ as a sequence of all different items, as follows:

$$\sigma = \langle \sigma(1), \sigma(2), \ldots, \sigma(n) \rangle, \tag{1}$$

where $\sigma(i) \in [n]$ denotes the item at position $i \in [n]$ in the sequence, and it holds that $\sigma(i) \neq \sigma(j)$ for any choice of $i, j \in [n]$ such that $i \neq j$.

There are $n!$ permutations of length n and their set is denoted as Σ_n, which is also referred to as the *symmetric group*. Indeed, Σ_n is a group in the algebraic sense because there exists a binary operation between permutations, known as *composition*, that is associative and admits an identity element along with an inverse for every permutation. Given $\sigma, \pi \in \Sigma_n$, their composition $\sigma\pi$ is defined as $\sigma\pi(i) = \sigma(\pi(i))$ for all $i \in [n]$. The identity permutation is usually denoted as $\iota = \langle 1, \ldots, n \rangle$, while the inverse of σ is the permutation σ^{-1} that satisfies the equivalences $\sigma^{-1}\sigma = \sigma\sigma^{-1} = \iota$. Note that, in general, the commutative property does not hold for all the permutations of Σ_n, therefore the group is not Abelian. Nevertheless, this group structure is a fundamental characteristic of permutations that has recently also been used for the development of both swarm and evolutionary algorithms [22,23].

The objective of a permutation optimization problem is to either minimize or maximize a real-valued function $f : \Sigma_n \to \mathbb{R}$. Because of the combinatorial nature of the domain of the objective function, permutation problems do not admit gradient-based algorithms (though model-based gradient search methods are however possible [24]). Additionally, often real-world objective functions require running simulations or experiments that cannot be expressed in closed analytical form, thus resulting in black-box optimization problems. In these cases, an effective algorithm can only gain knowledge about the problem at hand by testing various permutation solutions and observing the corresponding objective values. Even worse, black-box objective functions are often expensive to evaluate (in terms of time, memory, and other resources); therefore, a good algorithm should be able to locate a satisfactory solution in a limited number of function evaluations.

Though being in a black-box context, it is often possible to exploit the problem description in order to infer the key features of a permutation that are relevant for the problem. In fact, permutation problems can be broadly classified into two main categories:

- *Ordering problems*, where the goal is to find the optimal arrangement of items within a set A;
- *Assignment problems*, whose objective is to determine the best one-to-one mapping between two equally sized sets A and B.

Based on this classification, it is often possible to choose a movement operator or another operator (for example, exchange moves for assignment problems or insertions for ordering problems [25]). Similar choices can be inferred also for crossovers or other meta-heuristic operators [26].

Though these two categories may not cover every possible permutation problem, they encompass the most commonly encountered ones in the scientific literature. For instance, the Linear Ordering Problem (LOP) [27,28] and the Permutation Flowshop Scheduling Problem (PFSP) [29] are classic examples of ordering problems. On the other hand, the Assignment Problem, which can be solved in polynomial time [30], and its more complex NP-hard variant, the Quadratic Assignment Problem (QAP) [31], are typical examples of assignment problems. Additionally, the well-known Traveling Salesman Problem (TSP) [32], although focused on finding an optimal circular tour through a set of cities, is usually considered an ordering problem (simply designating a start/end city in the tour allows the solutions of the TSP to be represented as orderings over the remaining cities).

It is worth noting that the categorization ordering/assignment problems is not always clear-cut. Indeed, both LOP and TSP have been shown to be special cases of the QAP [31]. This indicates that the boundary between the ordering and assignment nature of permutation problems is not yet fully understood.

Two commonly employed methods for representing solutions to permutation problems are the traditional linear representation and the use of permutation matrices. The linear representation is based on the single-line notation of Equation (1), while the permutation matrix encoding represents a permutation as a binary matrix, where each row and column are one-hot vectors (i.e., they have one 1-entry and $n-1$ 0-entries). While the linear representation is applicable to both kinds of permutation problems, the matrix representation does not inherently encode any ordering information; rather, it only reflects the pairings between row and column indices. Therefore, in this work, we utilize the linear genotypic representation for permutation solutions.

However, as described in [21], two equivalent linear representations exist in ordering problems: the *ordering representation* and the *ranking representation*. As previously discussed, an ordering problem aims to optimally arrange a set A of n items based on a given objective function. The ordering representation maps positions to items in A, whereas the ranking representation maps items of A to their respective positions. Since the items in A are denoted by identification numbers in $[n]$, it is easy to confuse the two representations. However, both representations convey the same information and can be interconverted through a simple permutation inversion. In light of this, it is essential to specify which representation is being used when defining both the objective function and the algorithmic components used to tackle permutation problems. Using the wrong representation without the necessary conversion can lead to subtle errors that are difficult to detect but may significantly hinder the effectiveness of the algorithm. In this work, we adopt the ordering representation.

Furthermore, it is important to mention that within the permutation space, there exist various definitions of elementary moves which can be utilized to develop local search strategies or genetic operators. The most classical ones are swapping two adjacent elements (*adjacent swaps*), swapping two arbitrary elements (*exchanges*), and shifting an element to a new position (*insertions*). Exchanges are typically effective in assignment problems, while adjacent swaps and insertions are more suited for problems related to ordering sequences. Additionally, it easy to see that (i) adjacent swaps are special cases of insertions, and (ii) a single insertion can be expressed as a sequence of multiple adjacent swaps. As a result, insertion moves are frequently preferred when designing algorithms aimed at solving ordering problems.

3. Asteroid Routing Problem

López-Ibáñez et al. [5] first presented the Asteroid Routing Problem (ARP) as a testing framework for computationally intensive optimization problems involving permutations and black-box evaluations. The problem focuses on determining an optimal trajectory for a spacecraft that departs from Earth and needs to sequentially visit each member of a specified collection of n asteroids $A = \{a_1, a_2, \ldots, a_n\}$. The goal is to simultaneously minimize propellant usage and the duration of the mission.

From a computational point of view, the ARP can be seen as a bilevel optimization problem: the outer task focuses on finding the sequence for visiting all the asteroids in the set A, while the inner task involves computing, at each step in the sequence, the parking and transit times required by the spacecraft to reach the next asteroid.

A solution of the ARP is represented as a pair $(\sigma, \mathbf{t})$, where $\sigma \in \Sigma_n$ defines the sequence in which the asteroids in A are visited, and the vector $\mathbf{t} \in \mathbb{R}^{2n}_{\geq 0}$ specifies the parking and transit times. More precisely, let the Earth be denoted as the asteroid a_0, we have for each step $i = 1, 2, \ldots, n$ that the spacecraft stays in the orbit of (or is launched from, in case $i = 1$) asteroid $a_{\sigma(i-1)}$ for a time t_{2i-1}—called the *parking time*—then it travels from the orbit of $a_{\sigma(i-1)}$ to the orbit $a_{\sigma(i)}$ for a time t_{2i}—called the *transit time*.

Considering the spacecraft to be launched from the Earth at epoch τ_0, its launch epoch from the orbit of every asteroid in the sequence is $\tau_i = \tau_0 + \sum_{j=1}^{2i} t_j$, for $i = 1, 2, \ldots, n$. Between two consecutive epochs τ_{i-1} and τ_i, the spacecraft receives two impulsive maneuvers Δv_{2i-1} and Δv_{2i} for, respectively, transiting from $a_{\sigma(i-1)}$ to $a_{\sigma(i)}$ and aligning its orbit with

that of $a_{\sigma(i)}$. Impulsive maneuvers are rapid changes in velocity of the spacecraft, typically achieved by firing thrusters briefly, thus allowing the spacecraft to change its trajectory or orbit quickly. They are measured in meters per second, thus $\Delta\mathbf{v} \in \mathbb{R}^{2n}$. The magnitude of the impulses are calculated as

$$(\Delta v_{2i-1}, \Delta v_{2i}) = \text{Lambert}(a_{\sigma(i-1)}, a_{\sigma(i)}, \tau_{i-1}, t_{2i}), \tag{2}$$

i.e., they are the solutions of the Lambert problem: a two-point boundary value problem commonly used for preliminary mission details. Its mathematical details are not relevant for this work, though they can be found, e.g., in [5,33].

Once the permutation σ is determined as the solution of the outer task, Equation (2) is repeatedly solved in the inner task in order to find the impulsive maneuvers for each step $i = 1, 2, \ldots, n$. Therefore, after sequentially solving the n continuous optimization problems, we have the values for the $2n$-dimensional vectors $\mathbf{t}$ and $\Delta\mathbf{v}$.

In [5], the continuous problems in the inner task are solved by means of the sequential least squares programming (SLSQP) algorithm, which is a well-known quasi-Newton approach for non-linear optimization, having the peculiarity of being a deterministic method [34]. This aspect enables focusing on the outer task of the ARP, as the computation of the values of $\mathbf{t}$ and $\Delta\mathbf{v}$ is treated as an internal and deterministic process that solely depends on the given asteroid ordering σ. Its objective function to be minimized is therefore defined as

$$f(\sigma) = \sum_{i=1}^{2n} |\Delta v_i| + \frac{2 \text{ km/s}}{30 \text{ days}} \cdot \sum_{i=1}^{2n} t_i, \tag{3}$$

where the first term is directly related to the energy required by the spacecraft to execute all the maneuvers, while the second term accounts for the total time required to complete the mission. The constant preceding the second summation has been empirically determined in [5].

In conclusion, an ARP instance is fully characterized by the orbital parameters of the Earth and all asteroids, the initial epoch τ_0, and a gravitational constant. An instance generator is described in [5], which accepts as arguments the number of asteroids n and the seed for the random number generator. Implementations for both the ARP objective function and the instance generator can be found at https://doi.org/10.5281/zenodo.5725837 (accessed on 10 November 2024).

4. The Algorithm FAT-RLS

The Fast Adaptive Tabu-based Randomized Local Search (FAT-RLS), proposed in [21], is a meta-heuristic algorithm tailored for solving costly black-box permutation problems. It builds upon the traditional framework of randomized local search, to which it introduces two key enhancements: an adaptive strategy to adjust its perturbation strength, and a tabu-based methodology that prevents the repetition of redundant perturbations.

Randomized local search (RLS) consists of a trajectory-based approach that iteratively evolves a solution of the problem at hand. At each iteration, the current solution is compared to one of its neighbors selected at random. If the objective value is improved, the trial neighbor replaces the current solution for the next iteration. Despite its straightforward nature, RLS has been widely investigated within the theoretical framework of evolutionary computation [35], but it is less frequently applied in practical situations. We opted to utilize RLS as the search mechanism for our proposed algorithm due to its highly exploitative nature and the fact that it only requires a single evaluation of the objective function per iteration. This characteristic makes RLS particularly suited for maximizing the effectiveness of the limited evaluation budget in costly black-box scenarios. Additionally, each iteration merely involves applying a movement operator to the current solution, resulting in minimal computational overhead. This advantage is especially significant when compared to the computational demands of model learning and updating processes found in Bayesian methods such as those described in [36,37].

As our primary focus is on the ARP, which can be classified as a permutation ordering problem (as outlined in Section 3), we utilize insertion moves to perturb the solutions. For an ARP instance of size n and a solution $\sigma \in \Sigma_n$, an insertion move (i, j), where $i, j \in [n]$, entails shifting the element $\sigma(i)$ to position j within σ. Let π represent the permutation resulting from applying the insertion (i, j) to σ, in the case of a forward insertion, where $i < j$, the transformation can be expressed as

$$\pi(k) = \begin{cases} \sigma(k) & \text{if } k < i \text{ or } k > j, \\ \sigma(k+1) & \text{if } i \leq k < j, \\ \sigma(i) & \text{if } k = j. \end{cases} \tag{4}$$

Incontrast, for a backward insertion, where $i > j$, the transformation is given by

$$\pi(k) = \begin{cases} \sigma(k) & \text{if } k < j \text{ or } k > i, \\ \sigma(k-1) & \text{if } j < k \leq i, \\ \sigma(i) & \text{if } k = j. \end{cases} \tag{5}$$

To make clearer how insertions work, we provide two illustrative examples in Figure 1.

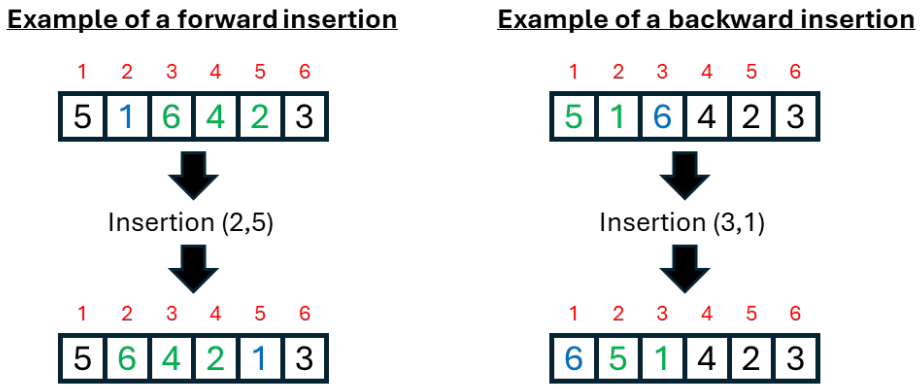

Figure 1. Illustrativeexamples for forward and backward insertions.

Equations (4) and (5) illustrate that an insertion move (i, j) rearranges $|i - j| + 1$ items within a permutation, which makes the insertion effectively equivalent to a sequence of $|i - j|$ adjacent swaps. In the case of forward insertions, as shown in Equation (4) (with the backward insertion case being similar), the insertion (i, j) corresponds to the following series of $|i - j|$ adjacent swaps:

$$\text{AdjSwap}(i, i+1),\ \text{AdjSwap}(i+1, i+2),\ \ldots,\ \text{AdjSwap}(j-1, j).$$

A notable implication is that the Kendall's-τ distance between π and σ (i.e., the number of discordant pairs of elements in the two permutations) equals $|i - j|$. Consequently, we define $d = |i - j|$ as the perturbation strength for a generic insertion (i, j).

In FAT-RLS, instead of randomly choosing a single insertion move, as is typical in traditional randomized local search methods, we regulate the perturbation strength d at each iteration. Specifically, FAT-RLS implements an adaptive perturbation strength strategy that requires two hyperparameters: the *initial perturbation strength* d_{ini} and the *steepness factor* β. Initially, the perturbation strength d is set to d_{ini}. FAT-RLS then perturbs the current solution by randomly selecting an insertion move (i, j) such that $|i - j| = d$. The value of d decreases monotonically according to a "skewed S-shaped" function influenced by β. This approach encourages a gradual transition from explorative behavior to exploitative behavior throughout the iterations, which is a widely recognized best practice in the design of meta-heuristic algorithms [38].

For $\beta \geq 1$, the "skewed S-shaped" $s : [0,1] \to [0,1]$ is defined as

$$s_\beta(p) = 1 - \frac{1}{1 + \left(\frac{1-p}{p}\right)^\beta}. \tag{6}$$

Consequently, the perturbation strength d for each iteration is calculated by rounding to the nearest integer the product of d_{ini} and the value of the s function when applied to the evolution percentage p (i.e., the ratio of the current iteration number to the total iteration budget). Figure 2 illustrates the behavior of the function $s_\beta(p)$ for various values of β. Specifically, when $\beta = 1$, the function simplifies to $s_1(p) = 1 - p$, resulting in a linear decline in perturbation strength. As β increases, the curve becomes steeper in the middle section, which prolongs the initial exploration phase and facilitates a more abrupt and rapid transition to the final exploitative phase.

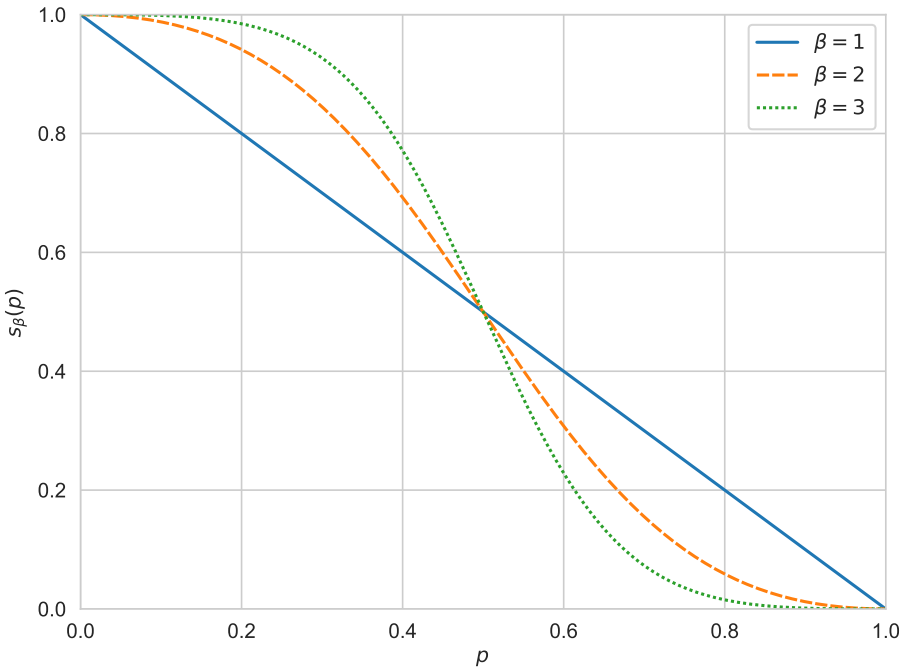

Figure 2. The "skewed S"-shaped function for $\beta = 1, 2, 3$.

Furthermore, since FAT-RLS is designed for low-budget optimization scenarios, we incorporate a tabu-based mechanism. This mechanism serves not only to prevent the trajectory of an FAT-RLS execution from revisiting previously explored solutions but also to ensure that the same elements of the current permutation are not shifted too frequently by the perturbation operation. Hence, FAT-RLS maintains a tabu queue, denoted as TQ, with a maximum size of k, which is a hyperparameter of the algorithm. During each iteration, an insertion (i, j) cannot be selected if the element $\pi(i)$ designated for the shift is marked as tabu (i.e., if $\pi(i) \in TQ$). When a non-tabu insertion (i, j) is selected and executed on the current permutation, the shifted element is added to the tabu queue TQ. If the queue reaches its maximum capacity, the oldest item is removed to accommodate the new entry.

In summary, FAT-RLS utilizes a straightforward randomized local search framework that conducts a single objective function evaluation per iteration. To enhance its effectiveness in low-budget contexts, the algorithm implements both an adaptive perturbation strength strategy and a tabu-based mechanism. The adaptive perturbation strength is implemented by considering insertion moves with a "shift length" that decreases during the iterations, while the tabu-based mechanism is realized by maintaining a queue data structure containing the items shifted during the last iterations. Therefore, these compo-

nents serve to limit the number of applicable perturbations in each iteration, facilitating a more rapid convergence toward potentially promising solutions.

For completeness, we provide the pseudocode for FAT-RLS in Algorithm 1. This algorithm is designed to minimize (without loss of generality) an objective function of the form $f : \Sigma_n \to \mathbb{R}$. It accepts three hyperparameters as input: $d_{\text{ini}} \in [1, n)$, $\beta \geq 1$, and $k \leq n$, as previously described.

Algorithm 1 Pseudocode of FAT-RLS

procedure FAT-RLS(d_{ini}, β, k)
  /* *Variables Initialization* */
  $\sigma \leftarrow$ random permutation from Σ_n
  Evaluate $f(\sigma)$
  $nfev \leftarrow 1$
  $TQ \leftarrow \emptyset$
  /* *Main loop* */
  while $nfev < budget$ **do**
    /* *Compute perturbation strength* */
    $p \leftarrow nfev/budget$
    $d \leftarrow \text{round}\big(\ 1 + s_\beta(p) \cdot (d_{\text{ini}} - 1)\ \big)$
    /* *Perturbation* */
    $\pi \leftarrow \sigma$
    Sample i, j s.t. $|i - j| = d$ and $\pi(i) \notin TQ$
    Perform the insertion (i, j) on π
    /* *Update current solution* */
    Evaluate $f(\pi)$
    if $f(\pi) < f(\sigma)$ **then**
      $\sigma \leftarrow \pi$
    end if
    $nfev \leftarrow nfev + 1$
    /* *Update tabu queue* */
    Push item $\pi(j)$ into TQ
    if $|TQ| = k$ **then**
      Pop oldest item from TQ
    end if
  end while
  return $\sigma, f(\sigma)$
end procedure

5. Other Algorithms for the ARP

The objective of this study is to evaluate the performance of the FAT-RLS algorithm on the ARP. To achieve this, we conduct an experimental comparison of FAT-RLS against four competing algorithms as follows: a basic random search (RS) method used as the baseline algorithm; Greedy Nearest Neighbor (GNN)—a heuristic scheme specifically introduced in [5] for the ARP; Combinatorial Efficient Global Optimization (CEGO)—a Bayesian algorithm for combinatorial problems introduced in [36]; and the Unbalanced Mallows Model (UMM)—an estimation-of-distribution algorithm described in [39].

In the work of [5], both CEGO and UMM were tested in two configurations: the *uninformedsetting*, where the algorithms were executed in their standalone forms; and the *informed setting*, where initial solutions were generated using GNN instead of being created randomly.

The next four subsections provide a brief overview of the four competitor algorithms, while the last subsection discusses the key differences of FAT-RLS with respect to UMM and CEGO.

5.1. Random Search (RS)

The basic random search procedure, referred to as RS, serves as a baseline method in this study. RS generates a specified number of permutation solutions uniformly at random by employing the well-known Fisher–Yates algorithm [40]. After generating these solutions, it evaluates the objective function for all of them and selects the best one.

Despite its simplicity, RS represents a noteworthy baseline method due to its ability to evaluate all solutions in parallel (at least theoretically) [41,42].

5.2. Greedy Nearest Neighbor (GNN)

The Greedy Nearest Neighbor heuristic, henceforth referred to as GNN, is inspired by the well-known nearest neighbor heuristic used in the Traveling Salesman Problem (TSP) [43].

The core concept of GNN is to construct a reasonably effective permutation of the asteroids by iteratively visiting the asteroid that is nearest—under Euclidean distance—to the last visited one. In order to calculate the distances, at each time step the locations of all unvisited asteroids need to be computed.

The pseudocode for GNN is outlined in Algorithm 2. For more detailed information, we direct interested readers to [5]. Moreover, an implementation of the GNN for ARP is available at https://doi.org/10.5281/zenodo.5725837 (accessed on 10 November 2024).

Algorithm 2 The GNN constructive heuristic

procedure GNN
  $s \leftarrow a_0$ ▷ Orbit of the Earth
  $\tau \leftarrow \tau_0$ ▷ Initial epoch for the spacecraft
  $U \leftarrow [n]$ ▷ Set of the unvisited asteroids
  for $i \leftarrow 1$ **to** $n-1$ **do**
    $\sigma(i) \leftarrow \arg\min_{j \in U} d_{Eucl}(s, a_j, \tau)$
    $(t_{2i-1}, t_{2i}) \leftarrow \mathrm{SLSQP}(s, a_{\sigma(i)})$ ▷ Inner task
    $\tau \leftarrow \tau + t_{2i-1} + t_{2i}$
    $U \leftarrow U \setminus \{\sigma(i)\}$
    $s \leftarrow a_{\sigma(i)}$
  end for
  return $\sigma, f(\sigma)$
end procedure

5.3. Combinatorial Efficient Global Optimization (CEGO)

The CEGO algorithm [36] builds upon the well-established Efficient Global Optimization (EGO) method [44], adapting it specifically for combinatorial optimization. Originally designed for continuous domains, EGO utilizes a Bayesian approach to iteratively learn a Gaussian process model which represents a surrogate of the true objective function. Contrarily, CEGO targets combinatorial spaces by employing a distance-based combinatorial surrogate model, substituting the classical Euclidean distance used in continuous spaces with a discrete distance function that is more appropriate for the specific search space at hand.

In [37], various distance functions suitable for the permutation space are explored. The available implementation of CEGO begins by generating a small set of initial solutions through a max–min distance procedure, which are then utilized to create an initial surrogate model. Then, a genetic algorithm with operators designed for permutations is used to find optimal or good-enough solutions of the surrogate function. The best solution identified during this search is subsequently evaluated using the true objective function, allowing for the acquisition of information about the objective function, thus allowing for the update of the surrogate model maintained by CEGO. This process is iterated until a specified termination criterion is fulfilled.

Hence, after the initial warm-up phase, each iteration of CEGO involves (i) optimizing the surrogate function, (ii) performing one true objective function evaluation, and (iii) updating the surrogate model. It is important to highlight that, as demonstrated experimentally in [21], updating the surrogate model within the permutation space presents its own set of challenges, necessitating considerable computational time, particularly as the number of training permutations (and their sizes) increases.

For further information regarding the settings of the genetic algorithm used in CEGO, we recommend consulting [37], while an implementation of CEGO can be accessed at https://cran.r-project.org/web/packages/CEGO (accessed on 10 November 2024).

5.4. Unbalanced Mallows Model (UMM)

The UMM algorithm [39] is part of the well-known family of estimation-of-distribution algorithms. It works by iteratively alternating between learning and sampling from a Mallows model, which is a well-established probability model for permutations [45]. Initially, a set of solutions is generated randomly to establish the initial Mallows model. In each subsequent iteration, UMM samples and evaluates a permutation, that is subsequently used to update the Mallows model.

The Mallows model is defined by a mode permutation $\sigma_0 \in \Sigma_n$ and a dispersion parameter $\theta \in \mathbb{R}$. The mode permutation is calculated using the *Unbalanced Borda* procedure, which assigns weights to the sampled solutions in such a way that 90% of the total weight is assigned to the top 10% of samples. The dispersion parameter θ is calculated considering the expected Kendall's-τ distance $E[D]$ between a sampled permutation and σ_0. This expected distance is gradually adjusted from $\binom{n}{2}/2$ to 1 over the course of the iterations.

For additional details about UMM, readers are encouraged to consult [39], while the implementation of UMM can be found at https://zenodo.org/record/4500974 (accessed on 10 November 2024).

5.5. Key differences of FAT-RLS with Respect to UMM and CEGO

The main difference between UMM and CEGO, on the one hand, and FAT-RLS, on the other, is that UMM uses a probability distribution to model a population of permutations, and CEGO adopts a distance-based probability model over permutations to define a surrogate function, whereas FAT-RLS does not use probability models over permutations at all. In contrast, FAT-RLS adopts a greedy randomized local search approach, which is not adopted either by UMM or by CEGO.

Moreover, FAT-RLS operators have a negligible linear complexity, while those of CEGO and UMM require quadratic costs to perform statistical operations over permutations.

Another difference is that FAT-RLS, in contrast to CEGO and UMM, is invariant to monotonic transformations of the objective function. This property is crucial in many practical contexts. For example, consider objective values which are monetary amounts or physical measurements. In real-world applications, it is essential that the algorithm's behavior—and thus the quality of the solution it produces—does not depend on the specific currency or unit of measurement under consideration.

There are, however, a few similarities. For example, CEGO may use the Ulam distance over permutations to construct its surrogate model. This distance metric counts the number of insertions needed to transform one permutation into another. In this sense, CEGO employs the same type of perturbation move used in FAT-RLS, i.e., insertions of items within a permutation.

Moreover, there are notable parallels with UMM. With this regard, let us first note that an insertion move that shifts an item by d positions results in a permutation that is d steps away from the current solution in terms of Kendall's-τ distance (which counts the number of pairwise disagreements between two permutations). In this way, FAT-RLS and UMM, despite being fundamentally different types of algorithms, exhibit somewhat similar search behaviors: both iteratively perturb a permutation by moving to another one at a specific Kendall's-τ distance, which is progressively reduced over iterations. However, there are

key differences, as follows: (i) UMM perturbs the centroid of the Mallows distribution, while FAT-RLS perturbs the best solution to date; (ii) UMM performs a jump at an expected Kendall's-τ distance in an isotropic manner, whereas FAT-RLS uses a more constrained insertion move; (iii) UMM linearly decreases the perturbation strength, while FAT-RLS uses a "skewed S"-shaped decay pattern.

6. Experiments

6.1. Experimental Setup

Experiments were conducted following the settings outlined in [5]. We generated ten ARP benchmark instances utilizing the instance generator and the specific seeds provided by the authors, which enabled us to leverage the results made available for the competitor and baseline algorithms. In particular, we considered two instances for each size n in the set $\{10, 15, 20, 25, 30\}$ using the seeds 42 and 73 (these seeds were arbitrarily chosen by the authors of the ARP instance generator and we adopt them in order to make our analysis comparable with their work). Each instance is denoted using the naming convention *n_seed*.

Two distinct experimental settings were employed as follows:

- *Uninformed setting*. In this scenario, the competing algorithms—FAT-RLS, UMM, and CEGO—are initialized randomly, while RS serves as the baseline method.
- *Informed setting*. Here, the GNN heuristic is utilized both as the baseline method and to generate reasonably effective initial solutions for FAT-RLS, UMM, and CEGO.

In both settings, all algorithms were executed 30 times for each instance, with a budget of 100 objective function evaluations allocated to each execution. Note that the low budget of 100 evaluations is motivated by the fact that the objective function is expensive to evaluate, a typical scenario in real-world black-box optimization. Indeed, we recall that the goal of this work is to investigate the performance of FAT-RLS in a costly black-box permutation problem. Moreover, the same budget was also used by López-Ibáñez et al. in [5].

Furthermore, note that in [5] two variants of UMM and CEGO were assessed: one that evaluates the evolved permutation directly and another that inverts the permutation before evaluation. However, as discussed in Section 2, only one of these variants is valid. Therefore, in this work, we focused on a single version of each competitor algorithm: the UMM-rank and the CEGO-order variants from [5], which also demonstrated superior performance compared to their counterparts UMM-order and CEGO-rank.

Lastly, the hyperparameters for the competing algorithms were configured according to the specifications in their original publications: [21] for FAT-RLS and [5] for CEGO and UMM. More specifically, the FAT-RLS hyperparameters were set to $d_{\text{ini}} = 0.5n$, $\beta = 1.2$, and $k = n$; UMM used weights 0.9 and 0.1, as explained in Section 5.4; while CEGO adopted the Kendall's-τ distance to build its surrogate model.

6.2. Results in the Uninformed Setting

The results of the uninformed experimental setting were analyzed from two different points of view: average performance and peak performance. The average performance of algorithm $\mathcal{A}$ on instance $\mathcal{I}$ is evaluated using the *average relative percentage deviation* (ARPD) measure, defined as follows:

$$ARPD_{\mathcal{A},\mathcal{I}} = 100 \cdot \frac{1}{k} \sum_{i=1}^{k} \frac{val_{\mathcal{A},\mathcal{I},i} - best_{\mathcal{I}}}{best_{\mathcal{I}}}, \tag{7}$$

where $k = 30$ is the number of executions carried out, $val_{\mathcal{A},\mathcal{I},i}$ is the final objective value achieved by algorithm $\mathcal{A}$ on its i-th execution on instance $\mathcal{I}$, and $best_{\mathcal{I}}$ is the best objective value found on instance $\mathcal{I}$ considering all the executions of every algorithm.

The ARPDs are summarized in Table 1. Performance comparisons between FAT-RLS and the competitor algorithms (CEGO, UMM, and RS) are indicated by the following symbols: ▲ marks cases where FAT-RLS significantly outperformed the competitor, ▽

indicates cases where FAT-RLS was significantly outperformed, and the absence of a mark means the difference was not statistically significant. Statistical significance was evaluated using the Mann–Whitney U test [46], with a standard significance level of 0.05.

Table 1 shows that while FAT-RLS does not perform competitively on the two smaller instances with size $n = 10$, it is the best algorithm in average on all the instances where $n \geq 15$. Most notably, the advantage of FAT-RLS is also statistically significant on the larger instances with $n \geq 20$.

Table 1. Average relative percentage deviations for uninformed experiments. Algorithms marked with ▲ were significantly outperformed by FAT-RLS, whereas those indicated by ▽ showed a significant performance advantage over FAT-RLS.

Instance	FAT-RLS	CEGO	UMM	RS
10_42	13.90	9.39 ▽	12.11	20.92 ▲
10_73	16.13	6.55 ▽	11.30 ▽	15.53
15_42	12.84	13.80	17.66 ▲	25.89 ▲
15_73	10.64	13.08	15.57 ▲	23.53 ▲
20_42	10.31	15.35 ▲	20.42 ▲	25.59 ▲
20_73	14.95	24.08 ▲	29.63 ▲	32.66 ▲
25_42	14.42	25.25 ▲	28.60 ▲	34.01 ▲
25_73	9.46	20.26 ▲	25.47 ▲	28.19 ▲
30_42	7.56	19.75 ▲	26.82 ▲	27.77 ▲
30_73	9.15	20.28 ▲	23.89 ▲	27.36 ▲

Regarding peak performances, Table 2 provides the objective values obtained in the best execution of each algorithm for every instance. The values in bold highlight, for each instance, the best value achieved by any algorithm.

Table 2. Best objective values for uninformed experiments. The results highlighted in bold indicate the highest performance achieved for each instance.

Instance	FAT-RLS	CEGO	UMM	RS
10_42	346.7	**346.7**	**346.7**	389.6
10_73	329.4	**324.7**	329.9	343.6
15_42	**505.4**	515.7	516.9	583.1
15_73	**515.1**	523.7	530.7	573.0
20_42	**698.9**	726.9	729.1	777.8
20_73	**676.3**	730.8	810.1	813.2
25_42	**837.4**	945.9	966.8	1075.1
25_73	**889.0**	988.7	1028.7	1089.6
30_42	**1062.3**	1183.0	1260.6	1271.0
30_73	**1098.2**	1212.1	1232.8	1344.2

Table 2 reveals that FAT-RLS achieved the best results in 9 out of 10 instances, with the only exception being instance 10_73, where it achieved the second-best peak performance after CEGO. This observation, along with the trends seen in Table 1, where the ARPDs of FAT-RLS seem to improve on larger instances, suggests that FAT-RLS is capable of delivering competitive results across a wide range of benchmarks. However, it demonstrates a lack of robustness on smaller instances.

In order to further validate these observations, in Figure 3 we provide a boxplot graph showing the distributions of the single relative percentage deviations—actually, the sum's arguments in Equation (7)—obtained by any single run of an algorithm on an instance of a given size. The graph clearly confirms the provided conclusions.

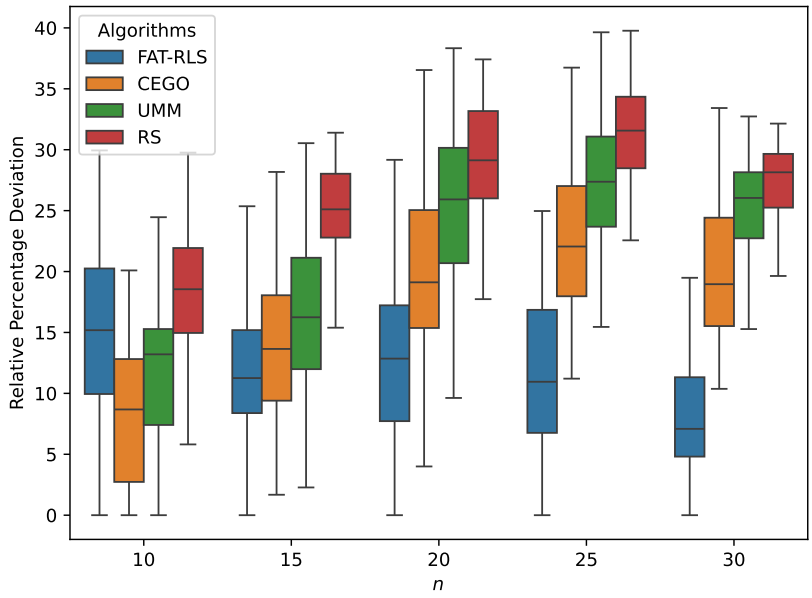

Figure 3. Relative percentage deviations recorded by the executions of the algorithms, aggregated by instance size n, in the uninformed setting.

6.3. Results in the Informed Setting

The analyses of average and peak performance discussed in Section 6.2 were also conducted in the informed setting, where algorithms are initialized with the solution generated by the GNN heuristic. Tables 3 and 4, respectively, present the average and best results achieved by FAT-RLS, CEGO, UMM, and GNN in the informed scenario.

Table 3. Average relative percentage deviations for informed experiments. Algorithms marked with ▲ were significantly outperformed by FAT-RLS, whereas those indicated by ▽ showed a significant performance advantage over FAT-RLS.

Instance	FAT-RLS	CEGO	UMM	GNN
10_42	9.97	8.12 ▽	10.29 ▲	12.86 ▲
10_73	19.69	9.59 ▽	18.54 ▽	22.67 ▲
15_42	1.08	1.28 ▲	2.21 ▲	3.51 ▲
15_73	3.11	2.60	6.38 ▲	12.50 ▲
20_42	6.45	6.92	15.71 ▲	22.12 ▲
20_73	1.29	1.42	4.93 ▲	5.98 ▲
25_42	6.34	9.46 ▲	14.45 ▲	17.50 ▲
25_73	1.62	8.18 ▲	12.93 ▲	13.72 ▲
30_42	3.43	3.75	6.71 ▲	8.27 ▲
30_73	1.93	1.65 ▽	6.85 ▲	7.63 ▲

The results indicate that all three meta-heuristic algorithms—FAT-RLS, CEGO, and UMM—showed improvement over the initial GNN-generated solution. However, the performance gains over the baseline were less substantial compared to the uninformed setting, likely because the initial solution in the informed scenario is heuristically constructed, rather than generated randomly, providing a stronger starting point.

Table 4. Best objective values for informed experiments. The results highlighted in bold indicate the highest performance achieved for each instance.

Instance	FAT-RLS	CEGO	UMM	GNN
10_42	381.3	**346.7**	381.3	391.3
10_73	385.4	**324.7**	367.6	398.3
15_42	493.1	491.4	**490.9**	508.1
15_73	**512.3**	519.9	532.0	576.4
20_42	**689.3**	707.2	729.7	841.7
20_73	659.1	**652.5**	672.8	691.5
25_42	**805.3**	865.7	895.8	946.3
25_73	**807.5**	863.7	885.7	918.3
30_42	**1045.3**	1065.2	1093.9	1131.7
30_73	959.6	**952.1**	1002.0	1024.7

FAT-RLS maintained its dominance in the comparison with informed UMM, significantly outperforming it in 9 out of 10 instances, similar to the uninformed scenario. However, the difference between informed FAT-RLS and informed CEGO is smaller. FAT-RLS significantly outperformed CEGO in three instances, while CEGO outperformed FAT-RLS in three other instances. Furthermore, FAT-RLS achieved the best solution in five instances, while CEGO was the best in four instances, showing a more balanced performance.

A notable observation arises from comparing the results of Tables 2 and 4. On the smaller instances, with $n = 10$, neither GNN nor informed FAT-RLS managed to outperform the results achieved by random search (RS) and black-box FAT-RLS, respectively. This suggests that these smaller instances have a "shallow landscape", where random exploration is sufficient to find good solutions, reducing the advantage provided by heuristic or informed initializations.

Finally, as for the uninformed setting, we validate the provided observations by presenting in Figure 4, a boxplot graph showing the distributions of the relative percentage deviations for the different instance sizes.

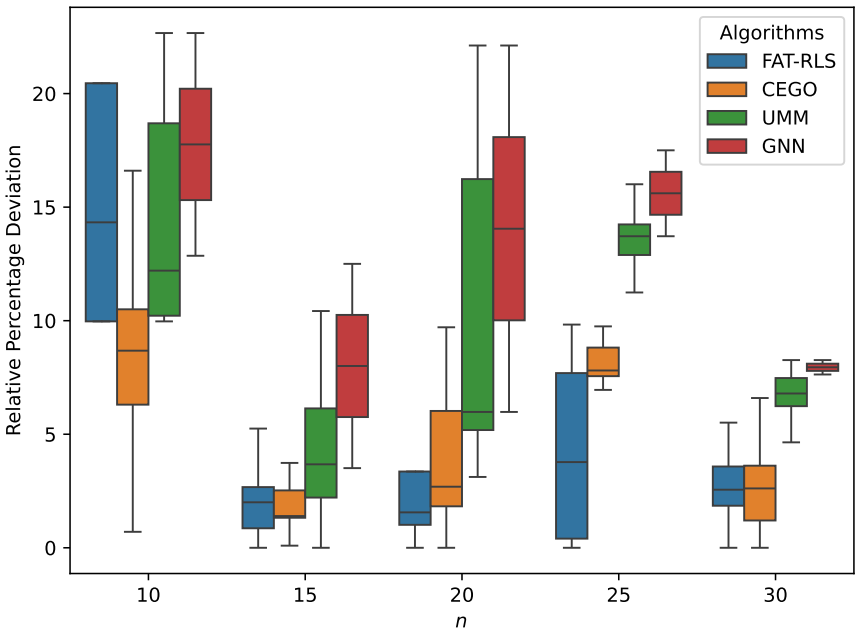

Figure 4. Relative percentage deviations recorded by the executions of the algorithms, aggregated by instance size n, in the informed setting.

7. Conclusions

In this study, we experimentally investigated the performance of FAT-RLS, a straightforward trajectory-based meta-heuristic, in addressing the Asteroid Routing Problem (ARP). FAT-RLS utilizes a well-established randomized local search approach and is enhanced by two additional components: a tabu data structure and a mechanism for adaptive perturbation strength.

We carried out a series of experiments using standard benchmark instances for the ARP to compare the performance of FAT-RLS against two established methods: CEGO, a Bayesian approach designed for combinatorial issues; and UMM, an estimation-of-distribution algorithm specifically created for permutation problems. The experiments were performed in two different contexts: an uninformed setting, where the initial solutions for the competing algorithms were generated randomly; and an informed setting, where a specifically defined heuristic method was employed to initialize the solutions for the competing meta-heuristics.

The results demonstrate that FAT-RLS, despite its straightforward design, surpasses both CEGO and UMM in the uninformed scenario, where the ARP is approached without prior knowledge. However, in the context of heuristic initialization, FAT-RLS significantly outperforms UMM, but there is no conclusive evidence indicating its superiority over CEGO.

Consequently, these findings reinforce the key message previously highlighted in [21]: simple algorithms like FAT-RLS, which primarily rely on robust exploitation strategies, can serve as a viable alternative to more complex techniques in low-budget and high-cost black-box combinatorial scenarios.

Moreover, the results examined, particularly those from the smaller instances, indicate that FAT-RLS lacks robustness in certain situations, highlighting potential areas for improvement. Thus, a direction for future research would be to change the algorithm search mechanism from the randomized local search framework to the $(1+1)$-EA scheme [47]. The core concept is to keep the expected number of insertion moves per iteration at one, while permitting the algorithm to execute multiple insertions within a single iteration. This strategy seeks to maintain a strong level of exploitation while minimizing the chances of the algorithm becoming stuck in a local optimum within the insertion neighborhood. Finally, another interesting future avenue of research is to approach other trajectory optimization problems in the field of space engineering [11–14].

Funding: This research has been partially supported by the research project "Università per Stranieri di Perugia–Finanziamento Dipartimentale alla Ricerca per Progetti di Ricerca di Ateneo–FDR 2024".

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Frazier, P.I. A tutorial on Bayesian optimization. *arXiv* **2018**, arXiv:1807.02811.
2. Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R.D.; Poloczek, M. Scalable global optimization via local bayesian optimization. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 5496–5507.
3. Baioletti, M.; Milani, A.; Santucci, V. An Extension of Algebraic Differential Evolution for the Linear Ordering Problem with Cumulative Costs. In Proceedings of the 14th International Conference on Parallel Problem Solving from Nature—PPSN XIV, Edinburgh, UK, 19–21 September2016; pp. 123–133. https://doi.org/10.1007/978-3-319-45823-6_12.
4. Santucci, V.; Baioletti, M.; Milani, A. An Algebraic Differential Evolution for the Linear Ordering Problem. In Proceedings of the Companion Material Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, 11–15 July 2015; pp. 1479–1480. https://doi.org/10.1145/2739482.2764693.
5. López-Ibáñez, M.; Chicano, F.; Gil-Merino, R. The asteroid routing problem: A benchmark for expensive black-box permutation optimization. In Proceedings of the International Conference on the Applications of Evolutionary Computation (Part of EvoStar), Madrid, Spain, 20–22 April 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 124–140.
6. 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.

7. Stuart, J.R.; Howell, K.C.; Wilson, R.S. Design of end-to-end trojan asteroid rendezvous tours incorporating scientific value. *J. Spacecr. Rocket.* **2016**, *53*, 278–288.
8. 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.
9. Cerf, M. Multiple space debris collecting mission—debris selection and trajectory optimization. *J. Optim. Theory Appl.* **2013**, *156*, 761–796.
10. Simões, L.F.; Izzo, D.; Haasdijk, E.; Eiben, A. Multi-rendezvous spacecraft trajectory optimization with beam P-ACO. In Proceedings of the Evolutionary Computation in Combinatorial Optimization: 17th European Conference, EvoCOP 2017, Amsterdam, The Netherlands, 19–21 April 2017; Proceedings 17; Springer: Berlin/Heidelberg, Germany, 2017; pp. 141–156.
11. Petropoulos, A.E.; Kowalkowski, T.D.; Vavrina, M.A.; Parcher, D.W.; Finlayson, P.A.; Whiffen, G.J.; Sims, J.A. 1st ACT global trajectory optimisation competition: Results found at the Jet Propulsion Laboratory. *Acta Astronaut.* **2007**, *61*, 806–815.
12. Izzo, D.; Hennes, D.; Simões, L.F.; Märtens, M. Designing complex interplanetary trajectories for the global trajectory optimization competitions. In *Space Engineering: Modeling and Optimization with Case Studies*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 151–176.
13. Izzo, D. Problem description for the 9th global trajectory optimisation competition. *Acta Futur.* **2017**, *11*, 49–55.
14. Izzo, D.; López-Ibáñez, M. Optimization challenges at the european space agency. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation, Lisbon, Portugal, 15–19 July 2023; pp. 1399–1415.
15. Lu, F.; Chen, W.; Feng, W.; Bi, H. 4PL routing problem using hybrid beetle swarm optimization. *Soft Comput.* **2023**, *27*, 17011–17024.
16. Lu, F.; Zhu, W.; Bi, H.; Huang, M.; Chen, W.; Zhao, Y. Two-level Tabu-predatory search for schedule risk control of IT outsourcing projects. *Inf. Sci.* **2019**, *487*, 57–76.
17. Zhang, Z.; Zhang, N.; Chen, Z.; Jiang, F.; Baoyin, H.; Li, J. Global Trajectory Optimization of Multispacecraft Successive Rendezvous Using Multitree Search. *J. Guid. Control. Dyn.* **2024**, *47*, 503–517.
18. HOLT, H.J.; Armellin, R.; Baresi, N.; Scorsoglio, A.; Furfaro, R. Low-thrust trajectory design using state-dependent closed-loop control laws and reinforcement learning. In Proceedings of the 2020 AAS/AIAA Astrodynamics Specialist Conference, Virtual Event, 9–12 August 2020.
19. Zuo, M.; Dai, G.; Peng, L. Multi-agent genetic algorithm with controllable mutation probability utilizing back propagation neural network for global optimization of trajectory design. *Eng. Optim.* **2019**, *51*, 120–139.
20. Pritchard, B.; Doyle, D.; Black, J. Linear Programming Trajectory Optimization vs. Artificial Potential Function Methods for Rendezvous and Proximity Operations. In Proceedings of the ASCEND 2020, Virtual Event, 16–18 November 2020; p. 4102.
21. Santucci, V.; Baioletti, M. A fast randomized local search for low budget optimization in black-box permutation problems. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–8.
22. Santucci, V.; Baioletti, M.; Milani, A. An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization. *Swarm Evol. Comput.* **2020**, *55*, 100673.
23. Santucci, V.; Baioletti, M.; Milani, A. Tackling Permutation-based Optimization Problems with an Algebraic Particle Swarm Optimization Algorithm. *Fundam. Informat.* **2019**, *167*, 133–158. https://doi.org/10.3233/FI-2019-1812.
24. Ceberio, J.; Santucci, V. Model-based Gradient Search for Permutation Problems. *ACM Trans. Evol. Learn. Optim.* **2023**, *3*, 1–35.
25. Baioletti, M.; Milani, A.; Santucci, V. Variable neighborhood algebraic Differential Evolution: An application to the Linear Ordering Problem with Cumulative Costs. *Inf. Sci.* **2020**, *507*, 37–52.
26. Larranaga, P.; Kuijpers, C.M.H.; Murga, R.H.; Inza, I.; Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artif. Intell. Rev.* **1999**, *13*, 129–170.
27. Santucci, V.; Ceberio, J. Using pairwise precedences for solving the linear ordering problem. *Appl. Soft Comput.* **2020**, *87*, 105998 https://doi.org/10.1016/j.asoc.2019.105998.
28. Ceberio, J.; Mendiburu, A.; Lozano, J.A. The linear ordering problem revisited. *Eur. J. Oper. Res.* **2015**, *241*, 686–696.
29. Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2007**, *177*, 2033–2049.
30. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97.
31. Silva, A.; Coelho, L.C.; Darvish, M. Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search. *Eur. J. Oper. Res.* **2021**, *292*, 1066–1084.
32. Pop, P.C.; Cosma, O.; Sabo, C.; Sitar, C.P. A comprehensive survey on the generalized traveling salesman problem. *Eur. J. Oper. Res.* **2024**, *314*, 819–835.
33. Izzo, D. Revisiting Lambert's problem. *Celest. Mech. Dyn. Astron.* **2015**, *121*, 1–15.
34. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nat. Methods* **2020**, *17*, 261–272.
35. Doerr, B.; Kelley, A. The Runtime of Randomized Local Search on the Generalized Needle Problem. *IEEE Trans. Evol. Comput.* **2024**, *early access*. https://doi.org/10.1109/TEVC.2024.3453776.
36. Zaefferer, M.; Stork, J.; Friese, M.; Fischbach, A.; Naujoks, B.; Bartz-Beielstein, T. Efficient global optimization for combinatorial problems. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Vancouver, BC, Canada, 12–16 July 2014; pp. 871–878.

37. Zaefferer, M.; Stork, J.; Bartz-Beielstein, T. Distance measures for permutations in combinatorial efficient global optimization. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Ljubljana, Slovenia, 13–17 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 373–383.
38. Chopard, B.; Tomassini, M. *An Introduction to Metaheuristics for Optimization*; Springer: Berlin/Heidelberg, Germany, 2018.
39. Irurozki, E.; López-Ibáñez, M. Unbalanced mallows models for optimizing expensive black-box permutation problems. In Proceedings of the Genetic and Evolutionary Computation Conference, Lille, France, 10–14 July 2021; pp. 225–233.
40. Eberl, M. Fisher-yates shuffle. *Arch. Form. Proofs* **2016**, *2016*, 19.
41. Santucci, V.; Milani, A.; Caraffini, F. An optimisation-driven prediction method for automated diagnosis and prognosis. *Mathematics* **2019**, *7*, 1051.
42. Santucci, V.; Milani, A. Particle Swarm Optimization in the EDAs Framework. In *Soft Computing in Industrial Applications*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 87–96. https://doi.org/10.1007/978-3-642-20505-7_7.
43. Rosenkrantz, D.J.; Stearns, R.E.; Lewis, P.M., II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.* **1977**, *6*, 563–581.
44. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492.
45. Ceberio, J.; Irurozki, E.; Mendiburu, A.; Lozano, J.A. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Trans. Evol. Comput.* **2013**, *18*, 286–300.
46. Hollander, M.; Wolfe, D.A.; Chicken, E. *Nonparametric Statistical Methods*; John Wiley & Sons: Hoboken, NJ, USA, 2013; Volume 751.
47. Doerr, B.; Ghannane, Y.; Brahim, M.I. Towards a stronger theory for permutation-based evolutionary algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference, Boston, MA, USA, 9–13 July 2022; pp. 1390–1398.