A simple yet effective algorithm for the Asteroid Routing Problem

Valentino Santucci[a]
University for Foreigners of Perugia, Perugia, Italy
valentino.santucci@unistrapg.it

Keywords: Expensive Black-Box Permutation Optimization, Asteroid Routing Problem, Randomized Local Search.

Abstract: In this paper we investigate the application of meta-heuristic algorithms in the context of expensive black-box permutation optimization. These problems are characterized by solution spaces composed of permutations of items, where the objective function is not explicitly defined in a closed mathematical form and is costly to evaluate. The benchmark problem under investigation is the Asteroid Routing Problem (ARP), which aims to determine the optimal order for a spacecraft to visit asteroids, starting from Earth's orbit, while minimizing both energy consumption and visit time. In particular, we examine the effectiveness of a simple algorithm, namely FAT-RLS, which is mainly based on the randomized local search scheme, equipped with a tabu structure and a mechanism to regulate the perturbation strength. A series of experiments were performed on accepted instances of the ARP in order to compare the effectiveness of FAT-RLS with respect to two established meta-heuristics for the ARP. The results clearly show that FAT-RLS achieves more effective results both considering a black-box setting and an informed setting, where the meta-heuristics are seeded with a purposely designed constructive algorithm for the ARP.

1 INTRODUCTION

Expensive black-box optimization is a challenging task in a variety of domains, from engineering to finance, and deals with problems where the objective function lacks a mathematical closed form and requires substantial computational resources, not only in terms of time but also memory or money. In such scenarios, algorithms can only gather information about the problem through successive objective function evaluations. However, due to the high cost of evaluations, these algorithms typically operate under strict computational budget constraints, de facto limiting the number of evaluations allowed.

When dealing with continuous decision variables, Bayesian optimization techniques (Frazier, 2018) are commonly used. These methods iteratively construct a surrogate model, typically a Gaussian process or Kriging model, of the objective function at hand. They then conduct a relatively large number of inexpensive surrogate evaluations to identify candidate solutions, which are subsequently evaluated using the true objective function. This approach reduces the amount of costly evaluations and allows to apply classical optimization strategies guided solely by the surrogate model.

[a] https://orcid.org/0000-0003-1483-7998

In this work, we focus on a family of combinatorial optimization problems, the permutation problems, where solutions are formed by permutations of items from a given set. In particular, we investigate a recently proposed benchmark problem for expensive black-box permutation optimization, the Asteroid Routing Problem (ARP) introduced in (López-Ibáñez et al., 2022).

The ARP takes inspiration from the *11–th Global Trajectory Optimisation Competition*[1] and deals with a spacecraft that, once launched from Earth, must visit all the asteroids in a given set by minimizing both its energy consumption and the total time required to complete the journey. While our focus is not on the astrophysical aspects of the problem, it is worth noting that the ARP's main motivation stems from the increasing demand for technological devices like mobile phones and computers, which is leading to a rapid decrease of mineral resources, such as silicon, quartz or boronite, on Earth. Consequently, one potential solution to this issue is the prospect of mining these materials from asteroids or other near-Earth objects.

From a computational perspective, the ARP presents a permutation problem in which the objec-

[1] For complete details about the competition we refer the interested reader to the following webpage https://sophia.estec.esa.int/gtoc_portal/?page_id=782.

tive function requires to run a costly internal computational procedure. As such, it stands as the first benchmark proposed for expensive black-box permutation optimization.

Due to its combinatorial nature, classical Gaussian-based Bayesian methods cannot be applied to the ARP. In (López-Ibáñez et al., 2022), two specific algorithms were employed: CEGO, a distance-based Bayesian approach for combinatorial problems, and UMM, an estimation of distribution algorithm for permutation problems that has been redesigned to work in low budget scenarios.

In (Santucci and Baioletti, 2022), it was shown that both CEGO and UMM were outperformed by a simple algorithm, called FAT-RLS and based on the well known randomized local search scheme, when dealing with classical permutation benchmark problems such as the Linear Ordering Problem or the Permutation Flowshop Problem. Here we are interested in studying the effectiveness of FAT-RLS on a proper expensive black-box permutation problem such as the ARP.

The rest of the paper is organized as follows. In Section 2 permutation problems and other preliminary concepts are presented. Section 3 describes the computational details of the ARP, while Section 4 presents the FAT-RLS algorithm. Section 5 briefly recalls the competitor algorithms used in the experimentation, while the experimental settings and results are presented and discussed in Section 6. Finally, conclusions are drawn in Section 7, where future lines of research are also depicted.

2 PERMUTATION PROBLEMS

Permutations are versatile algebraic objects that find applications in many different fields because of their ability to model a variety of concepts, including orderings and rankings of items, one-to-one mappings between two sets, as well as tours and sets of cycles within a collection of locations.

Permutations, which are usually denoted by Greek symbols such as π or σ, can be mathematically defined as bijective functions onto the set $[n] = \{1, 2, \dots, n\}$. Several notations are available, though the simplest and most common one is the so-called *single line notation*, where a permutation π is represented as a list of all different items, i.e.,

$$\pi = \langle \pi(1), \pi(2), \dots, \pi(n) \rangle, \tag{1}$$

where $\pi(i) \in [n]$ indicates the item in position $i \in [n]$ in the list, ensuring that $\pi(i) \neq \pi(j)$ for any pair $i \neq j$.

The set of all the permutations of degree n is denoted by $\mathbb{S}_n$, which has cardinality $n!$ and is also known as the *symmetric group*. In fact, the standard function composition operation allows to compose two permutations into a third permutation. Given two permutations π and σ, their composition is denoted by $\pi\sigma$ and its elements are $\pi\sigma(i) = \pi(\sigma(i))$, for all $i \in [n]$. The composition operation is associative, admits a unique identity permutation $\iota = \langle 1, 2, \dots, n \rangle$, and each permutation π has a unique inverse permutation π^{-1} such that $\pi\pi^{-1} = \pi^{-1}\pi = \iota$. These three properties prove that $\mathbb{S}_n$ is a group, a characteristic that has been exploited to design both swarm and evolutionary algorithms based on strong mathematical foundations (Santucci et al., 2020).

In permutation optimization problems, the goal is to minimize or maximize a given objective function of the form $f : \mathbb{S}_n \to \mathbb{R}$, i.e., a real-valued function whose domain is the set of permutations $\mathbb{S}_n$. Permutation problems are combinatorial in nature, so f is not differentiable and classical gradient-based algorithms cannot be adopted[2]. Moreover, if f has no analytical definition – for example, because it requires to run an experiment –, the problem is a black-box problem, so an algorithm can only gather information about f by testing multiple permutation solutions and observing the returned objective values. Often, black-box optimization problems are also characterized by an objective function that is expensive to evaluate in terms of time – for example, because it requires to run a computationally intensive simulation –, but also in terms of other resources such as memory or money. Therefore, a suitable algorithm for expensive black-box permutation optimization problems is required to navigate the search space of permutations and should be able to reach a good enough solution within a low budget of objective function evaluations.

Even in a black-box scenario, it is often possible to intuitively identify which features of a permutation are important for a given problem simply by examining the problem description. Indeed, two distinct families of permutation problems can be identified:

- *ordering problems*, where the objective is to determine the optimal sequence of items within a given set A, and
- *assignment problems*, where the aim is to find the best possible one-to-one correspondence between two given sets A and B of equal size.

While these classifications may not be exhaustive, they cover many permutation problems frequently encountered in the scientific literature. For example, the Linear Ordering Problem (LOP) (Santucci and Ceberio, 2020) and the Permutation Flowshop Scheduling

[2]Though model-based gradient search algorithms are possible (Ceberio and Santucci, 2023).

Problem (PFSP) (Santucci et al., 2016) are two typical examples of ordering problems, while the polynomially solvable Assignment Problem (Kuhn, 1955) and its NP-Hard quadratic variant – the Quadratic Assignment Problem (QAP) (Koopmans and Beckmann, 1955) – are common examples of assignment problems. Moreover, also the well known Traveling Salesman Problem (TSP) (Nagata and Kobayashi, 2013), though requiring to determine a circular tour among a given set of cities, can be seen as an ordering problem by (arbitrarily) designating a start/end city for all the tours, so that they can be represented as orderings over the remaining cities.

It is important to note that the distinction between ordering and assignment problems is not always clear-cut. In fact, it is known that both TSP and LOP instances can be seen as special cases of QAP instances (Loiola et al., 2007), thus suggesting that the boundary between the ordering or assignment nature of a permutation problem is not yet well understood.

Two of the most commonly used genotypic representations for permutation problems are the classical linear representation and the permutation matrix representation. The linear representation is essentially the transposition in memory of the single line notation described in Equation (1). In contrast, the permutation matrix representation encodes a permutation as a binary matrix with exactly one 1-entry in each row and each column. While the linear representation is suitable for both ordering and assignment problems, the matrix representation does not directly encode any ordering information, but only the pairings between rows and columns indices. For this reason, in this work we adopt the linear genotypic representation of permutation solutions.

In the context of ordering problems, there are two equivalent linear representations: *ordering representation* and *ranking representation*, as termed in (Santucci and Baioletti, 2022). As mentioned earlier, an ordering problem involves optimally arranging a set A of n items according to a given objective function. The ordering representation maps positions to items of A, while the ranking representation maps items of A to positions. Since the items of A are denoted by identification numbers in $[n]$, it is apparent how easy it is to confuse the two. In fact, both linear representations convey the same information and can be converted into each other by a simple permutation inversion. Anyway, it is crucial to specify which representation is being used when defining the objective function and the algorithms/operators for permutation problems. In fact, using an incorrect representation without the necessary conversion, when required by the objective function or the algorithm/operator at hand, can lead to errors that are difficult to detect but detrimental for the effectiveness. In this work we adopt the ordering representation.

Finally, it is worth noting that in the permutation space, different movement operators are available to build local search schemes or genetic operators such as mutation. The three most commonly used classes of movement operators are: swaps of two adjacent items (also called *adjacent swaps*), swaps of two generic items (also called *exchanges*), and shifts of an item to another position (also called *insertions*). While exchanges are known to be suitable for assignment problems, adjacent swaps and insertions are appropriate for ordering problems (Baioletti et al., 2020). Moreover, it is easy to see that insertion moves include adjacent swaps as special cases, and a single insertion move is equivalent to a chain of multiple adjacent swaps. Therefore, insertion moves are usually adopted in the design of algorithms for ordering problems.

3 THE ASTEROID ROUTING PROBLEM

The Asteroid Routing Problem (ARP) was introduced in (López-Ibáñez et al., 2022) as a benchmark for expensive black-box permutation optimization. It involves planning a route for a spacecraft that, once launched from Earth, must visit all the asteroids in a given set of n asteroids $A = \{a_1, a_2, \dots, a_n\}$ in such a way that it minimizes both its fuel consumption and the total time required to complete the journey.

The ARP is a bilevel optimization problem consisting of two nested tasks. The outer task involves determining an ordering of the asteroids in A, while the inner task requires calculating the parking and transit times for reaching each asteroid in the order given by the outer task.

A solution for the bilevel ARP is a pair $(\pi, \mathbf{t})$, where: $\pi \in \mathbb{S}_n$ encodes the ordering of A, while the vector $\mathbf{t} \in \mathbb{R}^{2n}_{\geq 0}$ encodes the parking and transit times for each asteroid. More specifically, assuming that a_0 represents the Earth, for each step $i \in [n]$:

- $a_{\pi(i)}$ is the i-th asteroid visited by the spacecraft,
- t_{2i-1} is the parking time spent by the spacecraft in the orbit of asteroid $a_{\pi(i-1)}$, and
- t_{2i} is the transit time required for the spacecraft to travel from the orbit of asteroid $a_{\pi(i-1)}$ to the orbit of asteroid $a_{\pi(i)}$.

To formalize the objective function of the ARP, we also consider the following auxiliary variables:

- τ_i, for $i \in \{0, 1, \dots, n-1\}$, is the launch epoch from the orbit of asteroid $a_{\pi(i)}$;
- Δv_{2i-1} and Δv_{2i}, for $i \in [n]$, are the impulses required for the maneuvers to insert the spacecraft into the transit orbit between asteroids $a_{\pi(i-1)}$ and $a_{\pi(i)}$, and then into the parking orbit of asteroid $a_{\pi(i)}$, respectively.

The spacecraft is on Earth at start epoch τ_0, which is given by the ARP instance, while the other launch epochs are computed as follows:

$$\tau_{i-1} = \tau_0 + \sum_{j=1}^{2i-1} t_j. \tag{2}$$

For each step $i \in [n]$, the transit and parking impulses Δv_{2i-1} and Δv_{2i} allow the spacecraft to rendezvous with asteroid $a_{\pi(i)}$. These velocity impulses are computed as solutions of the so-called Lambert's problem, i.e.,

$$(\Delta v_{2i-1}, \Delta v_{2i}) = \text{Lambert}(a_{\pi(i-1)}, a_{\pi(i)}, \tau_{i-1}, t_{2i}), \tag{3}$$

for which we refer the interested reader to (López-Ibáñez et al., 2022) and (Izzo, 2015).

Once the upper-level permutation π is available, Equations (2) and (3) define the inner task, which consists of n continuous optimization problems that, after being sequentially solved, yield the vector of times $\mathbf{t}$ and the vector of the velocity impulses $\Delta\mathbf{v}$.

In (López-Ibáñez et al., 2022), the inner continuous problems are solved using the Sequential Least Squares Programming (SLSQP) algorithm (Virtanen et al., 2020), a deterministic method which allows to focus on the outer task of the ARP problem by treating the computations of the values for $\mathbf{t}$ and $\Delta\mathbf{v}$ as an internal deterministic procedure depending only on the provided asteroid ordering π. Therefore, the two levels of the problem are collapsed into one, and the ARP becomes a black-box permutation problem of ordering nature, whose goal is to minimize the objective function defined on the domain of permutations $\mathbb{S}_n$ as

$$f(\pi) = \sum_{i=1}^{2n} |\Delta v_i| + \frac{2\text{ km/s}}{30\text{ days}} \cdot \sum_{i=1}^{2n} t_i, \tag{4}$$

where: the first summation is proportional to the energy consumed by the spacecraft to perform all the maneuvers, the second summation represents the total time taken by the spacecraft to complete its journey, while the constant in front of the second summation has been experimentally derived in (López-Ibáñez et al., 2022).

Since computing Equation (4) is computationally expensive (because it requires to sequentially solve n internal continuous optimization tasks), the ARP stands as an appropriate benchmark for expensive black-box permutation optimization.

Finally, note that an ARP instance is completely defined by: the orbital parameters of the Earth and all asteroids, the starting epoch τ_0, and a gravitational parameter. An instance generator is provided in (López-Ibáñez et al., 2022) which takes as input a seed parameter and the number of asteroids n, i.e., the size of the generated instance. The implementations of the ARP objective function and the instance generator are available from https://doi.org/10.5281/zenodo.5725837.

4 FAT-RLS

The Fast Adaptive Tabu-based Randomized Local Search (FAT-RLS), introduced in (Santucci and Baioletti, 2022), is an iterative optimization heuristic designed for expensive black-box permutation problems. FAT-RLS is based on the classical randomized local search scheme and incorporates two additional algorithmic components: an adaptive perturbation strength strategy and a tabu-based mechanism to avoid redundant perturbations.

The randomized local search (RLS) is a trajectory-based scheme that evolves a single solution by randomly selecting one of its neighbor at each iteration and, if the neighbor solution is fitter, it replaces the current solution for the next iteration. Due to its simplicity, RLS has been extensively studied in theoretical evolutionary computation (Neumann and Wegener, 2007), though it is not commonly used in practical scenarios. However, we chose RLS as the search engine for the proposed algorithm because it is very exploitative and it requires only one objective function evaluation per iteration. This makes it ideal for effectively utilizing the limited budget of evaluations in an expensive black-box scenario. Moreover, every iteration simply requires applying a movement operator to the current solution, thus resulting in minimal computational overhead. This is particularly advantageous, specially if compared to the computationally intensive model learning and update procedures of Bayesian approaches such as (Zaefferer et al., 2014b; Zaefferer et al., 2014a).

Since our focus is on the ARP that, as detailed in Section 3, can be classified as a permutation ordering problem, we consider insertion moves to perturb the solutions of FAT-RLS. Given an ARP instance of size n and a solution $\pi \in \mathbb{S}_n$, an insertion (i, j), with $i, j \in [n]$, involves shifting the item $\pi(i)$ to position j in π. Therefore, by denoting with σ the permutation obtained by applying insertion (i, j) to π, we have that, in a forward insertion with $i < j$:

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

while, in a backward insertion with $i > j$:

$$\sigma(k) = \begin{cases} \pi(k) & \text{if } k < j \text{ or } k > i, \\ \pi(k-1) & \text{if } j < k \le i, \\ \pi(i) & \text{if } k = j. \end{cases} \tag{6}$$

Equations (5) and (6) show that an insertion (i, j) rearranges $|i - j| + 1$ items in a permutation, making the insertion equivalent to a series of $|i - j|$ adjacent swaps. Considering forward insertions, as depicted in Equation (5) (though the case of backward insertions is analogous), insertion (i, j) corresponds to the following chain of $|i - j|$ adjacent swaps

$$(i, i+1),\ (i+1, i+2),\ \ldots,\ (j-1, j).$$

A clear consequence is that the Kendall's-τ distance[3] between σ and π is $|i - j|$. Therefore, we consider $d = |i - j|$ as the perturbation strength of a generic insertion (i, j).

In FAT-RLS, rather than randomly selecting one insertion move as in classical randomized local search schemes, we control the perturbation strength d at each iteration. Indeed, FAT-RLS employs an adaptive perturbation strength strategy, requiring two hyperparameters: the *initial perturbation strength* d_{ini} and the *steepness factor* β. After initializing the perturbation strength d to d_{ini}, FAT-RLS perturbs the current solution with a randomly selected insertion (i, j) such that $|i - j| = d$. Meanwhile, d monotonically decreases using a “skewed S-shaped” function regulated by β. This strategy fosters the transition from explorative to exploitative behaviour over iterations, a well known best practice in designing meta-heuristic algorithms (Chopard and Tomassini, 2018).

The “skewed S-shaped” function has both domain and codomain in $[0, 1]$ and, given $\beta \ge 1$, is defined as

$$s_\beta(p) = 1 - \frac{1}{1 + \left(\frac{1-p}{p}\right)^\beta}. \tag{7}$$

Hence, the perturbation strength d at each iteration is computed by rounding the product of the initial strength d_{ini} and the s function applied to the percentage of evolution made (i.e., the ratio of the current iteration number to the allowed iteration budget).

[3] A possible definition for the Kendall's-τ distance between two permutations is the number of adjacent swaps to make one permutation equal to the other.

Figure 1 depicts the function $s_\beta(p)$ across various β values. Specifically, when $\beta = 1$, $s_1(p) = 1 - p$, leading to a linear decrease of the perturbation strength, while as β increases the curve steepens in the central part, thus extending the initial exploration phase and making more abrupt and quick the transition to the final exploitative phase.

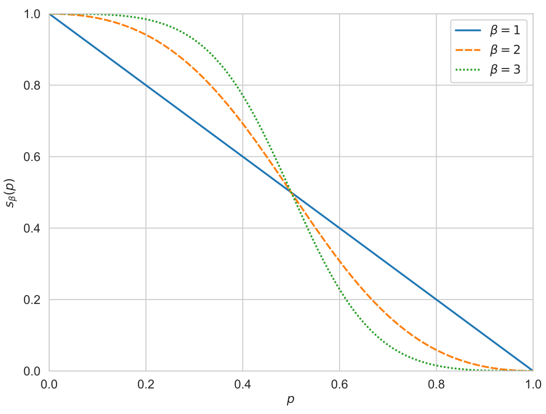

Figure 1: The “skewed S”-shaped function for $\beta = 1, 2, 3$.

Moreover, as FAT-RLS will be employed in a low-budget optimization scenario, we introduce a tabu-based mechanism. Its purpose is not only to avoid the trajectory followed by a FAT-RLS execution from revisiting solutions previously explored, but also to prevent the same items from being shifted too often by the perturbation operation. FAT-RLS maintains a tabu queue TQ of maximum size k, an algorithm hyperparameter. At each iteration, an insertion (i, j) cannot be selected if the item $\sigma(i)$ to be shifted is marked as tabu, i.e., if $\sigma(i) \in TQ$. Then, when a valid insertion (i, j), i.e., such that $\sigma(i) \notin TQ$, is selected and applied to the current permutation, the item $\sigma(i)$ is pushed into TQ and, if the queue is full, the oldest inserted item is removed.

In summary, FAT-RLS adopts a simple randomized local search scheme which only performs one objective function evaluation per iteration and, to improve its effectiveness in the low budget scenario typical of expensive black-box permutation problems, it employs both an adaptive perturbation strength strategy and a tabu-based mechanism which restrict the number of applicable perturbations in each iteration, aiming to accelerate towards potentially good solutions.

Finally, for the sake of completeness, we present the pseudocode of FAT-RLS in Algorithm 1. It aims to minimizes (without loss of generality) an objective function of the form $f : \mathbb{S}_n \to \mathbb{R}$, by taking in input the three hyperparameters $d_{ini} \in [1, n)$, $\beta \ge 1$, and $k \le n$, as previously discussed.

Algorithm 1 Pseudocode of FAT-RLS

```
1:  function FAT-RLS($d_{\text{ini}}$, $\beta$, $k$)
2:      /* Initialization */
3:      $\pi \leftarrow$ random permutation from $\mathbb{S}_n$
4:      Evaluate $f(\pi)$
5:      $nfev \leftarrow 1$
6:      $TQ \leftarrow \emptyset$
7:      /* Main loop */
8:      while $nfev < budget$ do
9:          /* Compute perturbation strength */
10:         $p \leftarrow nfev/budget$
11:         $d \leftarrow$ round $\big( 1 + s_\beta(p) \cdot (d_{\text{ini}} - 1) \big)$
12:         /* Perturbation */
13:         $\sigma \leftarrow \pi$
14:         Sample $i, j$ s.t. $|i - j| = d$ and $\sigma(i) \notin TQ$
15:         Perform the insertion $(i, j)$ on $\sigma$
16:         /* Update current solution */
17:         Evaluate $f(\sigma)$
18:         if $f(\sigma) < f(\pi)$ then
19:             $\pi \leftarrow \sigma$
20:         $nfev \leftarrow nfev + 1$
21:         /* Update tabu queue */
22:         Push item $\sigma(j)$ into $TQ$
23:         if $|TQ| = k$ then
24:             Pop oldest item from $TQ$
25:     return $\pi, f(\pi)$
```

5 OTHER OPTIMIZATION ALGORITHMS

The goal of this work is to investigate the performance of the FAT-RLS algorithm, described in Section 4, on the ARP, i.e., the permutation ordering problem detailed in Section 3. To this end, we experimentally compare FAT-RLS against four competitor algorithms: a simple Random Search (RS) scheme used as a baseline competitor, the constructive heuristic called Greedy Nearest Neighbor (GNN) specifically introduced in (López-Ibáñez et al., 2022) for the ARP, the Bayesian algorithm known as Combinatorial Efficient Global Optimization (CEGO), introduced in (Zaefferer et al., 2014b), and the estimation of distribution algorithm termed Unbalanced Mallows Model (UMM), introduced in (Irurozki and López-Ibáñez, 2021).

In (López-Ibáñez et al., 2022), both UMM and CEGO were executed under two settings: the *uninformed* setting, where the standalone versions of the algorithms were run, and the *informed* setting, where their initial solutions were created using GNN instead of being generated randomly.

The four competitor algorithms are briefly described in the following subsections.

5.1 Random Search (RS)

The trivial random search procedure, denoted by RS, is considered as a baseline method in this work. RS generates a given number of permutation solutions uniformly at random using the well known Fisher Yates algorithm (Eberl, 2016), then it evaluates the objective function on all the generated solutions and returns the best one. Although trivial, RS is an interesting baseline method because all solutions can be evaluated in parallel (at least in principle).

5.2 Greedy Nearest Neighbor (GNN)

The Greedy Nearest Neighbor heuristic, from now on referred to as GNN[4], is inspired from the well known nearest neighbor heuristic for the TSP (Rosenkrantz et al., 1977).

The main idea behind GNN is that a reasonably effective permutation of the asteroids can be formed by repeatedly visiting the asteroid which is closest, in terms of Euclidean distance, to the last visited one, after determining the positions of all unvisited asteroids at the time the spacecraft is arrived in the last visited one.

The pseudocode of GNN is provided in Algorithm 2, while for further details we refer the interested reader to (López-Ibáñez et al., 2022).

Algorithm 2 Pseudocode of the GNN heuristic

```
1:  function GNN
2:      $s \leftarrow a_0$                                   ▷ Earth's orbit
3:      $\tau \leftarrow \tau_0$                        ▷ Epoch of the spacecraft
4:      $U \leftarrow [n]$                                 ▷ Unvisited asteroids
5:      for $i \leftarrow 1$ to $n - 1$ do
6:          $\pi(i) \leftarrow \arg\min_{j \in U} d_{Eucl}(s, a_j, \tau)$
7:          $(t_{2i-1}, t_{2i}) \leftarrow$ SLSQP$(s, a_{\pi(i)})$   ▷ Inner task
8:          $\tau \leftarrow \tau + t_{2i-1} + t_{2i}$
9:          $U \leftarrow U \setminus \{\pi(i)\}$
10:         $s \leftarrow a_{\pi(i)}$
11:     return $\pi, f(\pi)$
```

[4]GNN implementation is available from https://doi.org/10.5281/zenodo.5725837.

5.3 Combinatorial Efficient Global Optimization (CEGO)

The CEGO algorithm (Zaefferer et al., 2014b) builds upon the well-known EGO method (Jones et al., 1998) and adapts it for combinatorial optimization problems. EGO was originally designed for continuous domains and employs a Bayesian approach to iteratively learn a surrogate Gaussian process model of the objective function. In contrast, CEGO addresses combinatorial spaces by using a distance-based combinatorial surrogate model, where the classical Euclidean distance of continuous spaces is replaced with a discrete distance function which is suitable for the search space at hand.

In (Zaefferer et al., 2014a), various distance functions for the permutation space are examined. The available implementation of CEGO[5] begins by generating a few initial solutions using a max-min-distance procedure, which are then used to construct an initial surrogate model. Subsequently, a genetic algorithm utilizing operators suitable for the permutation encoding is employed to search for optimal solutions of the surrogate function. The best solution found is then evaluated using the true objective function. This evaluation allows to gain information about the objective function and to update the surrogate model maintained by CEGO. The process is repeated until a specified termination criterion is met.

Therefore, after the initial warm-up, each round of CEGO consists of: (i) optimizing the surrogate function, (ii) one true objective function evaluation, and (iii) updating the surrogate model. It is important to note that, as experimentally shown in (Santucci and Baioletti, 2022), the update of the surrogate model in the permutation space is a hard problem in itself, requiring a significant amount of computational time, especially when the number of training permutations (and their size) gets large.

For more details on the settings of the genetic algorithm used in CEGO, we refer interested readers to (Zaefferer et al., 2014a).

5.4 Unbalanced Mallows Model (UMM)

The UMM algorithm (Irurozki and López-Ibáñez, 2021) belongs to the well known family of estimation of distribution algorithms. It iteratively alternates between learning and sampling from a Mallows model, a well-known probability distribution model for permutations (Ceberio et al., 2013). At the beginning, a few solutions are randomly generated to construct the initial Mallows model. Then, in each subsequent iteration, UMM samples a permutation, evaluates it, and updates the model accordingly.

The Mallows model is characterized by a mode permutation $\pi_0 \in \mathbb{S}_n$ and a dispersion parameter $\theta \in \mathbb{R}$. The mode permutation is determined using the so-called *Unbalanced Borda* procedure, which weighs previously sampled solutions based on their fitness, ensuring that the top 10% of samples contribute 90% of the weight. The dispersion parameter θ is correlated to the expected Kendall's-τ distance $E[D]$ between a sample and the mode π_0. This expected distance is adjusted by linearly decreasing it from $\binom{n}{2}/2$ to 1 over the iterations.

For more detailed information about UMM, interested readers can refer to (Irurozki and López-Ibáñez, 2021)[6].

6 EXPERIMENTS

6.1 Experimental Settings

Experiments were conducted using the same settings as described in (López-Ibáñez et al., 2022). Ten ARP benchmark instances were generated using the instance generator and the seeds provided by the authors of (López-Ibáñez et al., 2022), allowing us to reuse the results made available by them for the competitor and baseline algorithms. Specifically, two instances were considered for each size $n \in \{10, 15, 20, 25, 30\}$ using the seeds 42 and 73. The naming scheme *n_seed* is used to denote each instance.

Two different experimetnal settings are considered as follows.

- *Black-box setting*, where the competing algorithms FAT-RLS, CEGO and UMM are randomly initialized, while RS is considered as baseline method.
- *Informed setting*, where the GNN heuristic is used both as baseline method and to produce reasonably good initial solutions for FAT-RLS, CEGO and UMM.

In both settings, all the algorithms were run 30 times per instance, with each run having a budget of 100 objective function evaluations.

It is worth noting that in (López-Ibáñez et al., 2022), two variants of both CEGO and UMM were

[5]CEGO implementation is available at https://cran.r-project.org/web/packages/CEGO.

[6]UMM implementation is available at https://zenodo.org/record/4500974.

run: one that directly considers the evolved permutation for the objective function evaluation, and another that inverts the permutation before evaluating it. However, according to Section 2, only one of these variants is actually sound. Therefore, in this work, we consider just one CEGO and one UMM: the ones named CEGO-order and UMM-rank in (López-Ibáñez et al., 2022) (that clearly outperformed their counterparts CEGO-rank and UMM-order).

Finally, the hyperparameters of the competing algorithms were set according to their original papers, i.e., (Santucci and Baioletti, 2022) for FAT-RLS and (López-Ibáñez et al., 2022) for UMM and CEGO.

6.2 Results in the black-box setting

The results collected in the black-box setting were analyzed from two different perspectives: median performances and peak performances.

For median performances, we calculated the Median Relative Percentage Deviation (MRPD) for each algorithm $\mathcal{A}$ and instance I as follows

$$MRPD_{\mathcal{A},I} = 100 \cdot \frac{median_{\mathcal{A},I} - best_I}{best_I}, \qquad (8)$$

where $median_{\mathcal{A},I}$ is the median objective value obtained by algorithm $\mathcal{A}$ over 30 executions on instance I, while $best_I$ is the best objective value obtained by any algorithm in this setting for instance I.

The MRPDs are provided in Table 1. For the competitor algorithms UMM, CEGO and RS, values are marked with ▲ if FAT-RLS significantly outperformed them, and with ▽ if FAT-RLS was significantly outperformed. No mark indicates that the performance differences were not significant. The statistical analyses were conducted using the well known Mann Whitney U test (Hollander et al., 2013), with a significance threshold of 0.05.

Table 1 shows that, though FAT-RLS is not competitive enough on the two smallest instance of size $n = 10$, it is slightly better than CEGO and significantly better than the other competitors on the two instances with $n = 15$, while, most notably, it significantly outperforms all the competitors on the larger instances with $n > 15$.

For the peak performances, Table 2 presents the objective values returned by the best execution of each algorithm for each instance. Values in bold represent, for each instance, the best objective value ever achieved by a black-box algorithm.

Table 2 shows that FAT-RLS achieved the best results in 9 out of 10 instances, while in the remaining instance (10_73) it obtained the second best peak performance after CEGO. Therefore, as also partially indicated by Table 1, where the MRPDs of FAT-RLS improve as n gets larger, it seems that FAT-RLS is capable of achieving competitive results across the entire spectrum of benchmarks, although it shows a lack of robustness when the instance size is smaller.

Table 1: Median Relative Percentage Deviations on Black-box Experiments. Algorithms whose results are marked with ▲ are significantly outperformed by FAT-RLS, while those marked with ▽ significantly outperform FAT-RLS.

Instance	FAT-RLS	UMM	CEGO	RS
10_42	12.36	13.26	9.12 ▽	21.03 ▲
10_73	16.39	11.30 ▽	5.07 ▽	16.19
15_42	13.18	17.48 ▲	14.01	25.61 ▲
15_73	10.16	15.58 ▲	12.56	24.62 ▲
20_42	9.56	20.80 ▲	15.50 ▲	26.18 ▲
20_73	14.65	30.29 ▲	23.99 ▲	33.21 ▲
25_42	14.62	29.32 ▲	24.57 ▲	34.42 ▲
25_73	8.62	26.30 ▲	18.80 ▲	28.58 ▲
30_42	6.01	27.26 ▲	18.95 ▲	28.58 ▲
30_73	7.95	24.46 ▲	18.96 ▲	27.31 ▲

Table 2: Best Objective Values on Black-box Experiments. Bolded results represent the best performance for each instance.

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

6.3 Results in the informed setting

The median and peak performance analyses conducted in Section 6.3 were also carried out in the informed scenario, where algorithms are initialized using the solution obtained through the GNN heuristic.

Tables 3 and 4 present, respectively, the median and best results achieved by FAT-RLS, UMM, CEGO, and GNN in the informed setting.

The results indicate that all three meta-heuristics improved upon the initial solution provided by GNN. However, the performance gains over the baseline algorithm are slightly less pronounced than in the black-box setting. Probably, this is because the initial solution is heuristically constructed rather than randomly generated.

In the comparison between informed FAT-RLS and informed UMM, FAT-RLS significantly outper-

Table 3: Median Relative Percentage Deviations on Informed Experiments. Algorithms whose results are marked with ▲ are significantly outperformed by FAT-RLS, while those marked with ▽ significantly outperform FAT-RLS.

Instance	FAT-RLS	UMM	CEGO	GNN
10_42	9.97	10.22 ▲	8.68 ▽	12.86 ▲
10_73	20.46	18.69 ▽	10.73 ▽	22.67 ▲
15_42	0.96	2.21 ▲	1.32 ▲	3.51 ▲
15_73	2.68	6.17 ▲	2.53	12.50 ▲
20_42	3.36	16.25 ▲	5.95	22.12 ▲
20_73	1.01	5.06 ▲	1.73	5.98 ▲
25_42	7.69	14.29 ▲	8.37 ▲	17.50 ▲
25_73	0.81	13.64 ▲	7.56 ▲	13.72 ▲
30_42	3.59	6.77 ▲	3.55	8.27 ▲
30_73	2.06	6.85 ▲	1.18 ▽	7.63 ▲

Table 4: Best Objective Values on Informed Experiments. Bolded results represent the best performance for each instance.

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

formed UMM in 9 out of 10 instances, mirroring the trend observed in the black-box setting. Conversely, the comparison between informed FAT-RLS and informed CEGO was more balanced. FAT-RLS significantly outperformed CEGO in 3 instances and was significantly outperformed by CEGO in 3 other instances. Additionally, FAT-RLS achieved the best solution in 5 instances, while CEGO achieved the best solution in 4 instances.

Finally, by comparing the results of Tables 2 and 4, it is worth noting that on the two instances of size $n = 10$, GNN and informed FAT-RLS did not match the results of RS and black-box FAT-RLS, respectively. This likely indicates that the $n = 10$ instances have a relatively "shallow landscape" where random search is sufficient to provide good results.

7 CONCLUSION AND FUTURE WORK

In this work we have experimentally studied the effectiveness of FAT-RLS, a simple trajectory-based meta-heuristic, in dealing with the Asteroid Routing Problem (ARP). FAT-RLS adopts the well known randomized local search scheme, equipped with two additional algorithmic components such as a tabu data structure and an adaptive perturbation strength mechanism.

We conducted a series of experiments on standard benchmark instances for the ARP, comparing the effectiveness of FAT-RLS with two established approaches for the same problem: CEGO, a Bayesian method for combinatorial problems, and UMM, an estimation of distribution algorithm tailored for permutation problems. The experiments were conducted under two settings: a black-box setting, where initial solution(s) for the competing algorithms were randomly generated, and an informed setting, where a purposely defined heuristic method is used to initialize the solutions of the competing meta-heuristics.

The results clearly show that FAT-RLS, despite its simpler design, outperforms both CEGO and UMM in the black-box scenario, i.e., when the ARP is blindly addressed. However, when the heuristic initialization is adopted, while FAT-RLS clearly outperforms UMM, there is no definitive evidence of superiority over CEGO.

Therefore, these results allow us to reaffirm the take away message previously exposed in (Santucci and Baioletti, 2022), i.e., simple algorithms such as FAT-RLS that are mainly based on very strong exploitation approaches, might be a viable alternative to more sophisticated techniques when dealing in low budget expensive black-box combinatorial scenarios.

Furthermore, the analyzed results, particularly those from the smaller instances, also reveal that FAT-RLS shows a lack of robustness in some cases, suggesting opportunities for enhancement in this regard. Therefore, an interesting avenue for future research involves extending the search engine of the algorithm from the randomized local search scheme to the $(1+1)$-EA scheme (Doerr et al., 2022). Essentially, the idea is to maintain to one the expected number of insertion moves per iteration, while also allowing the algorithm to perform more than one insertion per iteration. This should maintain a strong level of exploitation, while also preventing the algorithm from easily becoming trapped in a local optimum within the insertion neighborhood.

ACKNOWLEDGEMENTS

Valentino Santucci has been partially supported by the research projects: "Università per Stranieri di Perugia – Finanziamento Dipartimentale alla Ricerca per Progetti di Ricerca di Ateneo – FDR 2023", "Università per Stranieri di Perugia – Finanziamento Dipartimentale alla Ricerca per Progetti di Ricerca di Ateneo – FDR 2024".

REFERENCES

Baioletti, M., Milani, A., and Santucci, V. (2020). Variable neighborhood algebraic differential evolution: An application to the linear ordering problem with cumulative costs. *Information Sciences*, 507:37–52.

Ceberio, J., Irurozki, E., Mendiburu, A., and Lozano, J. A. (2013). A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 18(2):286–300.

Ceberio, J. and Santucci, V. (2023). Model-based gradient search for permutation problems. *ACM Transactions on Evolutionary Learning and Optimization*, 3(4):1–35.

Chopard, B. and Tomassini, M. (2018). *An introduction to metaheuristics for optimization*. Springer.

Doerr, B., Ghannane, Y., and Brahim, M. I. (2022). Towards a stronger theory for permutation-based evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1390–1398.

Eberl, M. (2016). Fisher-yates shuffle. *Arch. Formal Proofs*, 2016:19.

Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*, volume 751. John Wiley & Sons.

Irurozki, E. and López-Ibáñez, M. (2021). Unbalanced mallows models for optimizing expensive black-box permutation problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 225–233.

Izzo, D. (2015). Revisiting lambert's problem. *Celestial Mechanics and Dynamical Astronomy*, 121:1–15.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Koopmans, T. C. and Beckmann, M. J. (1955). Assignment Problems and the Location of Economic Activities. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Loiola, E. M., De Abreu, N. M. M., Boaventura-Netto, P. O., Hahn, P., and Querido, T. (2007). A survey for the quadratic assignment problem. *European journal of operational research*, 176(2):657–690.

López-Ibáñez, M., Chicano, F., and Gil-Merino, R. (2022). The asteroid routing problem: A benchmark for expensive black-box permutation optimization. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 124–140. Springer.

Nagata, Y. and Kobayashi, S. (2013). A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS Journal on Computing*, 25(2):346–363.

Neumann, F. and Wegener, I. (2007). Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40.

Rosenkrantz, D. J., Stearns, R. E., and Lewis, II, P. M. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581.

Santucci, V. and Baioletti, M. (2022). A fast randomized local search for low budget optimization in black-box permutation problems. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Santucci, V., Baioletti, M., and Milani, A. (2016). Algebraic differential evolution algorithm for the permutation flowshop scheduling problem with total flowtime criterion. *IEEE Transactions on Evolutionary Computation*, 20(5):682–694.

Santucci, V., Baioletti, M., and Milani, A. (2020). An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization. *Swarm and Evolutionary Computation*, 55:100673.

Santucci, V. and Ceberio, J. (2020). Using pairwise precedences for solving the linear ordering problem. *Applied Soft Computing*, 87.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Zaefferer, M., Stork, J., and Bartz-Beielstein, T. (2014a). Distance measures for permutations in combinatorial efficient global optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 373–383. Springer.

Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., and Bartz-Beielstein, T. (2014b). Efficient global optimization for combinatorial problems. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 871–878.