A Fast Randomized Local Search for Low Budget Optimization in Black-Box Permutation Problems

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

Marco Baioletti
University of Perugia
Perugia, Italy
marco.baioletti@unipg.it

***Abstract*—Low budget black-box optimization is a relevant topic in many practical applications with expensive objective functions or tight real-time constraints. Recently, there has been a growing interest in addressing combinatorial permutation problems in a low budget and black-box scenario. In this context, most of the previously proposed algorithms learn a probabilistic model which guides the search by trying to somehow indicate the most effective areas of the permutation search space. However, the large size and the inherent discontinuity of the permutation space may lessen the effectiveness of this approach when a low, or very low, budget of evaluations is considered. Moving from this consideration, in this work we present a simpler elitist trajectory-based algorithm for low budget black-box optimization of permutation problems. The proposed algorithm, namely FAT-RLS, is based on three core ideas: a randomized local search scheme, an adaptive perturbation strength and the use of a tabu structure. A series of experiments held on commonly adopted benchmark problems clearly shows that FAT-RLS obtains better or comparable effectiveness with respect to the previous proposals. Moreover, its negligible computational overhead is of particular interest in mission critical situations where tight real-time constraints have to be matched.**

***Index Terms*—low budget optimization, optimization under real-time constraints, black-box permutation problems, randomized local search**

I. Introduction

Many relevant real-world optimization problems are black-box, i.e., they have objective functions which are not explicitly defined in a mathematical closed form. In these cases, algorithms can only rely on the values returned by the objective function and not on its internal structure and its mathematical properties, which are mostly unknown. Another important challenge often posed by real-world scenarios is the expensive evaluation of a solution, usually in terms of time, but also in terms of space, money or other resources. For instance, this is the case of many engineering problems (see e.g. [1] and [2]), where the solutions are evaluated on the basis of time-consuming experiments, physical measurements, simulations or complex prediction models. Furthermore, in mission critical situations (see e.g. [3]), even if the evaluation is fast, very tight real-time constraints usually preclude a thorough exploration of the search space.

Therefore, expensive black-box optimization problems require to be tackled with a low budget of objective function evaluations. These requirements cut out white-box algorithms typical of the operational research field (like e.g. branch-and-bound and branch-and-cut methodologies), but also many heuristic search algorithms which are often considered in the black-box optimization literature, such as local search methods and most of the population-based metaheuristics. In fact, the budget of allowed evaluations may be too small even for a single neighborhood scan of local search algorithms, while generational population-based metaheuristics are usually penalized by the evaluation of a complete population of solutions at a time.

When the search space is formed by continuous decision variables, Bayesian optimization methodologies [4] are widely adopted because they iteratively learn a surrogate model – usually, a Gaussian process or Kriging model – of the objective function and perform a (relatively) large number of cheaper surrogate evaluations to pick up a candidate solution which is then evaluated by means of the true objective function. This mechanism allows to save lot of expensive evaluations and to adopt a variety of classical optimization approaches informed only by the surrogate model (often, by means of an acquisition function which takes into account also the uncertainty level of the model).

A notable application example of Bayesian optimization methods is in the machine learning field, where they are often used for tuning the hyperparameters of many machine learning algorithms. See e.g. [5] and [6].

However, note that the number of candidate solutions and the computational requirements for learning and updating the surrogate model may pose issues in real-time scenarios. Most importantly, the accuracy of the surrogate model drastically deteriorates for moderately large search spaces. In fact, Bayesian optimization is practically limited to no more than 20 decision variables [7, 8]. This may constitute an important limitation in a variety of real-world applications. For instance, in [9] and [2], simple evolution strategies largely outperform Bayesian optimization in estimating the 60 to 120 parameters of a hydraulic model in a low budget scenario.

In this work we are interested in low budget optimization of black-box permutation problems, i.e., problems whose solution space is formed by permutations of a given set of items. These problems arise in a variety of practical domains such as, for instance, weld sequence optimization in industry [10], optimizing experimental designs of field trials in plant breeding

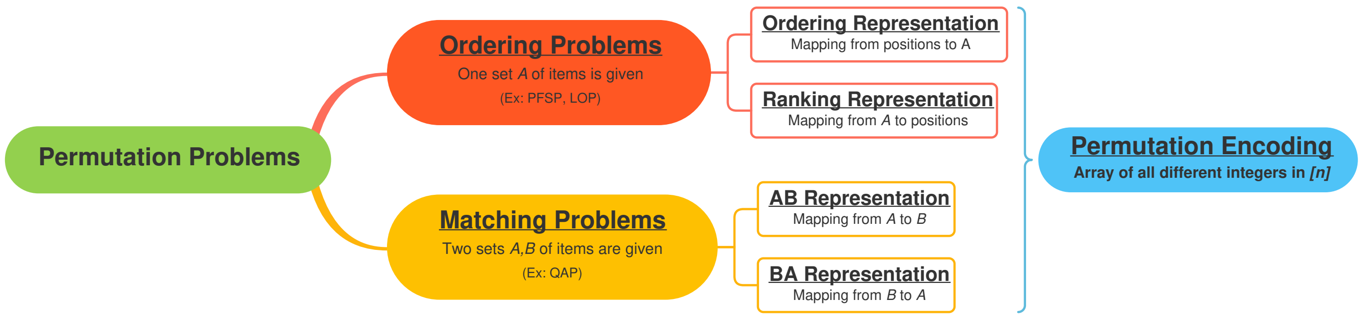


Fig. 1: Relations among semantic interpretations, representations and encoding in permutation problems.

science [11], and finding an optimal restoration plan in power grids [12] and smart grids [13].

Motivated by the large use in continuous domains, there have been attempts of adapting the surrogate-based methodologies to combinatorial search spaces [14]. For instance, [15] proposes a Bayesian optimization approach for binary problems, while a more general algorithm is the Combinatorial Efficient Global Optimization (CEGO) algorithm [16, 17] which allows to build surrogate models also in the space of permutations. However, the issues observed with continuous surrogate models are amplified in the combinatorial space because of the inherent discontinuity and ruggedness of combinatorial landscapes. Moreover, building a discrete surrogate model in CEGO requires an expensive computation which may cancel out the time saved by the cheaper evaluations. Therefore, alternative approaches have been proposed. The two most notable ones are the Ant Colony Optimization (ACO) proposed in [18] and the very recent Unbalanced Mallows Model (UMM) algorithm [19] which is based on the popular framework of estimation-of-distribution algorithms. In particular, in [19], it was shown that, with a budget of 400 evaluations, UMM has a much smaller computational overhead than CEGO – minutes vs hours – and achieves an effectiveness close to, even though worse than, that of CEGO.

Moving from these considerations, here we propose to approach low budget optimization in black-box permutation problems with a simpler elitist trajectory-based algorithm which quickly converges towards a good enough solution by employing a negligible computational overhead, thus making the proposal affordable even in mission critical problems under tight resource constraints. The proposed algorithm, namely Fast Adaptive Tabu-based Randomized Local Search (FAT-RLS), is based on three core ideas:

- a randomized local search scheme which guides the search,
- an adaptive perturbation strength which allows a fast convergence,
- a tabu structure which avoids redundant perturbations.

FAT-RLS, contrarily from CEGO and UMM, does not learn any model, is elitist and invariant to monotonic transformations of the objective function.

A comparison among FAT-RLS, CEGO and UMM is held by performing a series of experiments with a low budget of evaluations on instances of common benchmark permutation problems without violating the black-box assumption.

The rest of the paper is organized as follows. Preliminary concepts and related works are presented in Sect. II, while Sect. III describes the proposed algorithm. The conducted experimental analysis is presented and discussed in Sect. IV. Finally, conclusions are drawn in Sect. V where future lines of research are also depicted.

II. Background

A. Representing Solutions in Permutation Problems

Permutation problems all have an objective function of the form $f : \mathcal{S}_n \to \mathbb{R}$, where the domain $\mathcal{S}_n$ contains all the permutations of the set $[n] = \{1, \dots, n\}$. Hence, permutations are bijections of $[n]$ onto itself and, for this reason, any $\pi \in \mathcal{S}_n$ has its corresponding inverse $\pi^{-1} \in \mathcal{S}_n$.

Permutations are generally used to encode the solutions of two distinct classes of problems:

- *ordering problems*, where the goal is to find an optimal ordering of a given set of items (as e.g. in the permutation flowshop scheduling problem), and
- *matching problems*, where it is required to match, in the best possible way, two given equally sized sets of items (as e.g. in the quadratic assignment problem).

Both for ordering and matching problems, the fact that permutations are bijections of the first n integers has to be intended only as a genotypic encoding. Fig. 1 tries to clarify the relationships among semantic interpretations (ordering or matching), representations and genotypic encoding in permutation problems.

Let fix the semantic interpretation of the ordering problems. In an ordering problem, a set A of n items to be optimally ordered (on the basis of a provided objective function) is given. Hence, an ordering of the items in A can be represented in two distinct ways: as a mapping from positions to items (*ordering representation*), or as a mapping from items to positions (*ranking representation*). Anyway, we stress that the semantic interpretation (ordering of items) is exactly the same for both the representations (ordering or ranking). Once chosen a representation, the next step is to find a genotypic encoding. Clearly, positions are integers in $[n]$, while the items in A can be arbitrarily assigned to (all-different) identification numbers in $[n]$. Therefore, both positions and items are encoded as elements of $[n]$, thus a mapping between them can be easily encoded by a permutation in $\mathcal{S}_n$. In other words, the ordering and ranking representations share the same genotypic encoding but they still remain different representations of the same semantic interpretation. Moreover, it is possible to easily convert one representation into the other by simply inverting a permutation. Formally, if $\pi \in \mathcal{S}_n$ encodes the ordering representation of a given solution, then π^{-1} encodes the ranking representation of exactly the same solution.

This discussion should clarify why, before defining the objective function or an algorithm/operator for permutations, it is very important to specify the representation considered. Moreover, it should be now clear that if an objective function expects an ordering representation in input, it is an error to feed it with the permutation in its ranking representation. In fact, when the algorithms/operators and the objective function are specified with different representations, a conversion/inversion is always required.

Finally, note that the above argumentation, provided for the ordering problems, can be easily extended to matching problems (see Fig. 1).

B. Low Budget Algorithms for Permutation Problems

Here, we briefly describe the Combinatorial Efficient Global Optimization (CEGO) and Unbalanced Mallows Model (UMM) algorithms, which are considered later on in our experimental analysis.

CEGO. The CEGO algorithm [16] extends the well known EGO method [20] to combinatorial optimization problems. EGO is designed for continuous problems and follows the Bayesian approach by iteratively learning a Gaussian process model of the objective function. Conversely, CEGO deals with combinatorial spaces by adopting a combinatorial surrogate model similar to that of EGO, but with the Euclidean distance replaced by a suitable discrete distance function. In [17], a variety of distance functions for the permutation space are considered. The available implementation of CEGO[1] generates few initial solutions – by means of a max-min-distance procedure – and uses them to build an initial surrogate model. Then, a genetic algorithm based on permutation operators is adopted to search for good solutions of the surrogate function. The best solution is selected and evaluated with the true objective function. This evaluation allows to update the surrogate model, and the whole process is iterated till a given termination criterion is satisfied. For further information about the setting of the genetic algorithm adopted in CEGO, we refer the interested reader to [17].

UMM. The UMM algorithm [19] is an estimation-of-distribution algorithm which iteratively alternates learning and sampling of a Mallows model, a well known probability distribution model for permutations [21]. Hence, few initial solutions are randomly generated in order to build an initial Mallows model, then at each iteration UMM samples one permutation, evaluates it and updates the model. The Mallows model is parameterized by a mode permutation $\sigma_0 \in \mathcal{S}_n$ and a dispersion parameter $\theta \in \mathbb{R}$. The mode permutation is learned by means of the so-called Unbalanced Borda procedure which is run on all the previously sampled solutions that are weighted on the basis of their fitness in a such way that the best 10% of the samples have the 90% of the weight. Conversely, the dispersion parameter θ is shown to be correlated with the expected Kendall's-tau distance $\mathbb{E}[D]$ of a sample from σ_0, therefore it is set in such way that $\mathbb{E}[D]$ linearly decreases from $\binom{n}{2}/2$ to 1 with the iterations. For further information about UMM, we refer the interested reader to [19][2].

III. The Proposed Algorithm: FAT-RLS

A. Algorithmic Scheme of FAT-RLS

The Fast Adaptive Tabu-based Randomized Local Search – from now on simply referred to as FAT-RLS – is an iterative optimization heuristic which is based on three core ideas: (i) a randomized local search scheme to guide the search, (ii) an adaptive perturbation strength to allow a fast convergence towards a (hopefully) good solution, and (iii) a tabu queue to avoid redundant perturbations.

Algorithm 1 Algorithmic scheme of FAT-RLS

```
1: function FAT-RLS($d_{\mathrm{ini}}, \beta, k$)
2:     /* Initialization */
3:     $\pi \leftarrow$ a random permutation in $\mathcal{S}_n$
4:     Evaluate $f(\pi)$
5:     $nfev \leftarrow 1$                    ▷ Number of evaluations performed
6:     $TQ \leftarrow$ empty queue
7:     /* Main loop */
8:     while $nfev < budget$ do
9:         /* Compute the perturbation strength $d$ */
10:        $p \leftarrow nfev / budget$
11:        $d \leftarrow \mathrm{round}(\ 1 + s_\beta(p) \cdot (d_{\mathrm{ini}} - 1)\ )$        ▷ See Eq. (1)
12:        /* Perturb the current solution */
13:        $\sigma \leftarrow \pi$
14:        $i, j \leftarrow$ random positions s.t. $|i - j| = d$ and $\sigma_i \notin TQ$
15:        Perform the insertion $(i, j)$ on $\sigma$
16:        /* Update the current solution */
17:        Evaluate $f(\sigma)$
18:        if $f(\sigma) < f(\pi)$ then          ▷ Minimization is assumed
19:            $\pi \leftarrow \sigma$
20:        $nfev \leftarrow nfev + 1$
21:        /* Update the tabu queue */
22:        Push item $\sigma_j$ into $TQ$
23:        if $|TQ| = k$ then
24:            Pop the oldest item from $TQ$
25:     /* Return the best solution and its value */
26:     return $\pi, f(\pi)$
```

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

[2]UMM code is available at https://zenodo.org/record/4500974

The pseudo-code of FAT-RLS is provided in Alg. 1 and uses the ordering representation described in Sect. II-A, so π_i is the item at position i in the ordering represented by $\pi \in \mathcal{S}_n$.

FAT-RLS is an elitist trajectory-based algorithm which iteratively updates a single candidate solution $\pi \in \mathcal{S}_n$ in order to minimize a given objective function $f : \mathcal{S}_n \to \mathbb{R}$. The algorithm has three parameters: the initial perturbation strength d_{ini}, the adaptation parameter β, and the maximum size k of the tabu queue.

In lines 2–6 of Alg. 1, an initial permutation π is randomly generated, while the tabu queue TQ is initialized to an empty queue.

Then, in each iteration of the main loop (lines 8–24), a trial permutation σ is generated by performing a single insertion move on the current solution (line 15). The insertion (i, j) consists in shifting the item σ_i to position j in the permutation σ. In line 14, the indices i and j of the insertion are randomly selected from $[n]$ in such a way that $|i - j| = d$ and the item σ_i is not listed as tabu.

Therefore, the length of the shift is given by d which, in line 11, is iteratively decreased from d_{ini} to 1 by means of the "skewed S"-shaped function $s_\beta(p)$, defined as in Eq. (1), which takes in input the evolution percentage $p \in [0, 1]$ (see line 10) and is parametrized by $\beta \geq 1$.

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

Fig. 2 shows the behaviour of $s_\beta(p)$ for different values of β. In particular, when $\beta = 1$, we have $s_1(p) = 1 - p$, so the perturbation strength d is linearly decreased with the iterations, while the rationale for setting $\beta > 1$ is to extend both the first (explorative) and last (exploitative) parts of the evolution, thus shortening the transition from larger to smaller values of d in the middle part of the evolution.

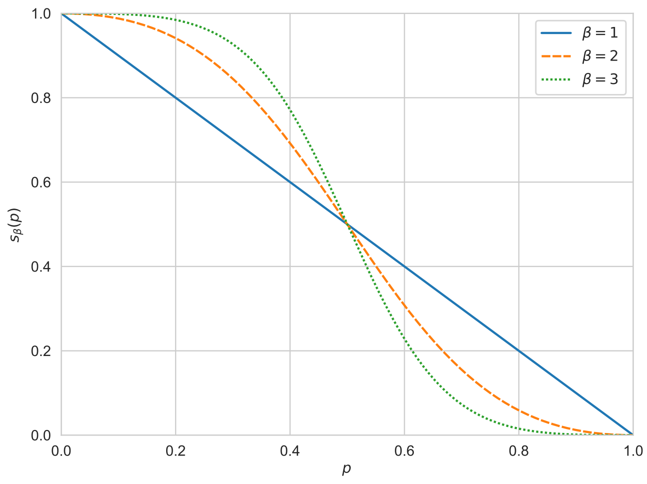


Fig. 2: The "skewed S"-shaped function $s_\beta(p)$ for $\beta = 1, 2, 3$.

The trial permutation σ replaces the current permutation π if and only if it is fitter than π (lines 17–20), while the shifted item is pushed into TQ which cannot exceed the size k (lines 21–24). Finally, the current solution – which is also the best solution ever visited – is returned in line 26.

B. Analysis of FAT-RLS

The high-level search scheme of FAT-RLS is that of the randomized local search (RLS) which has been widely analyzed for theoretical purposes (see e.g. [22]) but not too much in practical scenarios. The main idea of the RLS scheme is to select a random neighbor of the incumbent solution and accepting it only if it improves its fitness, i.e., following the elitist principle.

The neighborhood chosen for FAT-RLS is the insertion neighborhood which is widely adopted in metaheuristics for permutation problems (see e.g. [23, 24]). However, FAT-RLS does not consider all the possible neighbors but only those which can be obtained by a shift of length d and such that the shifted item is not marked as tabu.

With this regard, note that shifting an item by d positions corresponds to jump at a permutation which is at Kendall's-tau distance d from the current solution[3]. In this sense, FAT-RLS and UMM, though belonging to very different classes of algorithms, have a somehow similar search behaviour, because both iteratively perturb a permutation by jumping at a successive permutation at a given Kendall's-tau distance, which is forced to decrease with the iterations. The differences are that:

- UMM perturbs the Mallows centroid permutation, while FAT-RLS perturbs the best-so-far solution;
- UMM jumps at a given Kendall's-tau distance in expectation and isotropically, while FAT-RLS uses a restricted insertion move;
- the perturbation strength decreases linearly in UMM and following a "skewed S"-shaped function in FAT-RLS.

Furthermore, FAT-RLS, conversely from CEGO and UMM, does not learn any probabilistic model, so its computational overhead is practically negligible. In fact, excluding the objective function evaluation, the complexity of an iteration of FAT-RLS is given by the insertion operation, so it is $\Theta(d)$ and, since $d < n$, it is $O(n)$.

Another advantage of our proposal with respect to both UMM and CEGO is that FAT-RLS is invariant to monotonic transformations of the objective function. This aspect is relevant in a lot of practical scenarios. For instance, let consider objective values which are monetary quantities or physical measurements. Clearly, in real-world situations we do not want that the behaviour of the algorithm, so the quality of the solution produced, will change on the basis of the particular currency or unit of measurement adopted.

Finally, note that setting $d_{\text{ini}} \leq \lfloor n/2 \rfloor$ in FAT-RLS guarantees that there is always at least one available insertion to select in every iteration, independently of the size of the tabu queue.

[3]A shift of length d changes the relative order of the shifted item with respect to d other items.

IV. Experiments

A. Experimental Setup

Experiments were held with a twofold purpose: firstly, studying and calibrating the FAT-RLS parameters and, secondly, comparing the effectiveness and efficiency of FAT-RLS with respect to UMM and CEGO.

Instead of using real-world expensive problems, we followed the same line of [19] and the experiments were conducted with benchmark instances of classical permutation problems without violating the black-box assumption. This allowed for a faster experimentation. Furthermore, as in [19], a maximum budget of 400 evaluations and 10 executions per instance are considered.

The three selected problems are: the Linear Ordering Problem (LOP), the Permutation Flowshop Scheduling Problem (PFSP) and the Quadratic Assignment Problem (QAP). The LOP and the PFSP are ordering problems, while the QAP is a matching problem. In the following, we formulate the minimization version of the three objective functions by using the ordering representation (for the LOP and the PFSP) and the AB representation (for the QAP), as depicted in Sect. II-A.

LOP. Given a matrix $A \in \mathbb{R}^{n \times n}$, the minimization version of the LOP requires to minimize

$$\min_{\pi \in \mathcal{S}_n} f_{\mathrm{LOP}}(\pi) = \sum_{i=2}^{n} \sum_{j=1}^{i-1} A_{\pi_i, \pi_j}. \tag{2}$$

Note that the minimization form of the LOP is mathematically equivalent to the more widely adopted maximization form [25, 26, 27].

PFSP. The PFSP is defined by a matrix $P \in \mathbb{R}^{n \times m}$ of processing times and requires to minimize the makespan as follows:

$$\min_{\pi \in \mathcal{S}_n} f_{\mathrm{PFSP}}(\pi) = C_{n,m}, \tag{3}$$

where $C_{i,j} = P_{\pi_i,j} + \max\{C_{i-1,j}, C_{i,j-1}\}$ for $i \in \{2,\dots,n\}$ and $j \in \{2,\dots,m\}$, while $C_{1,j} = C_{i,1} = 0$ for $i \in [n]$ and $j \in [m]$. For further details about the PFSP, we refer the interested reader to [28].

QAP. Instances of the QAP are defined by two matrices $A, B \in \mathbb{R}^{n \times n}$ and the objective is to minimize

$$\min_{\pi \in \mathcal{S}_n} f_{\mathrm{QAP}}(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} B_{\pi_i, \pi_j}. \tag{4}$$

For further details about the QAP, we refer the interested reader to [29].

Finally, for fairer comparisons and aggregations of the results obtained in different instances, the final objective value v produced by any execution is transformed to the relative percentage deviation measure rpd as follows:

$$rpd = 100 \cdot \frac{v - best}{best}, \tag{5}$$

where $best$ is the best value observed in all the executions in the same benchmark instance. Then, the average relative percentage deviation – ARPD in short – of an algorithm on a given instance is computed as the average of the rpds obtained in the different algorithm executions on that instance.

B. Experimental Tuning of FAT-RLS Parameters

After few preliminary experiments, the following ranges of values for the three FAT-RLS parameters are considered:

- $d_{\mathrm{ini}} \in \{0.25n, 0.5n\}$;
- $\beta \in \{1, 1.2, 1.4, 1.6, 1.8, 2\}$;
- $k \in \{0, 0.25n, 0.5n, 0.75n, n\}$.

Hence, a full factorial analysis was performed by executing the 60 different settings of FAT-RLS on six benchmark instances that, in order to avoid overtuning, are different from those considered in Sect. IV-C. The selected instances are:

- N-p40-03 and N-t59f11xx for the LOP,
- rec07 and rec33 for the PFSP,
- kra30b and sko49 for the QAP.

Furthermore, any FAT-RLS setting was executed 10 times per instance with a budget of 400 evaluations.

In each instance, the ARPDs of the settings are recorded and used to rank the settings. Both the ranks and the ARPDs are averaged over all the instances and presented in Tab. I for the best five settings, ordered by average rank.

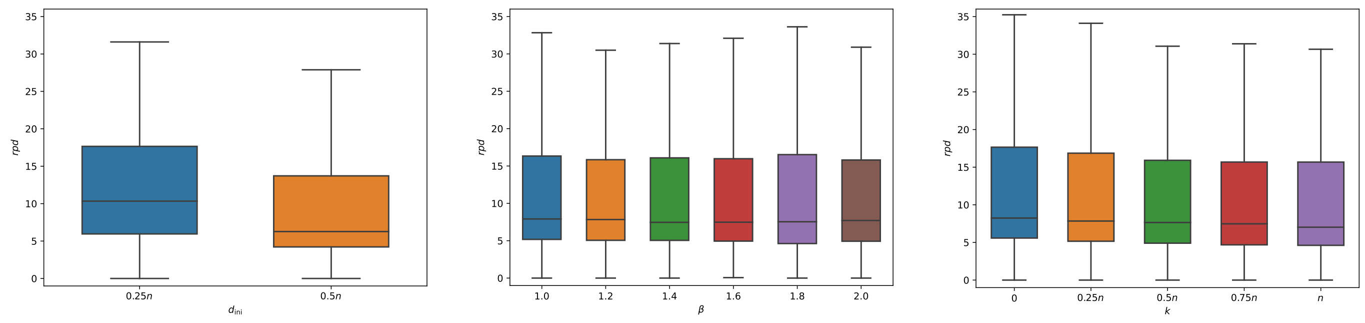
FAT-RLS Setting			**Avg Rank**	**Overall ARPD**
$\boldsymbol{d_{\mathrm{ini}}}$	$\boldsymbol{\beta}$	$\boldsymbol{k}$		
$0.5n$	1.2	n	5.00	12.37
$0.5n$	1.8	n	7.67	14.40
$0.5n$	1.4	n	8.00	11.95
$0.5n$	1.6	n	8.00	15.06
$0.5n$	2.0	n	8.00	16.09

TABLE I: Best five FAT-RLS settings by average rank.

Tab. I shows that the best setting is $d_{\mathrm{ini}} = 0.5n$, $\beta = 1.2$, $k = n$, therefore this is the setting adopted in Sect. IV-C for the experimental comparison.

A further analysis of the parameter settings is conducted. The Friedman test [30] shows that there are statistical differences among the 60 settings, while the Conover post-hoc test [30] reveals that the best 25 settings do not show statistical differences among them, but are significantly better than the others. A significance level of 0.05 is adopted for both the omnibus and the post-hoc tests.

The best five settings shown in Tab. I all have $d_{\mathrm{ini}} = 0.5n$ and $k = n$, thus giving indication that the other parameter β is less relevant for the effectiveness of the algorithm (at least, with a fixed budget of evaluations). These observations are validated also by the box-plots shown in Fig. 3, which graphically summarize the rpds obtained by varying each parameter value. Fig. 3 also shows that: (i) the most relevant parameter is the perturbation strength d_{ini}, and (ii) the worst executions have a relative deviation of around the 35% from



(a) Initial perturbation strength d_{ini} (b) Exponent β (c) Tabu size k

Fig. 3: Box-plot graphs of the tuning of the three FAT-RLS parameters.

the best ones. This last point is possibly explained by the low budget scenario considered for the experiments.

C. Experimental Comparison of FAT-RLS, UMM and CEGO

The experimental comparison of FAT-RLS with UMM and CEGO was performed on 16 benchmark instances as listed in the first two columns of Tab. II.

The eight LOP and the four PFSP instances are the same ones investigated in [19], thus in our analysis we were able to directly use the results provided in [19] (see the footnote 2 of this paper). Besides, four additional instances of the matching problem QAP were selected in order to have a more heterogeneous benchmark suite. Clearly, UMM and CEGO were run on these additional instances using the code provided in [19] and [17].

It is worthwhile to note that [19] investigates two variants of UMM and CEGO, i.e., one which passes the permutation as it is to the objective function, and another one which inverts the permutation before the evaluation. As argued in Sect. II-A, only one variant is reasonable therefore, in this work, we only consider the correct variant of both UMM and CEGO (that, clearly, is also the most effective).

Experiments were held using the tuned FAT-RLS configuration ($d_{\text{ini}} = 0.5n$, $\beta = 1.2$, $k = n$), while the parameter settings discussed in [19] were used for UMM and CEGO. Any algorithm was executed 10 times per instance and three different budgets of 100, 200 and 400 evaluations are investigated.

For the largest budget of 400 evaluations, Tab. II reports the ARPD of each algorithm in each instance and, in the last line, the overall ARPD over the whole benchmark suite. Best values are provided in bold, while the ARPDs of CEGO and UMM are marked with the symbols ▲ or ▽ when FAT-RLS obtained, respectively, significantly better or significantly worse performances; no mark is used if the differences in performance are not significant. The Mann Whitney U test [30], with a significance level of 0.05, was adopted to conduct the statistical analysis.

The results provided in Tab. II may be commented as follows:

Problem	Instance	FAT-RLS	CEGO	UMM
LOP	N-p40-01	5.33	**2.44** ▽	20.71 ▲
	N-p40-02	1.84	**1.17**	13.66 ▲
	N-p50-01	3.58	**1.61**	18.16 ▲
	N-p50-02	3.80	**2.43**	16.68 ▲
	N-sgb75.01	**3.46**	7.71 ▲	13.31 ▲
	N-sgb75.02	**4.51**	13.93 ▲	18.22 ▲
	N-t59b11xx	**4.93**	7.58	19.83 ▲
	N-t59d11xx	**11.46**	20.97 ▲	58.82 ▲
PFSP	rec05	**2.05**	2.51	2.51
	rec13	4.40	**1.89** ▽	5.20
	rec19	5.54	**4.31**	6.16
	rec31	**2.67**	4.64 ▲	4.76 ▲
QAP	kra30a	**5.12**	8.67 ▲	9.94 ▲
	sko42	**1.88**	4.06 ▲	6.69 ▲
	tai35a	2.96	**2.56**	5.98 ▲
	tho30	**4.74**	4.77	16.99 ▲
Overall ARPD		**4.27**	5.70	14.85

TABLE II: Experimental results with 400 evaluations.

- FAT-RLS has the best overall ARPD, largely better than that of UMM and slightly better than that of CEGO;
- there is no instance where UMM was able to outperform FAT-RLS and, moreover, in 13 out of 16 instances FAT-RLS significantly outperforms UMM;
- the comparison between FAT-RLS and CEGO is more balanced, though FAT-RLS is significantly better than CEGO in six instances, while it is significantly outperformed in only two instances.

In order to analyze the impact of the allowed computational budget, in Fig. 4 we show the box-plot graphs of the rpds registered by the three competitors when different numbers of evaluations are considered.

Fig. 4 clearly shows that lessening the budget makes all the three algorithms less robust. When 100 or 200 evaluations are allowed, FAT-RLS has good peak performances but CEGO looks to be more robust. Anyway, with 400 evaluations, as previously observed, FAT-RLS reaches the same level of robustness of CEGO with slightly better performances. Note

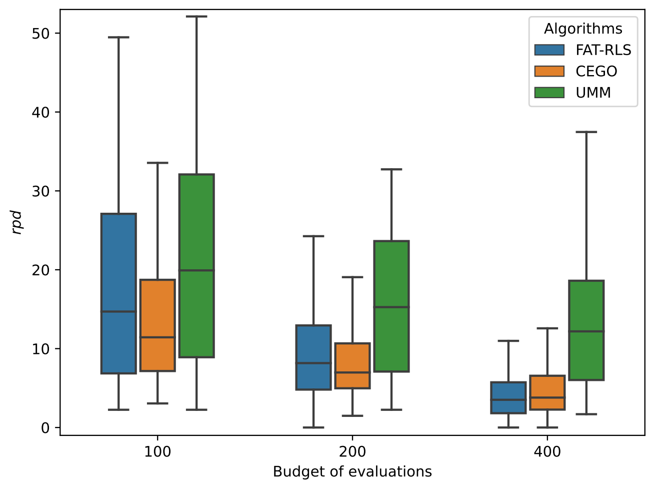


Fig. 4: Experimental results at different budgets of evaluations.

also that both FAT-RLS and CEGO clearly outperform UMM in all the scenarios considered.

Finally, it is very important to analyze the efficiency of the three algorithms. The computational times of single executions (with 400 evaluations allowed) of FAT-RLS, UMM and CEGO have three very different orders of magnitude: (around 20) hours for CEGO, minutes for UMM, and tenths of second for FAT-RLS. Therefore, though FAT-RLS and CEGO are more or less in line in terms of effectiveness, FAT-RLS has an incredibly lower computational overhead than CEGO.

V. Conclusion and Future Work

In this work we have proposed FAT-RLS, a trajectory-based algorithm for low budget optimization of black-box permutation problems. FAT-RLS is based on three core ideas: an elitist randomized local search scheme which guides the search, an adaptive perturbation strength to accelerate the convergence, and a tabu structure which avoids redundant moves.

Conversely from the previous proposals, FAT-RLS does not maintain and update any probabilistic model, thus giving it a negligible computational overhead both in terms of time and memory. With this respect, note that the computational time required by the algorithmic components of FAT-RLS is of the order of tenths of seconds, while that of CEGO is of the order of about two dozens of hours. This aspect is particularly important for mission critical scenarios with tight real-time constraints.

A series of conducted experiments reveal that the effectiveness of FAT-RLS is in line with, and often significantly better than, the effectiveness of the competitor algorithms.

Moreover, FAT-RLS, conversely from CEGO and UMM, is invariant to monotonic transformations of the objective function: a useful property in real-world situations where different unit of measurements for the same observation may be adopted.

Summarizing, an important take away message of this work is that, in low budget black-box combinatorial problems, simple algorithms like FAT-RLS might be a viable alternative to more sophisticated techniques based on probabilistic models that, until now, were the most considered approaches in the analyzed scenario. Hence, the results presented show that, in the context of low budget combinatorial optimization, there is still room for improvement.

As future line of research, it is interesting to further extend the experimental analysis of FAT-RLS and its competitors, possibly considering not only benchmark problems but also expensive and/or mission critical real-world applications.

Furthermore, it should be noted that, by design, FAT-RLS cannot escape local optima of the insertion neighborhood. Although, considering the results, this does not look to be a great issue, it would be an interesting future line of research to introduce an algorithmic mechanism that, at least theoretically, guarantees to escape basins of attraction.

Finally, for the sake of reproducibility, source code, full experimental results and the scripts used for the analysis were made available at (doi: 10.5281/zenodo.6567002).

Acknowledgment

This work was partially supported by the research grants: (i) “Università per Stranieri di Perugia – Progetto di ricerca *Artificial Intelligence for Education, Social and Human Sciences*”, and (ii) “Università per Stranieri di Perugia – *Finanziamento per Progetti di Ricerca di Ateneo — PRA 2022*”.

References

[1] Y. Zhang, D. W. Apley, and W. Chen, “Bayesian optimization for materials design with mixed quantitative and qualitative variables,” *Scientific reports*, vol. 10, no. 1, pp. 1–13, 2020.

[2] A. Agresta, M. Baioletti, C. Biscarini, F. Caraffini, A. Milani, and V. Santucci, “Using optimisation meta-heuristics for the roughness estimation problem in river flow analysis,” *Applied Sciences*, vol. 11, no. 22, 2021.

[3] D.-Y. Kim and J.-W. Lee, “Joint mission assignment and location management for uavs in mission-critical flying ad hoc networks,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 323–328.

[4] P. I. Frazier, “A tutorial on bayesian optimization,” *arXiv preprint arXiv:1807.02811*, 2018.

[5] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.

[6] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, “Hyperparameter optimization for machine learning models based on bayesian optimization,” *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[7] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, “Scalable global optimization via local bayesian optimization,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 5496–5507, 2019.

[8] R. Moriconi, M. P. Deisenroth, and K. S. Kumar, “High-dimensional bayesian optimization using low-dimensional feature spaces,” *Machine Learning*, vol. 109, no. 9, pp. 1925–1943, 2020.

[9] A. Agresta, M. Baioletti, C. Biscarini, A. Milani, and V. Santucci, “Evolutionary algorithms for roughness coefficient estimation in river flow analyses,” in *Proceedings of the 25th International Conference on the Applications of Evolutionary Computation (EvoApps 2021)*. Cham: Springer International Publishing, 2021, pp. 795–811.

[10] I. Voutchkov, A. Keane, A. Bhaskar, and T. M. Olsen, "Weld sequence optimization: The use of surrogate models for solving sequential combinatorial problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 30, pp. 3535–3551, 2005.

[11] V. Feoktistov, S. Pietravalle, and N. Heslot, "Optimal experimental design of field trials using differential evolution," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1690–1696.

[12] P. Kaufmann and C. Shen, "Generator start-up sequences optimization for network restoration using genetic algorithm and simulated annealing," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015, pp. 409–416.

[13] J. Hou, Z. Xu, Z. Y. Dong, and K. P. Wong, "Permutation-based power system restoration in smart grid considering load prioritization," *Electric Power Components and Systems*, vol. 42, no. 3-4, pp. 361–371, 2014.

[14] A. Moraglio and A. Kattan, "Geometric generalisation of surrogate model based optimisation to combinatorial spaces," in *European conference on evolutionary computation in combinatorial optimization*. Springer, 2011, pp. 142–154.

[15] R. Baptista and M. Poloczek, "Bayesian optimization of combinatorial structures," in *International Conference on Machine Learning*. PMLR, 2018, pp. 462–471.

[16] M. Zaefferer, J. Stork, M. Friese, A. Fischbach, B. Naujoks, and T. Bartz-Beielstein, "Efficient global optimization for combinatorial problems," in *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, 2014, pp. 871–878.

[17] M. Zaefferer, J. Stork, and T. Bartz-Beielstein, "Distance measures for permutations in combinatorial efficient global optimization," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2014, pp. 373–383.

[18] L. P. Cáceres, M. López-Ibáñez, and T. Stützle, "Ant colony optimization on a limited budget of evaluations," *Swarm Intelligence*, vol. 9, no. 2, pp. 103–124, 2015.

[19] E. Irurozki and M. López-Ibáñez, "Unbalanced mallows models for optimizing expensive black-box permutation problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 225–233.

[20] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[21] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 286–300, 2013.

[22] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," *Theoretical Computer Science*, vol. 378, no. 1, pp. 32–40, 2007.

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

[24] M. Baioletti, A. Milani, V. Santucci, and U. Bartoccini, "An experimental comparison of algebraic differential evolution using different generating sets," in *Proc. of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19, 2019, p. 1527–1534.

[25] V. Santucci, M. Baioletti, and A. Milani, "An algebraic framework for swarm and evolutionary algorithms in combinatorial optimization," *Swarm and Evolutionary Computation*, vol. 55, p. 100673, 2020.

[26] V. Santucci and J. Ceberio, "Using pairwise precedences for solving the linear ordering problem," *Applied Soft Computing*, vol. 87, 2020.

[27] M. Baioletti, A. Milani, and V. Santucci, "A new precedence-based ant colony optimization for permutation problems," in *Proc. of 2017 Asia-Pacific Conf. on Simulated Evolution and Learning (SEAL 2017)*. Cham: Springer, 2017, pp. 960–971.

[28] V. Santucci, M. Baioletti, and A. Milani, "Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm," *AI Communications*, vol. 29, no. 2, pp. 269–286, 2016.

[29] A. Silva, L. C. Coelho, and M. Darvish, "Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search," *European Journal of Operational Research*, vol. 292, no. 3, pp. 1066–1084, 2021.

[30] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013, vol. 751.