Gradient Search in the Space of Permutations

An application for the Linear Ordering Problem

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

Josu Ceberio
University of the Basque Country
josu.ceberio@ehu.eus

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

ABSTRACT

Gradient search is a classical technique for optimizing differentiable functions that has gained much relevance recently due to its application on Neural Network training. Despite its popularity, the application of gradient search has been limited to the continuous optimization and its usage in the combinatorial case is confined to a few works, all which tackle the binary search space. In this paper, we present a new approach for applying Gradient Search to the space of permutations. The idea consists of optimizing the expected objective value of a random variable defined over permutations. Such a random variable is distributed according to the Plackett-Luce model, and a gradient search over its continuous parameters is performed. Conducted experiments on a benchmark of the linear ordering problem confirm that the Gradient Search performs better than its counterpart Estimation of Distribution Algorithm: the Plackett-Luce EDA. Moreover, results reveal that the scalability of the Gradient Search is better than that of the PL-EDA.

CCS CONCEPTS

• **Computing methodologies → Search methodologies**; **Discrete space search**; **Randomized search**.

KEYWORDS

Gradient Search, Permutations, Symmetric Group, Optimization

ACM Reference Format:
Valentino Santucci, Josu Ceberio, and Marco Baioletti. 2020. Gradient Search in the Space of Permutations: An application for the Linear Ordering Problem. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Mathematical gradient-based optimization algorithms, hereinafter referred to as Gradient Search (GS) techniques, are classical technique for optimizing problems defined on the continuous domain. Recently, GS approaches gained popularity in the machine learning field where they are adopted for Neural Network training [11, 28]. Nevertheless, in the combinatorial domain, the application of GS has been limited to a few works. Clearly, the main reason is that, in a discrete space, there does not exist a well-defined notion of gradient.

Nevertheless, there have been proposals [5, 25, 34] to apply a model-based approach that can enable the use of GS in the combinatorial case. The main idea consists of: (i) optimizing the expected objective value of a random variable defined over the discrete space, and (ii) defining the model underlying the random variable in terms of continuous parameters. In this way, a gradient can be defined and computed over the parameters of the model.

Though this approach can be generally applied to practically any discrete search space, only applications to binary problems have been observed in the literature [4, 22, 25]. Other discrete search space such as, for instance, the space of permutations, have been mostly ignored so far in the research.

In this paper, we propose using Gradient Search for optimizing problems defined in the space of permutations. These problems, usually called permutation-based combinatorial problems, are optimization problems whose solutions are naturally described by permutations of n items [8].

To that end, among the broad range of probability modes defined on the symmetric group (space of permutations), due to its simplicity and interpretability, we focused on the Plackett-Luce model [20, 26]. Taking this model as a basis, we devised a Gradient Search scheme – namely, Plackett-Luce-GS (PL-GS) – which can be applied to any permutation problem.

For the sake of validating the idea proposed in this manuscript, a set of experiments was conducted on a benchmark of instances of the Linear Ordering Problem (LOP) where comparisons with the Plackett-Luce Estimation of Distribution Algorithm (PL-EDA) [9] were carried out. It is worth noting that in both approaches, PL-GS and PL-EDA, the solutions of the LOP, codified as permutations, are sampled from a previously adjusted Placket-Luce model. So far the similarities, in PL-EDA the maximum likelihood estimators of the parameters, are estimated, and the model is employed in the context of an EDA. However, in the newly proposed PL-GS, the weights are adjusted by means of a gradient ascent process that aims at maximizing the expected objective value over the sample space of the PL model. Hence, the optimization of the given objective function happens as a *side effect* of the previously described process. The experimental results revealed that, appropriately calibrated, PL-GS clearly outperforms PL-EDA in terms of quality measures, and also execution time.

The remainder of the paper is organized as follows. In the next section, an introductory background of gradient search for combinatorial optimization is provided. Afterwards, in Section 3, a revision of probability distributions defined on the space of permutations $\mathbb{S}_n$ is introduced. The contribution of this paper, the Gradient Search

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

for permutation problems, is presented in Section 4. Then, an experimental study that proves the validity of the proposed idea is conducted in Section 5. Finally, conclusions and future lines of research are discussed in Section 6.

2 GRADIENT SEARCH FOR COMBINATORIAL OPTIMIZATION

Gradient Search (GS) techniques are nowadays ubiquitous in the field of machine learning [11, 28], where they are usually adopted in order to optimize the loss function of a given probabilistic model on a given dataset of samples. Actually, the loss function is a differentiable function which maps a set of continuous variables – the parameters of the model – onto a real number which intuitively indicates the cost of the parameterized model. Therefore, GS techniques exploit the gradient of a differentiable objective function in order to iteratively adjust the parameters by moving them towards the direction of steepest ascent (descent) of the objective function to maximize (minimize).

Formally, let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a differentiable objective function to maximize, then the most basic GS technique – gradient ascent – iteratively updates the incumbent solution $x_t \in \mathbb{R}^n$ according to

$$x_{t+1} \longleftarrow x_t + \eta \nabla f(x_t), \tag{1}$$

where $\nabla f(x_t) \in \mathbb{R}^n$ is the gradient of f at point x_t, while $\eta > 0$ is the learning rate parameter of GS.

Though the GS approach is very natural and effective for a differentiable function, it is not apparent how to use it in combinatorial optimization. Very first attempts of using GS for optimizing discrete objective function have been proposed in [4–6, 34]. The key concept is to use GS in order to optimize the expected objective value of a random variable whose probability model, over the discrete space of solutions, is represented by continuous parameters. This idea has been later reused, systematized and extended in [22], [32] and [25]. Note anyway that, in all these works, only discrete problems over bit-strings have been considered. The only work which considers GS techniques over a discrete space different from the binary one is the very recent [1], which addresses the problem of learning the structure of a Bayesian network given a dataset of observations. It is worth noting that [1] is contemporary to our work here presented.

In order to be self-contained, we formally provide the main scheme of GS when applied to a combinatorial problem defined over a general discrete solutions space X and whose objective function to maximize is $f : X \longrightarrow \mathbb{R}$.

Let us choose a probability model over X, represented by a vector $\theta \in \mathbb{R}^m$, and such that its probability mass function p_θ is differentiable with respect to θ.[1]

Therefore, the objective function to be maximized by GS is

$$F(\theta) = E_\theta[f(x)] = \sum_{x \in X} f(x) p_\theta(x), \tag{2}$$

whose gradient, with respect to θ, can be written as

$$\begin{aligned} \nabla_\theta F(\theta) &= \nabla_\theta \textstyle\sum_{x \in X} f(x) p_\theta(x) \\ &= \textstyle\sum_{x \in X} f(x) \nabla_\theta p_\theta(x) \\ &= \textstyle\sum_{x \in X} f(x) \nabla_\theta p_\theta(x) \frac{p_\theta(x)}{p_\theta(x)} \\ &= \textstyle\sum_{x \in X} f(x) \left[\nabla_\theta \log p_\theta(x)\right] p_\theta(x) \\ &= E_\theta \left[f(x) \nabla_\theta \log p_\theta(x)\right], \end{aligned} \tag{3}$$

where the fourth line is due to the so-called "log derivative trick".

Since X has usually a prohibitive size, we resort to computing an estimate of the search gradient based on λ samples $x_1, \ldots, x_\lambda \in X$ drawn from the probability model represented by θ, i.e.,

$$\nabla_\theta F(\theta) \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} f(x_i) \nabla_\theta \log p_\theta(x_i). \tag{4}$$

Therefore, the GS scheme of formula (1) can now be used to iteratively update θ by summing to it the quantity $\eta \nabla_\theta F(\theta)$. As proved in [4], the optimization of $f(x)$ is obtained, in some sense, as a side effect of the optimization of $F(\theta)$. In fact, $F(\theta)$ is optimized when the probability mass p_θ is totally concentrated on the optimum solution of $f(x)$.

For the sake of clarity, we provide the pseudo-code of GS for combinatorial optimization in Algorithm 1.

Algorithm 1 Gradient Search for Combinatorial Optimization

```
 1: function GS($f : X \longrightarrow \mathbb{R}$, $\eta \in \mathbb{R}^+$, $\lambda \in \mathbb{N}^+$)
 2:     initialize $\theta_0$ in such a way that probabilities are uniformly distributed
 3:     $x^*$ maintains the best solution so far
 4:     $t \leftarrow 0$
 5:     while stopping criterion is not met do
 6:         for $i \leftarrow 1$ to $\lambda$ do
 7:             sample $x_i$ from $\theta_t$
 8:             evaluate $f(x_i)$ and update $x^*$ if improvement found
 9:             calculate $\nabla_{\theta_t} \log p_{\theta_t}(x_i)$
10:         end for
11:         calculate $\nabla_{\theta_t} F(\theta_t)$ according to equation (4)
12:         $\theta_{t+1} \leftarrow \theta_t + \eta \nabla F(\theta_t)$
13:         $t \leftarrow t + 1$
14:     end while
15:     return $x^*$
16: end function
```

Hence, an implementation for a concrete discrete space requires a sampling method (line 7) and a procedure to compute the derivatives of the log-probability (line 9).

3 PROBABILITY DISTRIBUTIONS ON $\mathbb{S}_n$

Modelling ranking data by means of probability distributions has been a hot research topic for decades in the applied mathematics and statistics field. In fact, a large number of papers have been published on the topic of probability distributions on the symmetric group $\mathbb{S}_n$. In the following, a quick summary of the most relevant probability distributions for permutation spaces will be provided. It is not the aim of this section, to provide a comprehensive review of such models, so we refer the interested reader to the work by Critchlow et al. [10, 13] and the tutorial work by Lozano and Irurozki [19].

[1] Clearly, since it is a probability mass function, we have that $\sum_{x \in X} p_\theta(x) = 1$ and $p_\theta(x) \geq 0$ for all $x \in X$.

Before starting, mathematically a permutation is a bijection function σ of the set $\{1, \ldots, n\}$ onto itself. $\sigma(i)$ (also denoted as σ_i) stands for the element at position i, and $\sigma^{-1}(i)$ represents the position of element i. The group of all permutations of size n, is denoted as $\mathbb{S}_n$, and is also known as the *symmetric group*.

According to Fligner et al. [14], the probability distributions for permutations are usually grouped into two main families: (1) the models that assign probabilities using a distance metric on permutations, known as *distance-based ranking models*, and those that use the item-effect approach, known as *order statistic models*.

Among the first family of models, in [23] a distance-based exponential ranking model, known as the Mallows model was introduced. Analogous to the Gaussian distribution over the continuous domain, the Mallows model is defined by two parameters, the central permutation σ_0 and the concentration parameter θ, and describes the variability of the permutations around the true permutation σ_0. The probability of every permutation in $\mathbb{S}_n$ decays exponentially with its distance, $D(\sigma, \sigma_0)$ to σ_0

$$P(\sigma) = \frac{\exp(-\theta D(\sigma, \sigma_0))}{\psi(\theta)} \tag{5}$$

where ψ is a normalization function that depends on the concentration parameter θ. Despite the initial papers on the Mallows model investigated distance metrics that count the number of discordant pairs among the permutations [15], recent works have extended the applicability of this model using four different metrics for permutations-coded solutions [12, 17].

As a generalization of the Mallows model, Fligner et al. [14] proposed a multi-stage probability model that requires the distance to be decomposed into $n-1$ terms. Then, the Generalized Mallows model makes use of $n-1$ concentration parameters, the vector $\boldsymbol{\theta}$, each of them affecting a particular component of the distance decomposition. The probability of any $\sigma \in \mathbb{S}_n$ is calculated as

$$P(\sigma) = \frac{\exp(-\sum_{j=1}^{n-1} \theta_j S_j(\sigma, \sigma_0))}{\psi(\boldsymbol{\theta})} \tag{6}$$

where S_j denotes the jth term of the decomposed distance. According to the work by Irurozki [17], most, although not all, of the distance-metrics on permutations, i.e., Kendall's-τ, Cayley, Hamming, can be decomposed as the sum of $n-1$ terms, and can be efficiently learnt and sampled.

Regarding the second group of models, *order statistic models*, there are numerous references such as Thurstone [30], Mosteller [24] and Babington-Smith[18]. However, one of the most well-known probability models in this group is the Plackett-Luce model (a Thurstone type model), and its special case, by paired comparisons, the Bradley-Terry model [27]. In this paper, the Plackett-Luce model is fundamental for the Gradient Search approach we propose, and so, in the following paragraphs, we provide extensive details on it.

Given a vector $\mathbf{w} = (w_1, \ldots, w_n)$ of positive weights, under the Bradley-Terry model, the probability of item i beating item j is calculated as

$$P(i \prec j) = \frac{w_i}{w_i + w_j}$$

where w_i and w_j denote the weights associated to the items i and j. The idea of "beating" is translated to the permutation as the preference for $\sigma(i) < \sigma(j)$, i.e., better rank.

Therefore, the probability of any permutation $\sigma \in \mathbb{S}_n$ (up to a normalization constant) is obtained by extending the previous expression to all pairwise comparisons as

$$P_{BT}(\sigma|\mathbf{w}) \propto \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \frac{w_{\sigma(i)}}{w_{\sigma(i)} + w_{\sigma(j)}}. \tag{7}$$

The Plackett-Luce model [20, 26] extends the comparison that the Bradley-Terry model makes to any number of items. For each item $i \in B$, where B ranges over all the possible subsets of items $\{1, \ldots, n\}$, $P_B(i)$ is the probability that item i is chosen as the most preferred item among those listed in B. Formally,

$$P_B(i) = \frac{w_i}{\sum_{j \in B} w_j}$$

Then, for every permutation σ and a parameter vector $\mathbf{w}$, the probability under the Plackett-Luce model is calculated as

$$P(\sigma|\mathbf{w}) = \prod_{i=1}^{n-1} \frac{w_{\sigma(i)}}{\sum_{j=i}^{n} w_{\sigma(j)}} \tag{8}$$

Thus, the higher w_i is, the more likely i is to appear in the top rank. By intuition, the vector $\mathbf{w}$ can be normalized to sum 1, so that w_i becomes the probability that item i is most preferred among the full set of items.

The Plackett-Luce model can be explained under numerous illustrative examples, and one is that proposed by Diaconis in the foreground of [13]. Let us consider we have n folders that are used with different frequencies. We want to arrange them so the most popular folder is on top, the next most popular folder next, and so on. If we do not know the popularity of the folders, it is natural to rearrange them by leaving the last used folder on top each time. If the frequency of use of folder i is w_i, this gives a Markov chain on the symmetric group whose stationary probability is given by the Plackett-Luce model.

4 GRADIENT SEARCH IN THE SPACE OF PERMUTATIONS

We propose an implementation, for the search space of permutations $\mathbb{S}_n$, of the discrete GS scheme presented in Section 2.

First of all, we choose the Plackett-Luce (PL) model (see Section 3) because, compared to the other models, PL has the following advantages: (i) the parameters are continuous, (ii) the probability mass function (and its logarithm) is differentiable, and (iii) the sampling procedure can be easily implemented in an unbiased way. Note that models such as those based on distances (see Eqs. 5 and 6) do not comply with these restrictions and, thus, cannot be used in the framework proposed in this work. In the following, we will refer to our discrete GS variant for permutations as PL-GS.

Let us note that the PL weights $w \in \mathbb{R}^n$ must be positive for equation (8) and the sampling procedure to be well defined. Moreover, let us also consider that, according to Section 2, we are planning to iteratively update w by summing to it a (scaled) gradient vector but, since the partial derivatives in the gradient are free to assume any value in $\mathbb{R}$, we cannot guarantee the feasibility of w after an update, i.e., one or more weights may become negative.

We address this issue by means of the exp function which has the property of being a strictly monotonic transformation that maps a generic real number onto a positive real number. Therefore, PL is reparametrized by the unconstrained vector $\tilde{w} \in \mathbb{R}^n$ in such a way that, given any permutation $\sigma \in \mathbb{S}_n$, its probability to be sampled is

$$P(\sigma|\tilde{\mathbf{w}}) = \prod_{i=1}^{n-1} \frac{\exp\left(\tilde{w}_{\sigma(i)}\right)}{\sum_{j=i}^{n} \exp\left(\tilde{w}_{\sigma(j)}\right)}. \tag{9}$$

Note anyway that the reparametrized variant is equivalent to the "standard" PL model described in Section 3. In fact, instead of directly maintaining the weight values, we maintain their logarithm, i.e., the following relations hold: $w = \exp \tilde{w}$ and $\tilde{w} = \log w$.

Therefore, by following the general case equations (2) and (4), given a permutation problem with objective function $f : \mathbb{S}_n \to \mathbb{R}$, PL-GS aims to optimize

$$F(\tilde{w}) = E_{\tilde{w}}[f(\sigma)] = \sum_{\sigma \in \mathbb{S}_n} f(\sigma) P(\sigma|\tilde{w}) \tag{10}$$

by iteratively updating the parameters $\tilde{w}$ using an estimate of $\nabla_{\tilde{w}} F(\tilde{w})$ obtained from λ sampled permutations $\sigma_1, \ldots, \sigma_\lambda$, i.e.,

$$\nabla_{\tilde{w}} F(\tilde{w}) \approx \frac{1}{\lambda} \sum_{i=1}^{\lambda} U(f(\sigma_i)) \nabla_{\tilde{w}} \log P(\sigma_i|\tilde{w}). \tag{11}$$

Note that, in equation (11), with respect to equation (4), we transformed the objective value $f(\sigma)$ by the (not necessarily strictly) monotonic transformation U. This has been already done in similar contexts [25, 32] in order to improve the effectiveness of the search process. Note anyway that, by setting U to the identity transformation, the original scheme of equation (4) is recovered. In section 4.1 we will refer to U as *utility function* and we discuss some possible choices.

In order to implement PL-GS a formula is required for computing the gradient of the log-probability (see also line 9 of Algorithm 1). Given any $\sigma \in \mathbb{S}_n$ and $\tilde{w} \in \mathbb{R}^n$, then, by using some calculus, it is possible to derive a formula for $\frac{\partial \log P(\sigma|\tilde{w})}{\partial \tilde{w}_{\sigma(i)}}$, i.e., the $\sigma(i)$–th entry of $\nabla_{\tilde{w}} \log P(\sigma|\tilde{w})$, as follows:

$$\frac{\partial \log P(\sigma|\tilde{w})}{\partial \tilde{w}_{\sigma(i)}} = 1 - \exp\left(\tilde{w}_{\sigma(i)}\right) \sum_{k=1}^{i} \frac{1}{\sum_{j=i}^{n} \exp\left(\tilde{w}_{\sigma(j)}\right)}. \tag{12}$$

Interestingly, with simple bookkeeping and by following the order induced by σ, it is possible to calculate all the n entries of $\nabla_{\tilde{w}} \log P(\sigma|\tilde{w})$ in $\Theta(n)$ time steps.

We now have the ingredients to build PL-GS, whose pseudocode is provided in Algorithm 2. Note that, in line 3, all the PL weights are initialized to the same value. With such a configuration, the PL model is equivalent to a uniform distribution over $\mathbb{S}_n$, thus PL-GS has no initialization bias. The rest of the algorithm follows the general scheme presented in Algorithm 1 and introduces an *utility function* (see lines 1–2 and equation (4)) and a *soft restart* mechanism when numerical accuracy degrades (lines 14–15). These additional aspects are described in Sections 4.1 and 4.2, respectively.

4.1 Utility functions

Given the samples $\sigma_1, \ldots, \sigma_\lambda$ drawn in a PL-GS iteration, three utility functions are considered as follows.

Algorithm 2 Plackett-Luce Gradient Search

```
 1: function PL-GS($f : \mathbb{S}_n \to \mathbb{R}$, $\eta \in \mathbb{R}^+$, $\lambda \in \mathbb{N}^+$, $U : \mathbb{R} \to \mathbb{R}$)
 2:     require: $U$ has to be monotonic
 3:     $\tilde{w}_t \leftarrow (0, \ldots, 0)$                ▷ Uniform distribution
 4:     $\sigma^*$ maintains the best permutation so far
 5:     $t \leftarrow 0$
 6:     while stopping criterion is not met do
 7:         for $i \leftarrow 1$ to $\lambda$ do
 8:             sample $\sigma_i$ from $\tilde{w}$
 9:             evaluate $f(\sigma_i)$ and update $\sigma^*$ if improvement found
10:             calculate $\nabla_{\tilde{w}_t} \log P(\sigma_i|\tilde{w}_t)$ according to eq. (12)
11:         end for
12:         calculate $\nabla_{\tilde{w}_t} F(\tilde{w}_t)$ according to equation (11)
13:         $\tilde{w}_{t+1} \leftarrow \tilde{w}_t + \eta \nabla F(\tilde{w}_t)$
14:         if numerical problems occurred then
15:             $\tilde{w}_{t+1} \leftarrow$ almost degenerate distr. with mode $\sigma^*$
16:         end if
17:         $t \leftarrow t + 1$
18:     end while
19:     return $\sigma^*$
20: end function
```

Fitness. U is the identity function, i.e., $U(f(\sigma_i)) = f(\sigma_i)$. Therefore, the fitness values are directly used as they are in the gradient computation as usual.

Normalized Fitness. $U(f(\sigma_i)) = f(\sigma_i)/\sum_{j=1}^{\lambda} f(\sigma_j)$, therefore the utilities sum up to 1 and are proportional to the fitness values.

Super Linear. Sort the samples from the best to the worst in terms of fitness and set $\mu = \lambda/2$. Assign null utility to the μ worst samples, while, for the remaining ones, temporarily assign to the i–th best sample $\exp(i)$ points of utility and, finally, normalize the utilities of the best μ samples. This utility function makes PL-GS invariant for monotonic transformations of the objective function and it is inspired by weights used in the CMA-ES algorithm [16].

4.2 Soft restart

As with other discrete GS schemes [4], with the passing of iterations, PL-GS will tend to converge towards a *degenerate configuration* of the probability model, i.e., a configuration of $\tilde{w}$ such that almost all the probability mass is concentrated on a single permutation.

However, the PL model has inherent numerical problems. For the sake of clarity, we use the classical parametrization w of PL as described in Section 3. Given any number $r > 1$ and $\sigma \in \mathbb{S}_n$, by setting w in such a way that $\frac{w_{\sigma(i)}}{w_{\sigma(i+1)}} \geq r$ for $i = 1, \ldots, n-1$, we have that

$$P(\sigma|w) \geq 1 - (n-1)r^{-1}. \tag{13}$$

Therefore, it is possible that almost all the probability is assigned to a single permutation when r reaches a much larger value than n, e.g., 10^6 or 10^9. Clearly, though theoretically possible, numerical problems arise when this happens. In fact, overflow and underflow problems were observed in the gradient calculation of equations (11) and (12).

To address these numerical issues, we devised a simple mechanism: at every iteration of PL-GS we verify if a computation resulted in an inf or nan value and, if so, the PL weights $\tilde{w}$ are reset to a configuration whose mode is the best permutation so far σ^*.

The reset is carried out by equally spacing the weights of $\tilde{w}$ in the interval $[lb, ub]$ following the reverse order induced by σ^*, i.e., $\tilde{w}_{\sigma(1)} = ub$ and $\tilde{w}_{\sigma(i+1)} = \tilde{w}_{\sigma(i)} - \delta$ with $\delta = (ub - lb)/(n - 1)$. In this way, the probability mass of σ^* is like in equation (13) with $r = \exp \delta$. After some experimentation, lb and ub were set to, respectively, -10 and 10, independently of n. This setting did not result in any numerical problem in our experimentation and, though not guaranteeing that only σ^* can be sampled, σ^* is sampled most of the times and, when not sampled, the drawn permutation is very similar to σ^*.

5 EXPERIMENTAL STUDY

In order to validate the idea presented in this paper, we conducted a set of experiments to (1) analyze the impact of the different utility functions introduced for PL-GS, (2) study the variability on the performance of the gradient search subject to η and λ parameters, and (3) compare the behavior of our proposal to the Plackett-Luce Estimation of Distribution Algorithm (PL-EDA) [9].

Conducted experiments have been carried on the LOLIB benchmark instances of the Linear Ordering Problem (LOP) [29]. In the LOP, we are given a square matrix $\mathbf{B} = [b_{ij}]_{n \times n}$ of parameters and the goal is to find the simultaneous permutation $\sigma \in \mathbb{S}_n$ of rows and columns in $\mathbf{B}$ such that the sum of the entries above the main diagonal is maximized. Then, the objective function is denoted as

$$f(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{\sigma(i)\sigma(j)}.$$

5.1 Impact of the utility functions

Based on the optimization framework we proposed, it can be expected that the utility function should have a meaningful impact on the convergence of the probability model across the optimization process. In this sense, in the following experiment the three utility functions "Fitness", "Normalized Fitness" and "Super Linear" explained in the previous section were considered for the analysis. Particularly, in order to see the level of convergence of the probability model across the optimization, we calculated the entropy[2] of the vector of weights $\mathbf{w}$ as

$$H(\mathbf{w}) = -\sum_{i=1}^{n} p_i \log p_i,$$

where $p_i = \frac{w_i}{\sum_{j=1}^{n} w_j}$. The parameter η was set constant to 0.01, and three different λ values $\{100, 1\,000, 10\,000\}$ were considered. Executions were stopped after 10^5 iterations. For the executions, *N-be75eec* instance from LOLIB was arbitrarily chosen.

Averaged results of 10 repetitions of each parameter combination are depicted in Fig. 1. In view of the plots in Fig. 1, as the optimization progresses, the entropy decreases. This suggests that the probability mass becomes more concentrated in certain solutions of the search space. However, as expected, the utility function has a relevant impact on the evolution of the probability model across the iterations. Despite with the three utility functions, the entropy is reduced, for the "Fitness" and "Normalized Fitness" functions, the descent is gradual, while for the "Super Linear" approach, it is steep and gets close to 0.5.

Contrarily, for the "Fitness" and "Normalized Fitness" functions, in the minimum case the entropy is relatively high. It is worth noting that a probability distribution with such high entropy is very close to the uniform distribution, and therefore, such distributions are not likely to be used for optimization as the convergence speed of the algorithm is very slow. To illustrate, note that in the minimum point of the plot, the maximum parameter in the vector of weights $\mathbf{w}$ was 0.36 and 0.37 respectively, while for the "Super Linear" function it is 0.91.

Finally, results point out that the convergence degree of the probability model is not affected by the λ parameter. However, a more detailed analysis will be carried out in the next experiment. In view of the results here, in the following we decided to use the "Super Linear" utility function exclusively.

5.2 Tuning η and λ parameters

In the literature, in order to obtain good performances, the importance of properly setting the parameters in the Gradient Search (in this case, η and λ parameters) has been published many times. In this sense, in a second experiment, the convergence of the best objective values obtained across 10^5 iterations is studied. Particularly, three different parameter values for η and λ, $\{0.01, 0.05, 0.1\}$ and $\{10^2, 10^3, 10^4\}$, were chosen, respectively, and PL-GS with all the pairwise combinations was tested. Averaged results of 10 repetitions on the instance *N-be75eec* are introduced in Fig. 2. In the results we can see that in the initial iterations, higher λ runs obtain better results, however, as the optimization progresses, algorithms converge to similar values. The convergence point is affected by the η parameter. In the first figure on the left, the convergence is the slowest $\eta = 0.01$, while the figure on the right shows the fastest convergence with $\eta = 0.1$. Usually, slow convergence is preferred, as it is a way to avoid premature convergence of the algorithm.

Regarding λ parameter, it is worth remembering that when considering a maximum number of performed objective function evaluations as stopping criterion, higher lambda implies fewer iterations of the algorithm. Therefore, for the rest of the experimental study, we think that an adequate parameter choice is $\eta = 0.01$ and $\lambda = 100$ as it preserves the algorithm from premature convergence, and uses a low number of samples to update the gradients.

5.3 Performance comparison with PL-EDA

Once the utility function, together with the η and λ parameters were decided for the proposed algorithm, we compared its performance with respect to the PL-EDA.

Each algorithm, PL-GS and PL-EDA, was run 20 times on the 50 instances of the LOLIB benchmark. In each case, the algorithms stopped after running $1000n^2$ evaluations. With respect to the tuning of the PL-EDA, the default parameters and design in [9] were considered.

5.3.1 Performance results.

Results are summarized in Table 1 as Median Relative Deviations (MRD) with respect to the optimum.

[2] Due to the excessive computational cost of computing the entropy of the PL distribution, we computed the entropy of the probability of each item of appearing at the first position. It is our belief that the second entropy measure can be used instead of the first as concentration measures of the probabilities in the PL distribution.

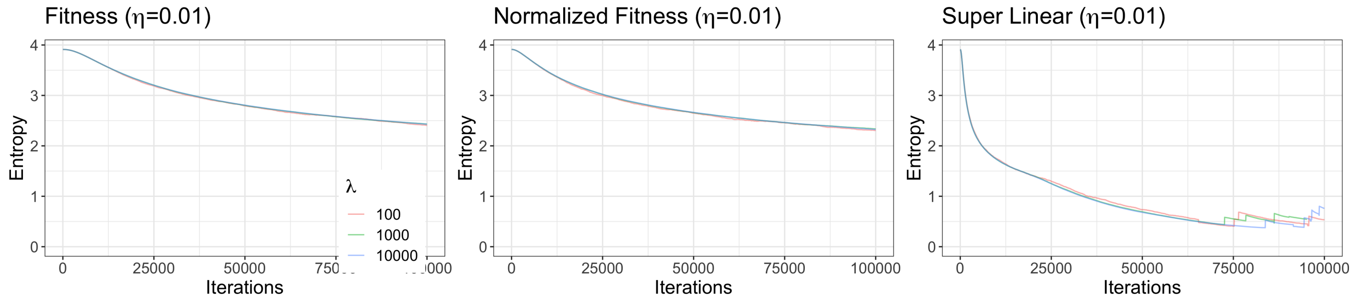

Figure 1: The entropy of the probability model defined at each iteration of the PL-GS using the three different utility functions and $\eta = 0.01$. Measures of 10 repetitions were averaged, and results for the three different λ values are presented.

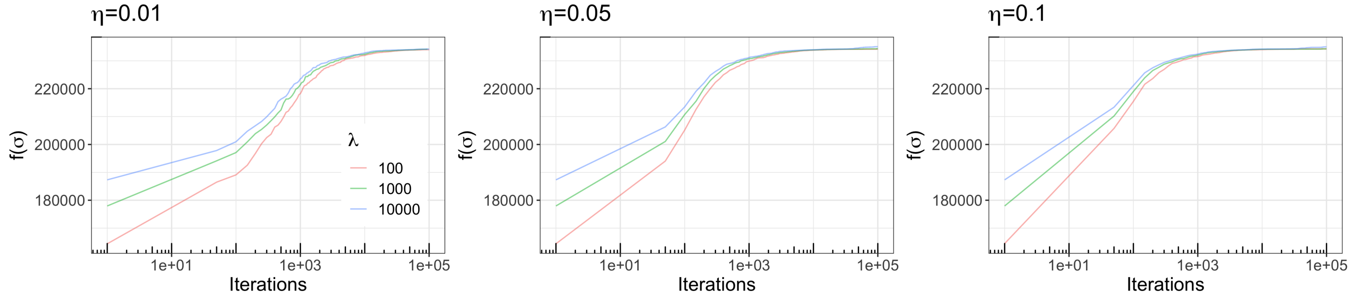

Figure 2: Objective function values obtained by the PL-GS algorithm for different values of η and λ parameters across 10^5 iterations. Particularly, $\eta \in \{0.01, 0.05, 0.1\}$ and $\lambda \in \{10^2, 10^3, 10^4\}$ parameter values are considered, and measures of 10 repetitions were averaged. Note the x-axis is in log-scale.

Specifically, median results found across the 20 repetitions were used, and the difference with respect to the best known results were normalized with this last value. According to the conducted experiments, PL-GS obtained the better results in 40 instances out of 50 instances in the LOLIB, while PL-EDA did so in 9 instances out of 50 (in the *N-t79n11xx* instance, both approaches obtained the same MRD measure).

In order to statistically assess the results obtained, we have followed the Bayesian approach presented in [3], and used the Bayesian equivalent of the Wilcoxon's test[3]. The statistical analysis was conducted on the MRD obtained by each algorithm in the 20 repetitions. The procedure used requires a quantitative description of what is understood as 'rope'. In our case, we have considered that both approaches are equivalent when the difference in MRD is smaller than 10^{-4}. Results of the analysis are depicted in Fig. 3.

As described in [3], the points in the plot represent a sampling of the posterior distribution of the probability of win-lose-tie. In other words, the closer a point is to the Gradient vertex of the triangle, the more probable it is that PL-GS produces better results (or equivalently, in the other vertices). Therefore, the three areas delimited by the dashed lines show the dominance regions, i.e., the area where the highest probability corresponds to its vertex.

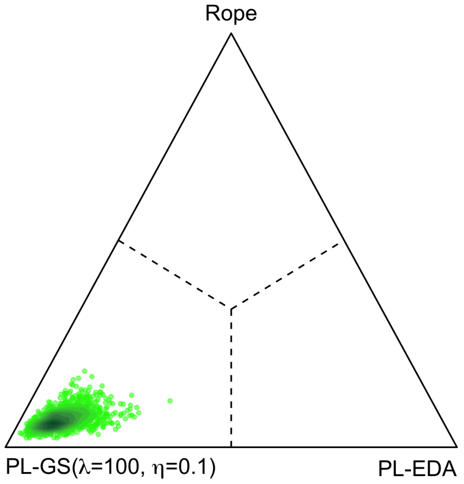

Figure 3: Simplex plot representing the posterior distribution of win-tie-lose probabilities of the PL-GS and PL-EDA algorithms

According to the Simplex plot, there is no uncertainty about the results. The most of the mass of the posterior probability distribution is close to the PL-GS vertex. Furthermore, the points are concentrated in a small region of the simplex plot, which reveals

[3] This statistical analysis is available in the development version of the **scmamp** R package [7] available at https://github.com/b0rxa/scmamp.

Table 1: Results of the PL-GS and PL-EDA for the LOP instances in the LOLIB benchmark. The Median Relative Deviations (MRD) measures of the values found across the 20 repetitions of the best known results are reported. Results in bold highlight the algorithm that obtained the lowest MRD. A maximum number of $1000n^2$ evaluations were performed by each of the algorithms.

Size	Instance	Best Known	PL-GS	PL-EDA	Size	Instance	Best Known	PL-GS	PL-EDA
50	N-be75eec	236464	**0.00975**	0.00986	44	N-t70k11xx	716994	**0.00116**	0.00311
50	N-be75np	111171	**0.00403**	0.00425	44	N-t70l11xx	980516	**0.00016**	0.00048
50	N-be75oi	362512	**0.00262**	0.00409	44	N-t70n11xx	541393	**0.00424**	**0.00424**
50	N-be75tot	553303	**0.00331**	0.00421	44	N-t70u11xx	209320	0.00187	**0.00169**
60	N-stabu70	147354	**0.00787**	0.01357	44	N-t70w11xx	122520	**0.00300**	0.00526
60	N-stabu74	8261545	**0.00764**	0.01080	44	N-t70x11xx	20928	**0.00240**	0.00277
60	N-stabu75	356758	**0.00700**	0.01252	44	N-t74d11xx	237739	0.00122	**0.00113**
44	N-t59b11xx	217295	0.00490	**0.00155**	44	N-t75d11xx	14469163	**0.00189**	0.00323
44	N-t59d11xx	16719	**0.00476**	0.01823	44	N-t75e11xx	32157	**0.00297**	0.00319
44	N-t59f11xx	138181029	**0.00047**	0.00214	44	N-t75i11xx	771149	0.00179	**0.00163**
44	N-t59i11xx	528419	**0.00046**	0.00208	44	N-t75k11xx	376725	**0.00127**	0.00189
44	N-t59n11xx	366469	**0.00282**	0.00583	44	N-t75n11xx	360336	**0.00515**	0.00543
44	N-t65b11xx	24785782	0.00588	**0.00481**	44	N-t75u11xx	60659200	**0.00125**	0.00128
44	N-t65d11xx	25253	**0.00333**	0.00407	56	N-tiw56n54	52704	0.00362	**0.00301**
44	N-t65f11xx	21716400	**0.00544**	0.00568	56	N-tiw56n58	224319954	**0.00328**	0.00350
44	N-t65i11xx	283808865	**0.00103**	0.00156	56	N-tiw56n62	566089	**0.00341**	0.00707
44	N-t65l11xx	578304	**0.00164**	0.00287	56	N-tiw56n66	2739219	**0.00560**	0.01227
44	N-t65n11xx	63567735	**0.00236**	0.00516	56	N-tiw56n67	108844	**0.00273**	0.00813
44	N-t65w11xx	93777	**0.00355**	0.00591	56	N-tiw56n72	52708323	**0.00324**	0.02243
44	N-t69r11xx	91554	0.00453	**0.00433**	56	N-tiw56r54	125224	**0.00453**	0.00650
44	N-t70b11xx	176715	**0.00214**	0.00222	56	N-tiw56r58	226547	0.00683	**0.00548**
44	N-t70d11xx	226033	**0.00466**	0.00594	56	N-tiw56r66	365146	**0.00474**	0.01239
44	N-t70d11xxb	102948	**0.00253**	0.00263	56	N-tiw56r67	129568	**0.00294**	0.00633
44	N-t70f11xx	209491	**0.00599**	0.00631	56	N-tiw56r72	222810	**0.00545**	0.01108
44	N-t70i11xx	270663	**0.00162**	0.00208	79	N-usa79	1813986	0.01313	**0.01303**

that the uncertainty related to the experiment is really low. To sum up, the expected probability of each situation, that PL-GS is (1) better, (2) equal or (3) worse than PL-EDA are 0.835, 0.066 and 0.097, respectively.

5.3.2 Time comparison. For the sake of analyzing the time required by each of the algorithms to run $1000n^2$ evaluations and obtain the results in Table 1, in Fig. 4, a summary of the time consumption of PL-GS and PL-EDA is provided. Specifically, time measures of the runs were aggregated by the size of the instances, and presented as boxplots. The boxplots in the figure show that the time consumption of the PL-GS algorithm is less than PL-EDA, systematically. Moreover, the results show that the scalability of the proposed algorithm is nearly constant when compared to that of PL-EDA, and the variance is very low.

6 CONCLUSIONS AND FUTURE WORK

Recently, Gradient Search (GS) has gained relevance due to its applications for the continuous parameter estimation when training neural networks. Nevertheless, its application has been mainly restricted to optimizing problems in the continuous domain, and works considering a model-based GS on the combinatorial case have been limited to a few publications that only consider binary spaces. In this paper, we approached the optimization of permutation-based problems by means of GS. In particular, we have devised the necessary mathematical tools in order to efficiently compute and update the continuous parameters of the Plackett-Luce probability model defined over the set of permutations.

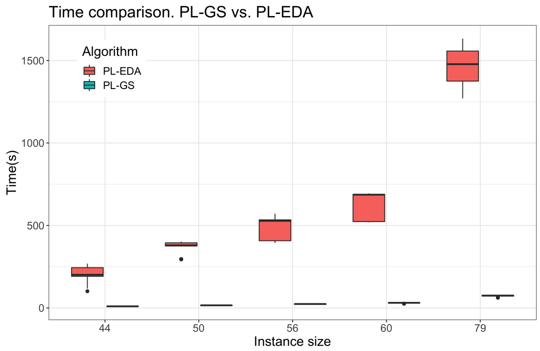

Figure 4: Summary of time required by PL-GS and PL-EDA algorithms to carry out the executions. Results have been aggregated by instance size, and presented as boxplots.

Interestingly, this work follows the recent trend in the literature (see for instance [2, 21, 31]) of bringing algorithms originally defined for the continuous domain to the space of permutations.

Experiments, conducted on a benchmark suite of 50 instances of the linear ordering problem, proved the validity of our proposal which obtained better performance and time measures than the competitor algorithm considered.

Possible future lines of work include trying other *flavors* of gradient search [11, 28], considering other *ordered weighted averaged operators* (OWAs) [33] as utility functions, and self-adapting the η and λ parameters of PL-GS. With regard to the probability model employed for defining the random variable to optimize, other models defined over permutation spaces such as Bradley-Terry [27] or models based on *inversion tables* [15] could also be considered in the same context.

ACKNOWLEDGMENTS

Josu Ceberio has been partially supported by the Research Groups 2019-2020 (IT1244-19) and Elkartek Program (PROMISE) from the Basque Government, the TIN2016-78365-R research project from the Spanish Ministry of Economy, Industry and Competitiveness, and the European Research Council H2020 (the EMPHATIC project. Valentino Santucci has been partially supported by the research projects: "Università per Stranieri di Perugia – Finanziamento per Progetti di Ricerca di Ateneo — PRA 2020", and "Università per Stranieri di Perugia – Artificial intelligence for education, social and human sciences".

REFERENCES

[1] C. Robinson N. Quadrianto A. Gadetsky, K. Struminsky and D. Vetrov. TO APPEAR. Low-variance Black-box Gradient Estimates for the Plackett-Luce Distribution. In Proceedings of AAAI 2020. https://arxiv.org/abs/1911.10036

[2] M. Baioletti, A. Milani, and V. Santucci. 2020. Variable neighborhood algebraic Differential Evolution: An application to the Linear Ordering Problem with Cumulative Costs. Information Sciences 507 (2020), 37–52.

[3] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon. 2017. Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis. Journal of Machine Learning Research 18, 77 (2017), 1–36.

[4] A. Berny. 2000. Selection and Reinforcement Learning for Combinatorial Optimization. In Parallel Problem Solving from Nature PPSN VI. Springer Berlin Heidelberg, 601–610.

[5] A. Berny. 2001. Statistical Machine Learning and Combinatorial Optimization. Springer Berlin Heidelberg, 287–306.

[6] A. Berny. 2002. Boltzmann Machine for Population-Based Incremental Learning. In Proceedings of the 15th European Conference on Artificial Intelligence. IOS Press, 198–202.

[7] B. Calvo and G. Santafe. 2015. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. The R Journal 8, 1 (2015), 248–256.

[8] J. Ceberio. 2014. Solving permutation problems with estimation of distribution algorithms and extensions thereof. PhD Dissertation. University of the Basque Country (UPV/EHU).

[9] J. Ceberio, A. Mendiburu, and J. A Lozano. 2013. The Plackett-Luce ranking model on permutation-based optimization problems. In 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico. IEEE, 494–501.

[10] D. E. Critchlow, M. A Fligner, and J. S. Verducci. 1991. Probability models on rankings. Journal of Mathematical Psychology 35, 3 (1991), 294 – 318.

[11] J. Ba D. P. Kingma. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[12] J.A. Lozano E. Irurozki, B. Calvo. 2016. PerMallows: An R Package for Permutations, Mallows and Generalized Mallows Models. Journal of Statistical Software 71, 1 (2016), 1–30.

[13] M. A. Fliger and J. S. Verducci. 1993. Probability Models and Statistical Analysis for Ranking Data. Springer.

[14] M.A. Fligner and J.S. Verducci. 1988. Multistage ranking models. J. Amer. Statist. Assoc. 83, 403 (1988), 892–901.

[15] M. A. Fligner and J. S. Verducci. 1986. Distance based ranking Models. Journal of the Royal Statistical Society 48, 3 (1986), 359–369.

[16] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. arXiv preprint arXiv:1604.00772 (2016).

[17] E. Irurozki. 2014. Sampling and Learning Distance-based Probability Models for Permutation Spaces. PhD Dissertation. University of the Basque Country (UPV/EHU).

[18] H. Joe and J. S. Verducci. 1993. On the Babington Smith Class of Models for Rankings. In Probability Models and Statistical Analyses for Ranking Data, M. A. Fligner and J. S. Verducci (Eds.). Springer New York, New York, NY, 37–52.

[19] J. A. Lozano and E. Irurozki. 2012. Probabilistic Modelling on Rankings. In Asian Conference on Machine Learning, 4-6 November 2012, Singapore.

[20] R. D. Luce. 1959. Individual Choice Behavior. Wiley, New York.

[21] V. Santucci M. Baioletti, A. Milani. 2018. MOEA/DEP: An Algebraic Decomposition-Based Evolutionary Algorithm for the Multiobjective Permutation Flowshop Scheduling Problem. In Evolutionary Computation in Combinatorial Optimization. 132–145.

[22] L. Malagò, M. Matteucci, and G. Pistone. 2011. Towards the Geometry of Estimation of Distribution Algorithms Based on the Exponential Family. In Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms. Association for Computing Machinery, 230–242.

[23] C. L. Mallows. 1957. Non-null ranking models. Biometrika 44, 1-2 (1957), 114–130.

[24] F. Mosteller. 1951. Remarks on the method of paired comparisons. I. The least squares solutions assuming equal standard deviations and equal correlations. Pyschometrika 16 (1951), 3–9.

[25] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen. 2017. Information-Geometric Optimization Algorithms: A Unifying Picture via Invariance Principles. Journal of Machine Learning Research 18, 1 (2017), 564–628.

[26] R. L. Plackett. 1975. The Analysis of Permutations. Journal of the Royal Statistical Society 24, 10 (1975), 193–202.

[27] M. E. Terry R. A. Bradley. 1952. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. Biometrika 39, 3/4 (1952).

[28] S. Ruder. 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016).

[29] V. Santucci and J. Ceberio. 2020. Using Pairwise Precedences for Solving the Linear Ordering Problem. Applied Soft Computing Journal 87 (2020).

[30] L. L. Thurstone. 1927. A law of comparative judgment. Psychological Review 34, 4 (1927), 273–286.

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

[32] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. 2014. Natural Evolution Strategies. Journal of Machine Learning Research 15, 27 (2014), 949–980.

[33] R. R. Yager. 1988. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics 18, 1 (1988), 183–190.

[34] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. 2004. Model-based search for combinatorial optimization: A critical survey. Annals of Operations Research 131, 1-4 (2004), 373–395.