Creating Difficult Instances of the Linear Ordering Problem

Aritz Pérez*and Josu Ceberio†
*Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain
†Faculty of Computer Science, 20018 Donostia, Spain
University of the Basque Country UPV/EHU
email: aperez@bcamath.org, josu.ceberio@ehu.eus

***Abstract*—Generating random instances of a given problem is a task that is addressed very frequently in combinatorial optimization. Some times this task is motivated by the lack of challenging instances, some other times practitioners have few instances and need to generate large benchmarks to evaluate their algorithms. Be that as it may, creating instances that are challenging enough for the algorithm at hand is not trivial, and implies other considerations such as defining the concept of difficulty.**

In this paper, we address the problem of randomly generating difficult instances of the linear ordering problem for local search algorithms. Based on the joint analysis of the problem and a neighborhood structure, we conclude that one way to create difficult instances is to organize its parameters in the way that the variance of the fitness in the neighborhood is very low. To this end, we propose a simple procedure that generates the parameters matrix in such a way that for any row and column, the sum of the parameters is constant. The experimental study shows that the proposed method generates instances of higher difficulty than the standard method, setting the parameters independently and identically distributed according to a uniform distribution.

I. Introduction

In the field of evolutionary computation, it is common to use benchmarks of instances of a given problem in order to evaluate the performance of existing and newly developed algorithms. When the final goal is to solve a specific real-world problem, real instances are used to carry out such comparisons, and, thus, we are not interested in an extensive performance evaluation. However, when the objective of the research is to contribute with methodological developments, then large benchmarks of instances are needed in order analyze the behavior of the proposed algorithm under different scenarios. In this sense, real-world benchmarks are usually limited in size and variety of instances.

At this point, it is a usual practice, based on the knowledge of the problem (limited in most cases), to create new "challenging" instances artificially [1], [2], [3], [4], [5]. In this sense, a recurrent option is to generate instances by sampling their parameters uniformly at random (u.a.r) in some ranges. The underlying assumption states that sampling u.a.r in the space of parameters is equivalent to sampling u.a.r in the space of instances and, thus, uniform in terms of difficulty. Nonetheless, recent works on this line [6], [7] have shown that such statement is not necessarily true.

In [6], the authors focused on the random generation of artificial instances of combinatorial optimization problems. Particularly, they stated that most of the algorithms take only into account the ordering of the solutions (better, worse...) during the optimization and, therefore, any instance can be seen as a ranking of all the solutions in the search space. In this context, the authors discovered that there exist classes of equivalence of the mentioned rankings according to their difficulty (in terms of number of local optima). Not limited to that, conducted experiments demonstrated that when sampling independently and identically distributed (i.i.d.) parameters according to a uniform distribution, the majority of the instances induce rankings that belong to "easy" classes of equivalence (have one or two local optima).

On that research line, in [7] the authors tried to go deeper in the difficulty of instances for best improvement local search algorithms when sampling the parameters u.a.r. Despite their simplicity, local search algorithms permit easily tracking the optimization process and, thus, it is possible to carry out an exact study of the difficulty of the instance on the basis of fitness landscape analysis [8]. The work in [7] focused on three problems and four neighborhood structures, and tried to answer the following question: *are the artificially generated instances uniform in terms of difficulty?* The experimental study revealed that (1) by sampling the parameters uniformly at random the obtained instances are non-uniform in terms of difficulty, (2) the distribution of the difficulty strongly depends on the pair problem-neighborhood considered, and (3) given a problem, the distribution of the difficulty seems to depend on the landscape induced by the neighborhood structure and its size.

However, what creates the difficulty for a specific algorithm in each problem is unique and it is not clear. In this work, we take as case of study the *Linear Ordering Problem* (LOP) [9], [10] and *best improvement local search algorithm* (BI) under the insert neighborhood. In that context, we approach the problem of creating difficult instances.

Firstly, we defined the difficulty of an instance as 1 minus the probability of reaching the global optimum solution after performing a BI starting from a random solution. Secondly, based on the joint study of the LOP and insert neighborhood, we considered that one way to create difficult artificial instances can be to sample a set of parameters that induces low

fitness variations in the neighborhood of a solution.

On that line, we propose a procedure for generating artificial instances by defining a parameters matrix of the instance in which every row and column sum a constant value. This type of parameters matrix tends to induce low fitness differences of the solutions in the insert neighborhoods.

In the experimental section, we generate thousands of instances of the LOP for different sizes of the problem. The instances have been generated by using i) the classical uniform procedure, and ii) the proposed procedure. Then, we estimate the difficulty of each instance generated for BI under the insert neighborhood and we analyze the evolution of the difficulty of the generated instances with respect to the size of the problem, n. The results point out that the proposed procedure obtain more difficult instances than the classical option.

The remainder of the paper is organized as follows: in Section II the necessary background on combinatorial optimization, LOP, local search and difficulty is introduced. In Section III, the main contribution of the paper, a method to create difficult instances of the LOP for BI is proposed. Afterwards, the experimental study is presented in Section IV. Finally, conclusions and research lines for future work are summarized in Section V.

II. Background

A combinatorial optimization problem (COP), denoted as $\mathbf{P} = (\Omega, f)$, consists of a finite (or infinite countable) domain of solutions Ω, also known as search space, and an *objective function* f which is formalized as

$$\begin{aligned} f \quad : \quad & \Omega \to \mathbb{R} \\ & x \mapsto f(x) \end{aligned}$$

We call to the value $f(x)$ the *fitness* of x. The definition of a problem includes certain inputs that are known as *parameters*. The collection of parameters that describe a particular case of the problem is called the *instance*.

The aim in a COP is to, given an instance, find $x \in \Omega$ such that f is maximized (or minimized). As the computation of $f(x)$ is closely tied to the instance, in the remainder of the paper, we say that f is an instance or an objective function indistinctly. The LOP is an example of a COP. In the following, a brief introduction to it is presented.

A. The Linear Ordering Problem

Given a matrix $B = [b_{i,j}]_{n \times n}$ of numerical entries, the LOP [9], [10] consists of finding a simultaneous permutation σ of the rows and columns of B, such that the sum of the entries above the main diagonal is maximized (or equivalently, the sum of the entries below the main diagonal is minimized). The equation below formalizes the objective function:

$$f(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{\sigma_i, \sigma_j} \tag{1}$$

where σ_i denotes the index of the row (and column) ranked at position i in the solution σ. The numerical entries $b_{i,j}$ for $i, j = 1, ..., n$ are the parameters of the problem and, generally, $b_{i,i} = 0$ for $i = 1, ..., n$. A particular assignment of values to these parameters constitutes an instance. Fig. 1 introduces an illustrative example of an LOP instance in its *triangulation* form. Gray background parameters are the only

	1	2	3	4	5
1	0	16	11	15	7
2	21	0	14	15	9
3	26	23	0	26	12
4	22	22	11	0	13
5	30	28	25	24	0

Fig. 1: LOP instance of size $n = 5$.

ones considered in the computation of the fitness function. The search space of solutions Ω consist of all permutations of size n, so its size is $|\Omega| = n!$.

B. Local search on the LOP

Local Search algorithms (LS) optimize based on a neighborhood system, a function that relates a set of solutions to each solution $x \in \Omega$. Formally a *neighborhood* is defined as

$$\begin{aligned} N : \quad & \Omega \to 2^{|\Omega|} \\ & x \mapsto N(x) \end{aligned}$$

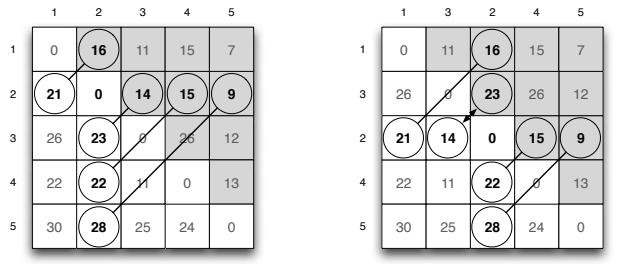
where $2^{|\Omega|}$ represents the power set of the domain Ω.

LS is a sequential optimization heuristic that progresses by moving from a solution to a neighboring solution. Consequently, the behavior of a particular LS is determined by the neighborhood defined, and the criteria used to select a solution from the neighborhood. Two of the most used strategies are the *best improvement* (BI) and the first improvement selection criteria. BI progresses by selecting, among the neighboring solutions the solution with the best fitness, while the first improvement progresses by selecting a neighboring solution with a better $f(\sigma)$ value. Both algorithms are said to converge to a (local) optimum solution when all the neighboring solutions have equal or worse $f(\sigma)$ value.

In local search optimization, a wide range of neighborhood structures have been proposed in order to deal with combinatorial problems. Depending on the characteristics of the problem, some neighborhoods show better performance than others [11]. According to [7], the insert neighborhood is the structure that most favorable landscapes generates for the LOP. From our perspective, this is probably motivated by its considerable size and the few number of parameters exchanged between neighboring solutions.

Let us now consider a permutation σ of size n. The insert neighborhood considers as neighbors all the solutions that can be obtained by moving any item from a position i to any position j. The items between positions i and j are shifted. As a result, the size of the neighborhood is $(n-1)^2$.

Fig. 2 presents an example of an LOP instance for two different solutions: e and σ. The second solution is obtained by performing an insert movement of item 2 to position 3. In both cases, the pairs of parameters associated to item 2 are circled. As can be seen in the figure, due to the insert



(a) $e = (1, 2, 3, 4, 5),\ f(e) = 138.$ (b) $\sigma = (1, 3, 2, 4, 5),\ f(\sigma) = 147.$

Fig. 2: Two different solutions for an instance of $n = 5$. Circled values denote the parameters associated to item 2.

movement, a pair of parameters, $(14, 23)$ is swapped from side, and thus, $f(\sigma)$ increases from 138 to 147. Note that the rest of the parameters, despite they are reallocated, they remain in the same side with respect to the main diagonal. In general, if we insert the i-th element in the j-th position we swap $|i - j|$ parameters [7]. The general effect of an insert operation is illustrated in Figure 3.

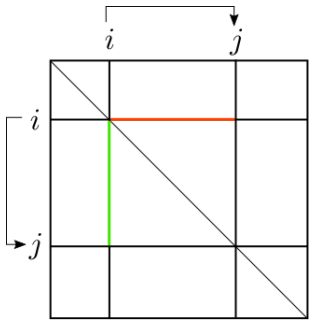


Fig. 3: The parameters swapped by insert i-th element in position j. Green lines and red lines represent the parameters that are added and subtracted to the objective function, respectively

C. Measuring the Difficulty

As proposed in [7], we consider the difficulty of an instance of the LOP as 1 minus the probability of reaching the global optimum solution by using BI and by picking a solution of Ω at random. In other words, the difficulty is 1 minus the size of the basin of attraction of the global optimum divided by the size of the search space, $n!$. In this work, we estimate the difficulty of an instance by running 100 times BI initialized with a random solution. The difficulty is estimated as 1 minus the percentage that BI have reached the global optimum.

It is our intuition that if exists a solution in the neighborhood whose fitness is quantitatively larger than the rest, then decision to move to this solution by the BI tends to increase the chance to reach to the global optimum. In this scenario, we say that the decision is supported with strong evidence and the neighborhood becomes more informative. In this sense, in the next section, we propose a method to generate artificial instances in which the solutions in the neighborhood have small differences in the fitness.

III. Creating Difficult Instances of the LOP

Usually, in the literature, in order to generate artificial instances of LOP the parameters are sampled according to a uniform distribution [7]. Using this procedure it is possible to generate an instance B with unbalanced sums by rows and by columns which can be easily exploited by heuristic optimization algorithms, such as BI under the insert neighborhood. Nonethelss, if we generate instances in which the independent sum of rows and columns are equally weighted, then the insert neighborhood tends to provide similar fitness variations in the neighborhood of a solution, and it becomes less informative (see Figure 3).

In this work, with the aim of generating more difficult instances than the uniform approach, we propose a method that produces a matrix B for which the sum of each row and each column sum 1. Specifically, the scheme proceeds as follows. First, two matrices R and C are created. In R, the rows of parameters are obtained by sampling a Dirichlet distribution. Similarly, in C, the sampled vectors correspond to columns in the matrix. Next, once R and C are obtained, the randomly generated instance is given as $B = 1/2 \cdot (R + C)$. Finally, following the next iterative normalization procedure we obtain a matrix B for which any row or column sum one. The procedure works as follows, first, the parameters of every row are normalized independently in order to sum 1. Next, the same normalization process is applied to the columns. Both steps are subsequently repeated until result converges to the matrix with the characteristics above.

Note that in LOP, $b_{i,i} = 0$ for $i = 1, .., n$. Thus the diagonal of R and C have to be zero and, thus, we sample a Dirichlet distribution with $n-1$ parameters for each row and column of R and C, respectively. In this work, the parameters are equal to one, that is $\alpha = \alpha_i =$ for $i = 1, ..., n - 1$.

IV. Experimental Study

In this section, we artificially generate LOP instances following two different schemes, and, we analyze the generated instances in terms of the distribution of the difficulty.

For this purpose, we have artificially generated 10^4 instances of the LOP for each problem size $n = 8, ..., 20$. The parameters of the instances have been sampled by using i) the classic uniform procedure in the $[0, 1]$ interval and ii) the proposed Dirichlet procedure with $\alpha = 1$. Next, we have obtained the global optimum of each instance by solving it using the branch and bound algorithm proposed by Charon et al. [12]. Finally, for each instance generated we have run a BI under the insert neighborhood 100 times, and estimated its

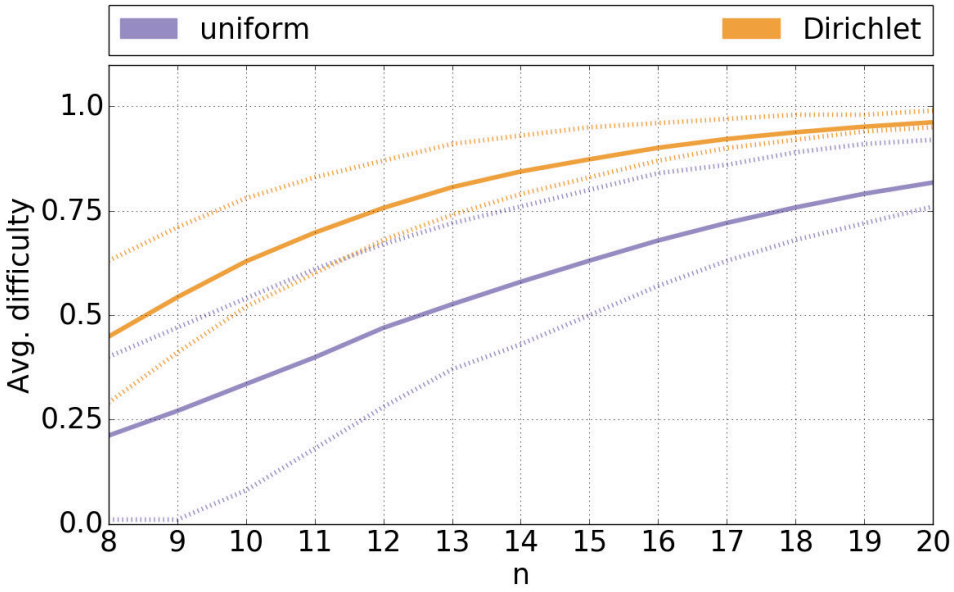


Fig. 4: The evolution of the average difficulty (together with the first and third quartiles) for the uniform and the Dirichlet with respect to the size of the problem $n = 6, ..., 20$.

difficulty by counting the percentage of times it reaches the global optimum.

We have analyzed the evolution of the distribution of the difficulty with respect to the size of the LOP, n. The results are summarized in Figs. 4 and 5. The first illustrates the evolution of the average difficulty (and the first and third quartiles) for the compared schemes for the different n values. Figure 5 provides a more detailed view of the results showing the distribution of the difficulty.

Results reveal that, as n increases, instances tend to be more difficult using both procedures. In addition, we see that the instances generated by the Dirichlet procedure are more difficult. The dispersion of the generated difficulties using the Dirichlet procedure is smaller than the obtained by the uniform procedure, especially for the largest instances. Moreover, as n increases the third quartile of the difficulty for the Dirichlet procedure is higher than the fist quartile of the uniform procedure.

V. Conclusion & Future Work

In this work, we have addressed the problem of generating difficult instances of the linear ordering problem for the best improvement local search under the insert neighborhood. Based on the joint analysis of the LOP instances and the insert neighborhood structure, we concluded that a possible manner is to organize its parameters in such a way that the variance of the fitness in the neighborhood is very low. To this end, we propose a procedure for generating artificial instances as a matrix of parameters in which the sum of each row and each column is constant. Conducted experiments show that the proposed method tends to generate instances of higher difficulty than other standard methods such as setting the parameters i.i.d. according to a uniform distribution.

However, there are a number of questions still pending. For instance, results in Fig. 5 show small peaks in the distribution of the difficulty, that are consistently repeated for the two instance generation methods across the different n values. The answer should be in the definition of the problem and the structure of the neighborhood.

In addition, we aim to study the effect of the α parameters of the Dirichlet procedure in the distribution of the difficulty. It is our intuition that α parameter could be used in order to further increase the difficulty of the generated instances.

Finally, we find interesting to extend this work to other problem-neighborhood pairs. For instance, for the pair Quadratic Assignment Problem (QAP) and interchange neighborhood [11], [13], it is very probable that the same principle of low variance fitness neighbors will produce difficult instances. Nevertheless, the manner in which the instances are generated for QAP is apparently much more complex as it doubles the number of parameters of the LOP. Not limiting to the domain of local search algorithms, it is possible to extend the work to other algorithms, as the principle of low variance fitness instances may produce difficult scenarios for population-based algorithms as well.

Acknowledgment

This work has been partially supported by the Research Groups 2013-2018 (IT-609-13), BERC 2014-2017, and ELKARTEK programs (Basque Government), the projects TIN2016-78365-R and TIN2017-82626-R (Spanish Ministry of Economy, Industry and Competitiveness) and Severo Ochoa Program SEV-2013-0323 (Spanish Ministry of Economy, Industry and Competitiveness).

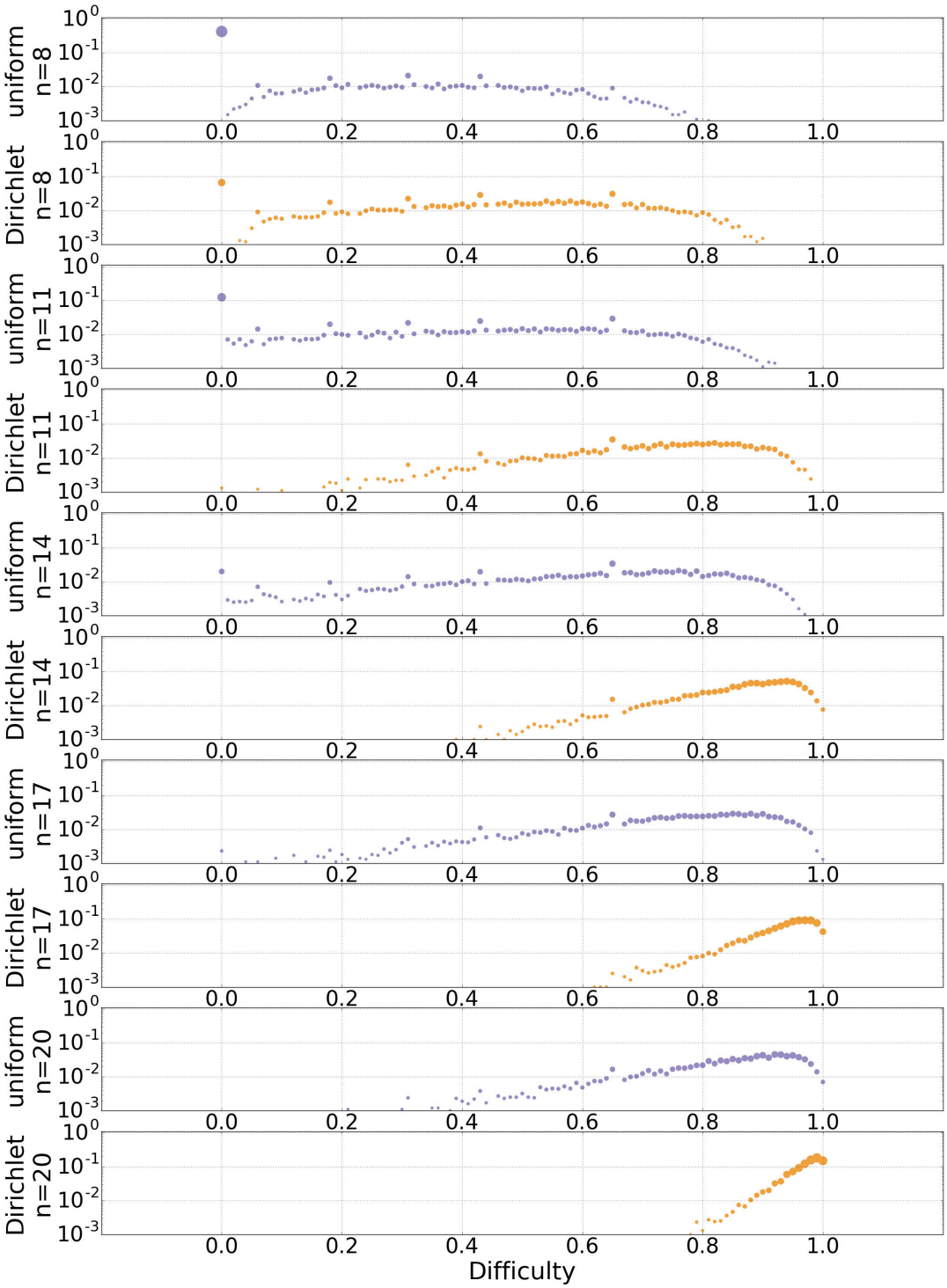


Fig. 5: The distribution of the difficulty of the instances generated by using the uniform and Dirichlet procedures for different problem sizes, $n = 8, 11, 14, 17, 20$. The x axis is the difficulty and the y axis the frequency in which instances of a given difficulty are generated. Each point corresponds to a difficulty and its area is proportional to the number of instances generated with this difficulty.

References

[1] I. P. Gent and T. Walsh, "The TSP phase transition," *Artificial Intelligence*, vol. 88, no. 1–2, pp. 349 – 358, 1996.

[2] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, 1993.

[3] Z. Drezner, P. Hahn, and É. Taillard, "Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods," *Annals of Operations Research*, vol. 139, no. 1, pp. 65–94, 2005.

[4] T. Schiavinotto and T. Stützle, "The linear ordering problem: Instances, search space analysis and algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 367–402, 2005.

[5] A. Duarte, M. Laguna, and R. Martí, "Tabu search for the linear ordering problem with cumulative costs," *Computational Optimization and Applications*, vol. 48, no. 3, pp. 697–715, 2011.

[6] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Are we generating instances uniformly at random?" in *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2017, pp. 1645–1651.

[7] A. Perez, J. Ceberio, , and J. A. Lozano, "Are the artificially generated instances uniform in terms of difficulty?" in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018.

[8] M. Tayarani-N. and A. Prgel-Bennett, "On the landscape of combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 420–434, June 2014.

[9] R. Martí and G. Reinelt, *The linear ordering problem: exact and heuristic methods in combinatorial optimization*. Springer, 2011, vol. 175.

[10] J. Ceberio, A. Mendiburu, and J. A. Lozano, "The Linear Ordering Problem Revisited," *European Journal of Operational Research*, vol. 241, no. 3, pp. 686–696, 2014.

[11] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano, "A review of distances for the mallows and generalized mallows estimation of distribution algorithms," *Computational Optimization and Applications*, vol. 62, no. 2, pp. 545–564, 2015.

[12] I. Charon and O. Hudry, "A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments," *Discrete Applied Mathematics*, vol. 154, no. 15, pp. 2097 – 2116, 2006.

[13] J. Ceberio, A. Mendiburu, and J. A. Lozano, "Kernels of mallows models for solving permutation-based problems," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15. ACM, 2015, pp. 505–512.