Generating Linear Ordering Problem Instances by Controlling Structural Inconsistencies

Maialen Beristain-Cortés[1*][0009−0003−4876−5889], Leticia Hernando[2][0000−0003−2828−0748], and Jose A. Lozano[1,3][0000−0002−4683−8111]

[1] Basque Center for Applied Mathematics (BCAM), Bilbao, 48009, Spain
[2] University of the Basque Country EHU, Leioa, 48940, Spain
[3] University of the Basque Country EHU, Donostia-San Sebastián, 20018, Spain
{mberistain,jlozano}@bcamath.org, leticia.hernando@ehu.eus

Abstract. In combinatorial optimization, diverse instances are essential to build representative benchmarks for algorithm evaluation. Since many of these problems are NP-hard, worst-case complexity provides limited insight into practical instance difficulty, making empirical evaluation across diverse instances necessary. For the linear ordering problem, the Fourier transform is a useful tool to quantify the weight of the NP-hard component, offering a structural indicator associated with instance difficulty, not relaying on any particular optimization algorithm.
In this work, we propose a novel instance generation framework for the LOP based on controlling the number of 3-cycles, which capture minimal inconsistencies that break transitivity, thus influencing structural-complexity. We analyze the resulting instances using two complementary indicators: 3-cycle density and the NP-hard component weight. Experiments show that varying the number of 3-cycles yields a broader range of instances than standard benchmarks, enabling more diverse test sets for algorithm evaluation.

Keywords: Linear Ordering Problem · Instance Generation · Instance Difficulty · Instance Complexity.

1 Introduction

Combinatorial optimization comprises a broad class of problems characterized by discrete decision spaces. Among these, problems belonging to the *NP-hard* complexity class occupy a central role, as there is no known deterministic algorithm capable of solving all their instances in polynomial time on the instance size. This complexity classification characterizes the intrinsic hardness of such combinatorial optimization problems (COPs) in the worst case. However, the computational complexity of a COP does not determine the practical difficulty of solving a particular instance. While NP-hardness guarantees the existence of hard instances, the difficulty encountered in practice may vary depending on the

*Corresponding author.

structural properties of the instance at hand and on how algorithms explore the search space. Thus, instance difficulty is inherently algorithm-dependent [24].

These observations explain the extensive use of both exact and heuristic approaches in combinatorial optimization and have motivated long-standing research aimed at identifying the features that drive instance difficulty, as such insight is essential for the design of effective and robust algorithms [23]. This perspective is further reinforced by the No Free Lunch (NFL) theorems, which state that no algorithm outperforms all others across all instances[27]. In this scenario, benchmark instances are commonly used to evaluate algorithms, but real-world sets are often limited in size and diversity, leading to the use of artificially generated "challenging" instances[6,16]. In this work, we address this limitation by proposing a new structure-aware generation procedure for linear ordering problem (LOP) instances with controlled and distinct structural patterns. These instances complement standard benchmarks such as LOLIB library [21], enabling a more comprehensive evaluation of algorithmic behavior.

The LOP is a classical COP that, given a square matrix $A = [a_{ij}]_{n\times n}$, consists of finding the joint permutation of rows and columns that maximizes the sum of the entries above the main diagonal [4,20,22]. Formally, the aim is to find a permutation σ that maximizes the following function:

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

where $\sigma(k)$ denotes the row/column of A located in the k-th position, and $\sigma \in \Sigma_n$ where Σ_n is the set of all permutations of size n.

The interest in the LOP is justified by its long research history, due to both its theoretical challenges and its practical relevance in diverse fields, such as archeology [12] and economics [18]. These applications often give rise to instances with markedly different structural properties, reinforcing the need for instance-aware analysis. In this line, several works have used different measures of instance difficulty [3,15,16,22], showing empirically a clear correlation between these metrics and the difficulty of LOP instances. From a spectral point of view, recent studies [1,2] have introduced the Fourier transform (FT) as a useful tool to quantify the weight of the NP-hard component of LOP instances, offering a structural indicator associated with instance difficulty, not relaying on any particular optimization algorithm.

From a complementary perspective, this paper presents two main contributions to the structural study of LOP instances, introducing properties that are expected to influence instance difficulty. First, we introduce the number of cycles in the graph representation of an instance as a novel variable for characterizing its structural-complexity[4], since cycles are minimal structures that break transitivity and induce inconsistencies. Second, we propose an instance generation

[4] Here and throughout the paper, "structural-complexity" refers to the richness of the structural configuration of the instance, and should not be confused with computational complexity or instance difficulty.

method that enables controlled variation of these structural properties. Based on this, we define two complementary structural-complexity indicators: (i) the density of 3-cycles, and (ii) the weight of the NP-hard component obtained via the FT. These indicators define a two-dimensional space in which instances can be positioned according to their values.

The remainder of the paper is organized as follows. In the next section, we discuss the notion of difficulty in LOP instances and define the aforementioned two structural-complexity indicators. Section 3 describes the experimental setup and the proposed instance generation methodology, and analyzes the resulting instances within the two-dimensional space defined by the structural indicators. We include an initial set of computational experiments aimed at exploring whether certain structural configurations may favor particular neighborhood structures. Finally, some conclusions and outlines directions for future work are presented in Section 4.

2 Measuring the Difficulty of an LOP Instance

As previously noted, instance difficulty lies in the structural characteristics of the instance and in the specific algorithm used to explore the search space. In this section, we define two structural-complexity indicators for LOP instances.

2.1 Cycles: Structural Sources of Inconsistencies

We first introduced the LOP through the formulation given in Equation (1), commonly referred to as the triangulation problem. However, depending on the context of the study, the LOP admits several equivalent reformulations [20]. Among these perspectives, the graph-based representation of the LOP proves particularly intuitive for structural analysis, as it offers a natural way to visualize pairwise interactions.

Let $D = (V, E)$ be a directed graph (or digraph), where V is a finite set of vertices and $E \subseteq V \times V$ is a set of arcs (or directed edges), consisting of ordered pairs of distinct vertices (i, j) with $i, j \in V$. Let w_{ij} denote the weight associated with the arc (i, j), for all $(i, j) \in E$ [8]. The LOP aims to find a linear ordering of the vertices such that the total weight of the forward arcs, according to the given ordering, is maximized [20]. That is, the objective is to maximize

$$\sum_{\sigma(i)<\sigma(j)} w_{ij}, \tag{2}$$

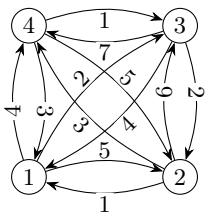
over all possible linear orderings $\sigma \in \Sigma_n$, where $\sigma(k)$ denotes the position of vertex k in the ordering.

Within this framework, an LOP instance matrix A naturally defines a pairwise preference structure on the digraph, where each entry a_{ij} is interpreted as the weight of the arc (i, j). Precisely, for any pair of distinct vertices $i, j \in V$, the direction (i, j) is preferred to (j, i) if $a_{ij} > a_{ji}$. In this setting, the preference

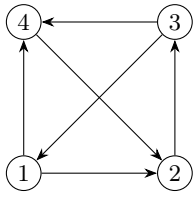
structure can be represented as a tournament (a directed graph in which every pair of distinct vertices is connected by exactly one arc [8,19]), where the arc between each pair of vertices (i, j) is oriented according to the corresponding preferred direction. An illustrative example can be found in Figure 1.

$$A = \begin{pmatrix} 0 & 5 & 2 & 4 \\ 1 & 0 & 6 & 3 \\ 4 & 2 & 0 & 7 \\ 3 & 5 & 1 & 0 \end{pmatrix}$$

(a) LOP instance matrix



(b) Complete digraph



(c) Induced tournament

Fig. 1: (a) LOP instance matrix A; (b) complete weighted digraph given by the entries a_{ij} in A; (c) tournament induced by preferences $a_{ij} > a_{ji}$.

A cycle in a directed graph is a directed path that begins and ends at the same vertex [8]. The preference structure shown in Figure 1(c) contains several directed cycles, such as $(1, 2, 3)$, $(2, 3, 4)$ and $(1, 4, 2, 3)$. Formally, an ordered triple (i, j, k) is said to form a directed 3-cycle, or triangle, if

$$a_{ij} > a_{ji}, \quad a_{jk} > a_{kj}, \quad \text{and} \quad a_{ki} > a_{ik}. \tag{3}$$

Equivalently, we denote this situation by

$$i \succ j, \quad j \succ k, \quad k \succ i, \qquad \text{or} \qquad i \to j \to k \to i, \tag{4}$$

where, for any two vertices $i, j \in V$, we write $i \succ j$ or $i \to j$ whenever $a_{ij} > a_{ji}$. Note that $j \to k \to i \to j$ represents the same triangle as in Equations (3) and (4). The presence of a cycle implies a violation of transitivity, a fundamental property of linear orderings: if an ordering ranks i before j, and j before k, then it must also rank i before k. Such consistency cannot be achieved in a cycle since the cyclic structure necessarily produces inconsistent preferences (see Equation (4)). Thus, transitivity is equivalent to acyclicity, and vice versa. Therefore, if an LOP instance matrix induces a cyclic preference structure, which is almost always the case, no solution can satisfy all preferences simultaneously, as every feasible solution corresponds to a linear ordering. The problem therefore consists in selecting an acyclic tournament that minimizes the total weight of backward arcs, deciding which arcs must violate the preference structure.

Several works have investigated the number of directed cycles of various lengths and their structural arrangement in graphs and tournaments [13,14]. In particular, Charon and Hudry [5] point out that triangle-based information is insufficient to explain the computational difficulty of linear ordering instances, they do not capture all the constraints defining the convex hull of the linear

ordering polytope. However, instance difficulty arises from global interactions beyond the constraints induced by triangles. Nevertheless, 3-cycles remain the minimal algebraic structures that break transitivity and therefore represent the elementary sources of inconsistency.

From an experimental point of view, this motivates the use of the number of 3-cycles as a first-order descriptor of instance inconsistency. Although triangle-based information does not capture the full global structure underlying computational difficulty, it does completely characterize non-transitivity. Accordingly, we expect this measure to be associated with instance difficulty. In this work we adopt 3-cycle density as a natural baseline for quantifying local inconsistency, with the aim of examining whether increasing minimal violations of transitivity are associated with higher weights of the NP-hard component. This choice is not meant to fully characterize instance hardness, but rather to investigate the contribution of local inconsistency to difficulty.

2.2 Weight of the NP-Hard Component

The Fourier transform (FT) is a tool for decomposing functions into sums of orthogonal components. Beyond its classical use for real-valued functions, the FT has been extended to functions over arbitrary finite groups and applied to decompose objective functions in COPs [7,25]. This perspective provides a unified framework for analyzing different optimization problems through a common set of mathematical components. In particular, any LOP instance can be decomposed into two non-trivial Fourier coefficients, namely $\hat{f}_{\rho_{(n-1,1)}}$ and $\hat{f}_{\rho_{(n-2,1,1)}}$ (see [9] and [11] for further details). As a consequence, the objective function of a given LOP instance can be split into two subfunctions, each depending exclusively on one of these Fourier coefficients, denoted as $f_{\rho_{(n-1,1)}}$ and $f_{\rho_{(n-2,1,1)}}$, obtained via the inverse FT [1,17]. Moreover, [10] shows that each subfunction defines itself an LOP objective function, with fundamentally different computational properties: one corresponds to a polynomial-time solvable (P) problem, while the other is associated with an NP-hard optimization problem:

$$f(\sigma) = f_{\rho_{(n-1,1)}}(\sigma) + f_{\rho_{(n-2,1,1)}}(\sigma) + \bar{f} = f_P(\sigma) + f_{NP}(\sigma) + \bar{f},$$

for every $\sigma \in \Sigma_n$, where $\bar{f}$ is the mean objective function value derived from the trivial representation [1,17]. From this perspective, instance generation can be naturally approached by manipulating the relative weight of each component, as proposed by Elorza et al. in [10]. Conversely, for a given instance, the contribution of each component can be recovered through its Fourier decomposition, as demonstrated by Benavides et al. in [1]. More precisely, these latter authors presented in [2], for the more general class of k-adic Assignment Problems, a form of quantification of the weight of each component, by means of the variance. In the particular case of the LOP, this reduces to

$$W_{\mathrm{P}} = \frac{\mathrm{Var}(f_P)}{\mathrm{Var}(f)}, \quad W_{\mathrm{NP}} = \frac{\mathrm{Var}(f_{NP})}{\mathrm{Var}(f)}, \tag{5}$$

where W_{P} and W_{NP} denote the relative weight of the P and NP components, respectively. [5]

3 Experimental Study

In this section, we present a structural-controlled instance generation method, based on inducing a controlled number of 3-cycles in each instance. For every generated instance, we compute two structural-complexity indicators: (i) the density of directed 3-cycles in the associated digraph, and (ii) the relative weight of the NP-hard component of the objective function, obtained via its Fourier decomposition by the formula presented in Equation (5). These measures provide complementary information on structural properties, from graph-theoretical and spectral perspectives, respectively, that are expected to influence instance difficulty. We then analyze the distribution of the generated instances within the two-dimensional space defined by both indicators.

3.1 Instance Generation

In order to generate LOP instances with controllable inconsistencies, we adopt a constructive procedure based on perturbations of a consistent ordering. Specifically, we start by sampling a uniformly random permutation σ of the vertex set $V = \{1, \ldots, n\}$. This permutation defines a consistent instance matrix $A = [a_{ij}]_{i,j \in V}$, where for each $i, j \in V$ the preference is set so that $a_{ij} > a_{ji}$ if and only if vertex i appears before vertex j in σ. By construction, this yields an acyclic preference tournament structure whose optimal solution coincides with σ. Cycles are then generated by independently flipping pairwise preferences. More precisely, for each $i, j \in V$, the values a_{ij} and a_{ji} are exchanged with probability $\alpha \in [0, 1]$. Thus, α represents the probability of reversing a given pairwise preference with respect to the underlying permutation.

The number of 3-cycles induced by values of $\alpha > 0.5$ are equivalent to that induced by $1-\alpha$, as reversing a majority of pairwise preferences yields a structure analogous to preserving a minority of them, although the corresponding optimal solution may differ. Consequently, by symmetry of the structural properties, we restrict α to the interval $[0, 0.5]$ without loss of generality (this symmetry is illustrated in Figure 2). It should be borne in mind that α does not represent the actual density of 3-cycles in the instance. Rather, it acts as a control parameter that is empirically observed to be strongly correlated with 3-cycle density, as illustrated in Figure 2. Within the restricted interval $[0, 0.5]$, small values of α produce instances that are close to acyclic, whereas larger values induce a higher proportion of reversed directions. Accordingly, the density of 3-cycles reaches its expected maximum at $1/4$, since each pair of nodes has two possible orientations, resulting in eight possible configurations over three nodes of which only two form a directed 3-cycle.

[5] The variance can be computed explicitly without evaluating all objective values, by exploiting the structure of permutations and counting contributions of matrix entries according to their index configurations.

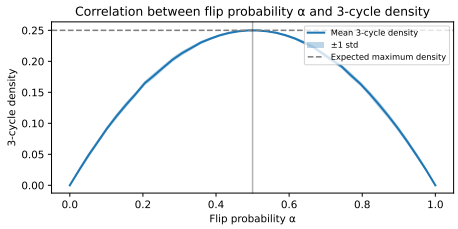


Fig. 2: Average 3-cycle density (blue curve) as a function of the flipping probability α for instances with $n = 100$, computed over 50 independent replications per value of α. Shaded light blue region indicates ± 1 standard deviation; the dotted line shows the expected maximum.

3.2 Instance Distribution among the Indicators Space

Using the instance generation procedure described above, we evaluate the distribution of the resulting instances among the two-dimensional space defined by previously introduced indicators. To this end, we consider values of $\alpha \in [0, 0.5]$ and partition this interval into disjoint regions of width 0.1. For each region, we generate 1000 instances by sampling α uniformly at random within the corresponding interval. For every generated instance, we compute two structural-complexity indicators: (i) the density of directed 3-cycles in the associated digraph, and (ii) the relative weight of the NP-hard component of the objective function, obtained via its Fourier decomposition by the formula presented in Equation (5). We include three different neighborhoods to conduct an initial exploratory study of whether certain structural configurations may favor specific neighborhood structures. For this purpose, we apply a best-improvement local search using three different neighborhood structures: insert, swap and reverse [22,26]. Each algorithm is run in a multistart setting with 25 independent starts per instance, and the best solution obtained is retained. To ensure a fair comparison between neighborhoods, all three local search methods are initialized from the same random starting solution at each multistart iteration.

Figure 3 illustrates the best-performing neighborhoods as a function of the 3-cycle density (x-axis) and the weight of the NP-hard component of the instance (y-axis), for different sizes of instances ($n = 8, 20, 100$). For each instance size, a separate set of plots is shown, one for each neighborhood that achieves the best performance. In each plot, instances for which the corresponding neighborhood is optimal are highlighted in color, while the remaining instances are shown in gray, allowing a clear visual identification of the regions associated with each neighborhood. Additionally, each plot reports the percentage of instances for which the corresponding neighborhood achieves the best performance.

From the graphical analysis, several conclusions can be drawn. First, the insert neighborhood clearly outperforms the others overall. Moreover, as the instance size increases, the proportion of instances for which insert is the best neighborhood also increases, reaching 99.7% for $n = 100$. At the same time,

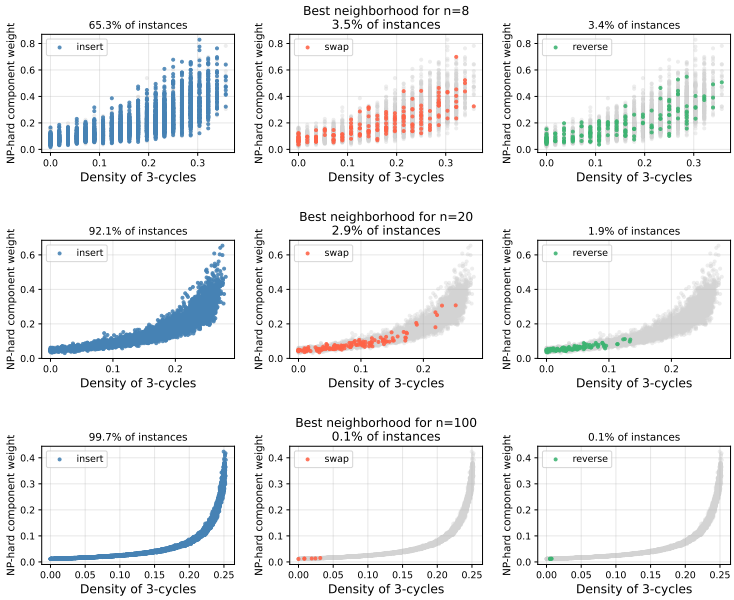


Fig. 3: Distribution of instances for which a single neighborhood achieves the best performance in the indicators space defined by the 3-cycle density (x-axis) and the NP-hard component weight (y-axis). Colored points highlight instances with a unique best neighborhood, while the remaining instances are shown in gray.

the regions where the swap and reverse neighborhoods outperform become increasingly restricted to instances with low 3-cycle density and low NP-hard component weight, suggesting that these neighborhoods are primarily effective for structurally simpler instances. Additionally, a noticeable difference can be observed in the distribution of instances for $n = 8$ compared to larger sizes, which is due to the limited number of possible 3-cycles in small instances. It should also be noted that the percentages of instances shown do not sum to 100%. This is because ties between neighborhoods are also taken into account, as illustrated in Figure 4. Nevertheless, similar patterns are observed: for small instance sizes, tie instances are distributed across the entire space, whereas for larger sizes they tend to concentrate in the region corresponding to simpler instances.

Common Benchmark Instances

We analyze the location of commonly used benchmark instances within the indicators space by computing the respective indicator values of instances from the LOLIB library [21] and on uniformly randomly generated instances (see Figure 5). The gray points represent instances of size $n = 100$ generated in the previous experiments, used as a reference distribution (although LOLIB instances have varying sizes, the observed trend appears to gradually approach that of

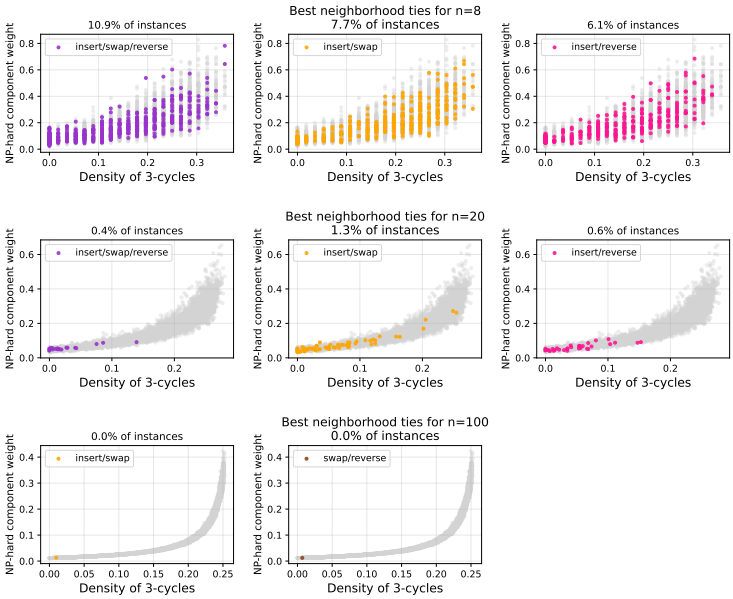
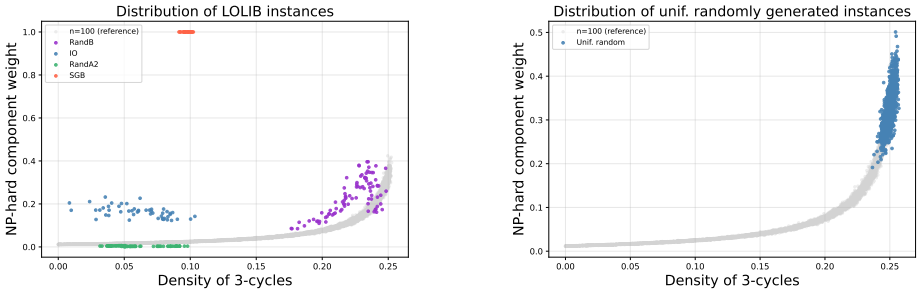


Fig. 4: Distribution of tie instances (where multiple neighborhoods achieve equivalent performance) in the indicators space defined by 3-cycle density (x-axis) and NP-hard component weight (y-axis). Colored points represent instances with ties, while the remaining instances are shown in gray.

the generated instances of size $n = 100$ as the problem size increases). Different colors represent distinct classes of LOLIB and uniformly generated instances.

On the one hand, Figure 5(a) shows that instances from each LOLIB class are concentrated in specific regions of the space, which do not align with the reference curve (in gray) corresponding to generated instances of size $n = 100$. In particular, both IO (blue) and SGB (red) instances originate from real-world data [21]. IO instances, derived from input-output economic matrices, exhibit strong and consistent structural patterns, which explains their low density of 3-cycles and the relatively small weight of the NP-hard component. In contrast, SGB instances display a more irregular structure, often leading to a large number of ties in the objective function values. These ties give rise to weak or ambiguous preferences, so that the interpretation of the LOP as an ordering of consistent preferences becomes less appropriate, and the resulting instances tend to be more challenging to solve. Conversely, RandB (purple) and RandA2 (green) classes contain randomly generated instances widely used for benchmarking purposes [20]. RandA2 instances exhibit a relatively structured form of randomness, leading to low densities of 3-cycles and a small contribution of the NP-hard component [1]. In fact, according to the structural conditions characterizing polynomial-time solvable LOP instances [10], such instances are necessarily acyclic, which is consistent with the near-transitive nature observed in

RandA2 (low density of 3-cycles). In contrast, RandB instances display higher levels of structural inconsistency, resulting in a broader dispersion across the space. On the other hand, instances of size $n = 100$ generated uniformly at random (Figure 5(b)) occupy a region corresponding to high 3-cycle density, closely following the reference curve, as expected under a uniform distribution of 3-cycles. These observations highlight the main contribution of our approach: the proposed structural-controlled instance generation method produces many instances located in previously underrepresented regions of the space, thereby providing a richer and more diverse benchmark set.



(a) (b)

Fig. 5: Distribution of common benchmark instances in the space defined by the 3-cycle density (x-axis) and the NP-hard component weight (y-axis). Gray points correspond to the reference generated instances distribution ($n = 100$), while colored points represent (a) instances from the LOLIB library and (b) $n = 100$ size uniformly randomly generated instances.

4 Conclusions and Future Work

In this paper, we propose a novel instance generation method for the LOP that enables the creation of instances with controlled number of 3-cycles. Together with the NP-hard component weight derived from the Fourier transform, these features define a two-dimensional structural-complexity space. We analyzed both generated and benchmark instances, as well as the performance of several neighborhoods. Results show that the insert neighborhood generally dominates, limiting the clear algorithm-instance associations. Nevertheless, the main contribution lies in demonstrating that structural-complexity can be effectively controlled. The proposed generation method produces instances spanning diverse and previously underrepresented regions, enhancing benchmark diversity. Future work will explore additional structural features and develop a comprehensive difficulty profile combining structural and algorithm-dependent aspects.

Acknowledgments. Maialen Beristain-Cortés acknowledges a predoctoral grant from the Basque Government (PRE_2025_1_0158). This work was also supported by the

Basque Government (IT2109-26, BERC 2026–2029 program, project KK-2026/00055 through Elkartek program) and the Spanish Ministry of Science, Innovation and Universities (projects PID2022-137442NB-I00 and PID2023-149195NB-I00, BCAM Severo Ochoa accreditation CEX2021-001142-S/MICIN/AEI/10.13039/501100011033).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Benavides, X., Hernando, L., Ceberio, J., Lozano, J.A.: P-NP Instance Decomposition Based on the Fourier Transform for Solving the Linear Ordering Problem. Evolutionary Computation **33**(3), 395–423 (Sep 2025). https://doi.org/10.1162/evco_a_00368
2. Benavides, X., Hernando, L., Ceberio, J., Lozano, J.A.: Fourier Transform-based instance decomposition for k-adic Assignment Problems. IEEE Transactions on Evolutionary Computation pp. 1–1 (2026). https://doi.org/10.1109/TEVC.2026.3674527
3. Ceberio, J., Hernando, L., Mendiburu, A., Lozano, J.A.: Understanding Instance Complexity in the Linear Ordering Problem. In: Intelligent Data Engineering and Automated Learning – IDEAL 2013, vol. 8206, pp. 479–486. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41278-3_58
4. Ceberio, J., Mendiburu, A., Lozano, J.A.: The linear ordering problem revisited. European Journal of Operational Research **241**(3), 686–696 (Mar 2015). https://doi.org/10.1016/j.ejor.2014.09.041
5. Charon, I., Hudry, O.: An updated survey on the linear ordering problem for weighted or unweighted tournaments. Annals of Operations Research **175**(1), 107–158 (Mar 2010). https://doi.org/10.1007/s10479-009-0648-7
6. Chicano, F., Daolio, F., Ochoa, G., Vérel, S., Tomassini, M., Alba, E.: Local Optima Networks, Landscape Autocorrelation and Heuristic Search Performance. In: Parallel Problem Solving from Nature - PPSN XII, vol. 7492, pp. 337–347. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32964-7_34
7. Diaconis, P., Rockmore, D.: Efficient computation of the Fourier transform on finite groups. Journal of the American Mathematical Society **3**(2), 297–332 (1990). https://doi.org/10.1090/S0894-0347-1990-1030655-4
8. Diestel, R., Diestel, R.: Graph Theory. No. 173 in Graduate Texts in Mathematics, Springer, New York, NY, 2. ed edn. (2000)
9. Elorza, A., Benavides, X., Ceberio, J., Hernando, L., Lozano, J.A.: Transforming Combinatorial Optimization Problems in Fourier Space: Consequences and Uses. IEEE Transactions on Evolutionary Computation **29**(4), 977–989 (Aug 2025). https://doi.org/10.1109/TEVC.2024.3457268
10. Elorza, A., Hernando, L., Lozano, J.A.: Transitions from P to NP-hardness: The case of the Linear Ordering Problem. In: 2022 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE, Padua, Italy (Jul 2022). https://doi.org/10.1109/CEC55065.2022.9870392
11. Elorza, A., Hernando, L., Lozano, J.A.: Characterizing Permutation-Based Combinatorial Optimization Problems in Fourier Space. Evolutionary Computation **31**(3), 163–199 (Sep 2023). https://doi.org/10.1162/evco_a_00315

12. Glover, F., Klastorin, T., Klingman, D.: Optimal Weighted Ancestry Relationships. Management Science **20**(8,), 1190–1193 (1974)
13. Goodman, A.W.: On Sets of Acquaintances and Strangers at any Party. The American Mathematical Monthly **66**(9), 778–783 (Nov 1959). https://doi.org/10.1080/00029890.1959.11989408
14. Grzesik, A., Král', D., Lovász, L.M., Volec, J.: Cycles of a given length in tournaments. Journal of Combinatorial Theory, Series B **158**, 117–145 (Jan 2023). https://doi.org/10.1016/j.jctb.2022.07.007
15. Hernando, L., Mendiburu, A., Lozano, J.A.: Generating Customized Landscapes in Permutation-Based Combinatorial Optimization Problems. In: Nicosia, G., Pardalos, P. (eds.) Learning and Intelligent Optimization, vol. 7997, pp. 299–303. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-44973-4_33
16. Hernando, L., Mendiburu, A., Lozano, J.A.: A Tunable Generator of Instances of Permutation-Based Combinatorial Optimization Problems. IEEE Transactions on Evolutionary Computation **20**(2), 165–179 (Apr 2016). https://doi.org/10.1109/TEVC.2015.2433680
17. Huang, J., Guestrin, C., Guibas, L.: Fourier Theoretic Probabilistic Inference over Permutations. Journal of Machine Learning Research **10**, 997–1070 (2009)
18. Leontief, W.: Input-Output Economics. Oxford University Press, New York, 2nd ed edn. (1986)
19. Lin, H.: Research on Minimum Feedback Arc Set Problem. Highlights in Science, Engineering and Technology **134** (2025). https://doi.org/10.54097/az8w1s78
20. Martí, R., Reinelt, G.: The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization, Applied Mathematical Sciences, vol. 175. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-16729-4
21. Martí, R., Reinelt, G., Duarte, A.: A benchmark library and a comparison of heuristic methods for the linear ordering problem. Computational Optimization and Applications **51**(3), 1297–1317 (Apr 2012). https://doi.org/10.1007/s10589-010-9384-9
22. Schiavinotto, T., Stützle, T.: The Linear Ordering Problem: Instances, Search Space Analysis and Algorithms. Journal of Mathematical Modelling and Algorithms **3**(4), 367–402 (Dec 2004). https://doi.org/10.1023/B:JMMA.0000049426.06305.d8
23. Simon, H.A., Newell, A.: Heuristic Problem Solving: The Next Advance in Operations Research. Operations Research **6**(1), 1–10 (Feb 1958). https://doi.org/10.1287/opre.6.1.1
24. Smith-Miles, K., Lopes, L.: Measuring instance difficulty for combinatorial optimization problems. Computers & Operations Research **39**(5), 875–889 (May 2012). https://doi.org/10.1016/j.cor.2011.07.006
25. Terras, A.: Fourier Analysis on Finite Groups and Applications. No. 43 in London Mathematical Society Student Texts, Cambridge University Press, Cambridge, repr., transferred to digital print edn. (1999)
26. Torralbo, M., Hernando, L., Contreras-Torres, E., Lozano, J.A.: On the Use of Second Order Neighbors to Escape from Local Optima. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 375–383. ACM, Lisbon Portugal (Jul 2023). https://doi.org/10.1145/3583131.3590473
27. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (Apr 1997). https://doi.org/10.1109/4235.585893