Understanding Instance Complexity in the Linear Ordering Problem

Josu Ceberio, Leticia Hernando, Alexander Mendiburu, and Jose A. Lozano

Intelligent Systems Group,
Department of Computer Science and Artificial Intelligence
University of the Basque Country UPV/EHU
Donostia, 20018 Gipuzkoa, Spain
{josu.ceberio,leticia.hernando,alexander.mendiburu,ja.lozano}@ehu.es

Abstract. The Linear Ordering Problem is a combinatorial optimization problem which has been frequently addressed in the literature due to its numerous applications in diverse fields. In spite of its popularity, little is known about its complexity. In this paper we analyze the linear ordering problem trying to identify features or characteristics of the instances that can provide useful insights into the difficulty of solving them. Particularly, we introduce two different metrics, *insert ratio* and *ubiquity ratio*, that measure the difficulty of solving the LOP with local search type algorithms with the insert neighborhood system. Conducted experiments demonstrate that the proposed metrics clearly correlate with the complexity of solving the LOP with a multistart local search algorithm.

Keywords: linear ordering problem, instance, complexity, local optima.

1 Introduction

The Linear Ordering Problem (LOP) is a classical combinatorial optimization problem which has received the attention of the research community since it was studied for the first time by Chenery and Watanabe [1]. Due to its numerous applications in diverse fields such as archeology [4], economics [7] or mathematical psychology [6], a wide variety of optimization strategies have been proposed in the literature. As proof of this, Marti and Reinelt [8] presented a review of the most successful exact and heuristic algorithms, including branch and bound, constructive heuristics, local searches or variable neighborhood search. Garey and Johnson [3] demonstrated that the LOP is a *NP-hard* problem, which means that there is no known algorithm able to solve up to the optimality all LOP instances in polynomial time. However, as seen for most of the combinatorial optimization problems, the difficulty of solving the instances varies with the size of the instance, and other additional unknown parameters. In the particular case of the LOP, one can easily find instances of large size that are easier to solve than other of smaller size for a wide set of algorithms.

In this regard, the community has also tried to extract characteristics from the instances that could be used to measure their difficulty or to provide additional

H. Yin et al. (Eds.): IDEAL 2013, LNCS 8206, pp. 479–486, 2013.

information to the algorithm used for solving it. For instance, Schiavinotto and Stutzle [9] analyzed LOP instances from different benchmarks, extracting features that later were used to rank the benchmarks according to their difficulty, and proposed new algorithms for optimizing the LOP. However, we think that the properties they studied could be further developed.

In this work, we investigate this direction trying to identify features or characteristics of the LOP instances that influence the complexity of solving LOP instances with local search type algorithms. Particularly, we introduce two different metrics: *insert ratio* and *ubiquity ratio* that give a measure of the difficulty of achieving the optimal solution with an insert neighborhood system.

The rest of the paper is organized as follows: in the following section, the definition of the LOP is given. In Section 3, the instance complexity of the LOP is studied by introducing the insert ratio and ubiquity ratio. Next, in Section 4, some experiments are run in order to evaluate the validity of the proposed metrics. Finally, some conclusions and ideas for future work are presented in Section 5.

2 The Linear Ordering Problem

Given a matrix $B = [b_{ij}]_{n \times n}$ of weights, the LOP consists of finding a simultaneous permutation σ of the rows and columns of B, such that the sum of the weights above the main diagonal is maximized. The equation below formalizes the LOP function:

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

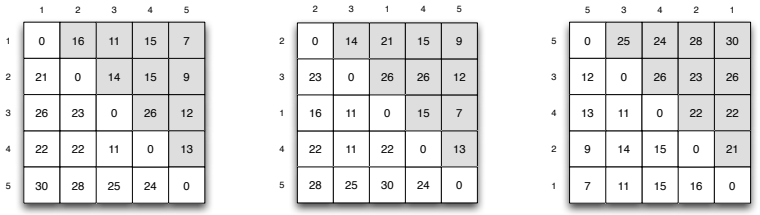
where σ_i denotes the index of the row (and column) ranked in position i in σ.

Let us consider an example of 5×5 LOP instance (see Fig. 1). The initial matrix is represented by the identity permutation $\sigma = (1, 2, 3, 4, 5)$ (see Fig. 1a), and the associated fitness $f(\sigma)$ is 138. The solution $\sigma' = (2, 3, 1, 4, 5)$ introduces a different ordering of the indices that provide a better solution than σ (see Fig. 1b). The optimal solution for this example is given by $\sigma^* = (5, 3, 4, 2, 1)$ (see Fig. 1c).

For the sake of studying the LOP function, we analyze separately the contribution of each index of the solution to the objective function. When an index j is ranked in position i, $\sigma_i = j$, the contribution of the index j to the objective function is given by the sum of weights of the column j in the rows $\sigma_1, \ldots, \sigma_{i-1}$ and weights of the row j in the columns $\sigma_{i+1}, \ldots, \sigma_n$. Formally it is expressed as

$$f_\sigma(\sigma_i) = \sum_{z=1}^{i-1} b_{\sigma_z \sigma_i} + \sum_{z=i+1}^{n} b_{\sigma_i \sigma_z}$$

Note that the rank i indicates only the number of weights of the corresponding column $\{1, \ldots, i-1\}$ and row $\{i+1, \ldots, n\}$ that will be considered in the sum. The specific row and column weights come given by the previously ranked $i-1$



	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

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

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

(b) $\sigma' = (2, 3, 1, 4, 5)$ $f(\sigma') = 158$.

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

(c) $\sigma^* = (5, 3, 4, 2, 1)$ $f(\sigma^*) = 247$.

Fig. 1. Example of a 5×5 LOP instance. The configuration of the matrix of weights for three different solutions is described.

indices $\{\sigma_i, \ldots, \sigma_{i-1}\}$ and the posterior $n-i+1$ indices $\{\sigma_{i+1}, \ldots, \sigma_n\}$. However, note that the contribution of an index is independent to the ordering of the previously ranked indices and to the posterior indices.

Following the previous example, the contribution of the index 4 in σ (see Fig. 1a) consist of the sum of the weights associated to this index in the upper triangle (15+15+26+13). If we check the contribution of the index 4 in σ' (see Fig. 1b), we see that the contribution of the index remains equal regardless of what the ordering of the indices is in the previous $\{1, 2, 3\}$ positions and in the posterior positions $\{5\}$.

3 Analysis of Complexity

As previously mentioned, in this work we aim to study the properties of the instances that affect the complexity of local search type algorithms. Since many works in the literature [9,2] clearly state that the insert neighborhood is the most appropriate system for solving the LOP with a local search algorithm, our study will be carried out for this neighborhood.

In the following lines, the *insert ratio* and *ubiquity ratio* are introduced in detail.

3.1 Insert Ratio

The insert ratio measures the intercalation between the weights of the column and of the row associated to an index. It is calculated as follows:

1. Put all row and column weights in a vector and order these numbers.
2. For each pair of consecutive weights belonging to the column, calculate the distance
3. Repeat the same process as before for the weights of the row.
4. The insert ratio is calculated by summing the obtained distances, and dividing by $n - 1$.

The overall insert ratio of the instance is calculated by averaging the insert ratio of each of the indices. Fig. 2 illustrates the insert ratio calculated for the index 4 of the matrix.

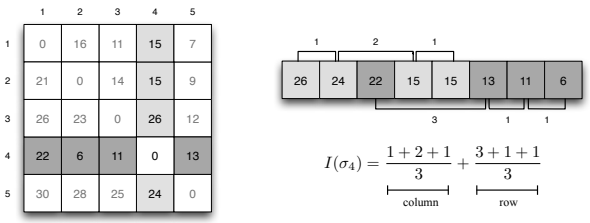


Fig. 2. Insert ratio of the index σ_4

Given a specific index i, when all the weights of the column i are higher than the weights of the row i, the index i can only generate local optima in the last position of σ, and only in the first position when the opposite scenario occurs. The indices that agree with the exposed cases receive very low insert ratios, suggesting that this type of indices are easy to rank. When the overall insert ratio associated to an instance is low, the instance is supposed to be easy to solve. On the contrary, a high insert ratio, suggests that the indices are more susceptible to generate local optima in many ranks of σ, and thus, the difficulty of the instance is higher.

3.2 Ubiquity Ratio

The ubiquity ratio measures the percentage of locations where the indices are susceptible to generate local optima using the insert neighborhood. An index may generate local optima in a certain position if all the possible insert operations over the index decreases its contribution. Note that for each index i, the weights b_{ij} and b_{ji} for $j = \{1, \ldots, n\}$ appear always in symmetric positions with respect to the main diagonal, i.e. they never appear together in the upper triangle. Therefore, an index i may generate a local optima when ranked in position j if there exists an ordering σ such that the following two constraints are true.

$$\sum_{z=1}^{j-1} b_{\sigma_z,\sigma_j} - b_{\sigma_j,\sigma_z} > 0 \wedge \sum_{z=j+1}^{n} b_{\sigma_z,\sigma_j} - b_{\sigma_j,\sigma_z} < 0$$

In order to determine the ranks where the indices may generate local optima, we count the positions where the index ranked in position j cannot generate a local optima, that is to check whether the sum of all the differences in positions $\{1, \ldots, j-1\}$ is negative and the sum of all the differences in positions $\{j+1, \ldots, n\}$ is positive. The ubiquity ratio of an index is calculated by dividing the number of positions where the index may generate local optima, with the total

number of positions. The global ubiquity ratio is then calculated by averaging the ratios of each of the indices. Fig. 3 illustrates the procedure of calculating the ubiquity ratio of the index 4.

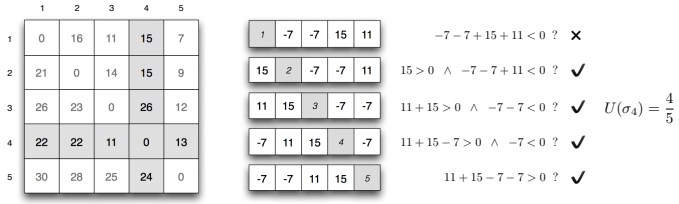


Fig. 3. Ubiquity ratio of the index σ_4

4 Experimentation

In order to confirm the validity of the metrics proposed, we introduce two different set of experiments. Additionally, two artificially generated benchmarks of instances are used.

Benchmarks A and B of instances. 540 instances of sizes $\{8, 9, 10, 11, 12, 13\}$ (benchmark A) and 600 instances of sizes $\{20, 30, 40, 50, 60, 70\}$ (benchmark B) have been generated following the next procedure. We start by generating the weights associated to the first index, then the second and so on. Note that when we generate the weights for the first index, we have also indirectly generated some of the weights for the rest of the indices, and the same for the second index and so on. In order to generate the weights for index i, we uniformly at random sample a vector of $2(n-i-1)$ weights in the range [0,999]. Next, we order these weights. After that, following the order of the weights, we make groups of size t (t is a parameter of the procedure that ranges from 0 to 10). Then, we randomly decide where, column or row, to place each pair of groups (one to the row and the other to the column).

Experiment 1. We analyze the correlation of the insert and ubiquity ratios with respect to the estimated number of local optima for the benchmark B of LOP instances. Particularly, the benchmark B of instances is used. Following the recommendations of a recently published review on local optima estimation methods [5], we have chosen two methods to estimate the number of local optima of the instances: *ChaoBunge* and *ChaoLee2*.

Fig. 4 and 5 introduce the results for the proposed metrics according to the average of 10 repetitions of *ChaoLee2* and *ChaoBunge*. Results show a good correlation between the *ubiquity* and *insert* ratios and the number of local optima. In fact, the higher the value of the metric, the higher the number of local optima.

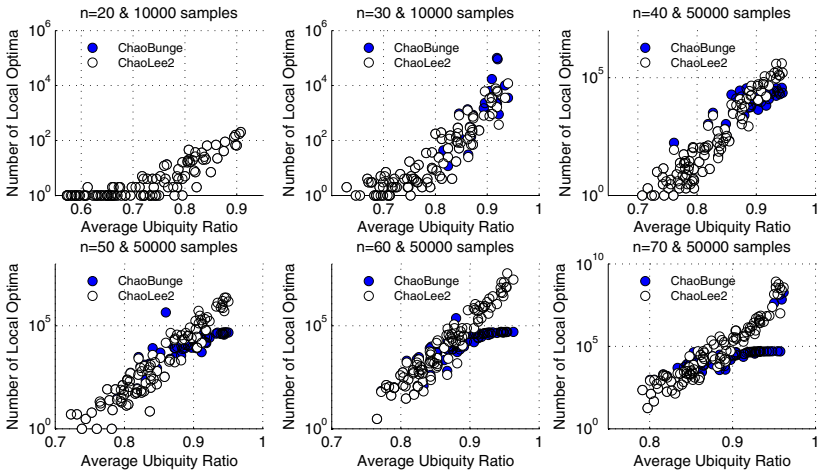


Fig. 4. The estimated number of local optima related to the average *ubiquity* ratio of the instance.

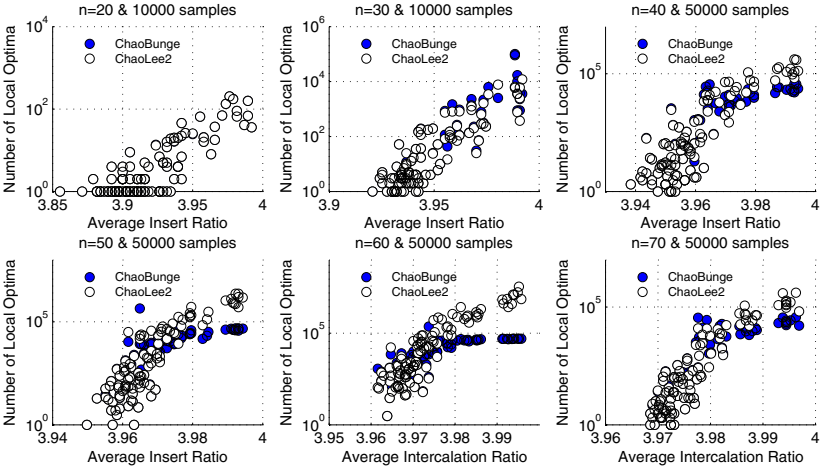


Fig. 5. The estimated number of local optima related to the average *insert* ratio of the instance

Experiment 2. In this experiment, we run a muti-start local search greedy algorithm (MLS) with the insert neighborhood, and we compute the number of evaluations needed to achieve the optimal solution of the benchmark A instances.

Fig. 6 and 7 introduce the results for the proposed metrics according to the average of 10 repetitions of the MLS. Results show that the higher the *ubiquity* and *insert* ratios, the higher the number of evaluations performed. This behavior is especially evident for instances of size $\{11, 12, 13\}$.

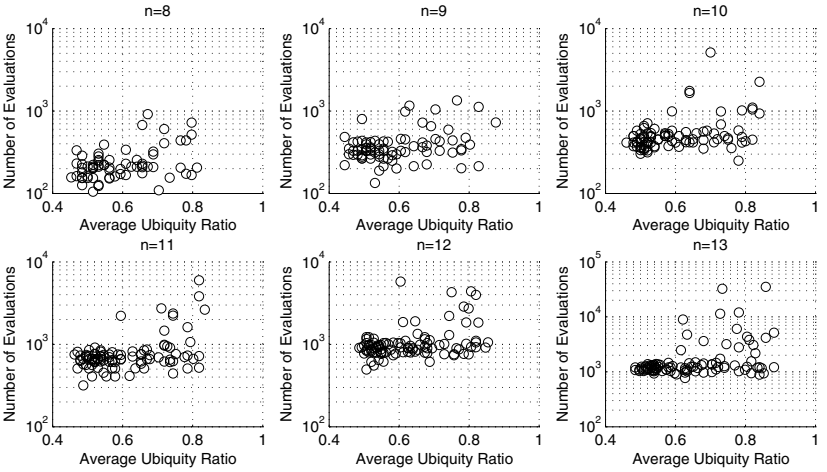


Fig. 6. The average number of evaluations performed by the MLS related to the average *ubiquity* ratio of the instance.

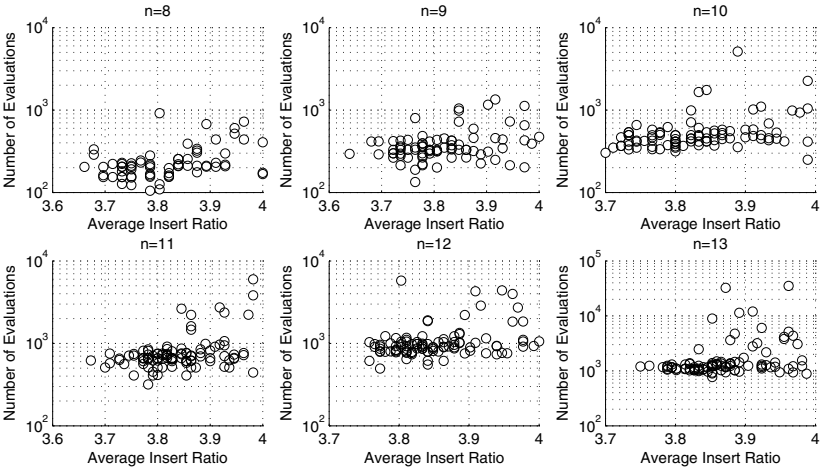


Fig. 7. The average number of evaluations performed by the MLS related to the average *insert* ratio of the instance

5 Conclusions and Future Work

In this paper, a preliminary analysis of the instance complexity of the LOP was presented. Particularly, we focused on identifying the characteristics of the instances that influence the complexity of solving them with local search type algorithms with the insert neighborhood. Characterizing the conditions required by each index in the solution to generate local optima for the insert neighborhood, we presented two new metrics, *insert ratio* and *ubiquity ratio*. Conducted

experiments showed a correlation between the proposed metrics and the complexity of solving an artificially generated set of LOP instances with multistart local search algorithms.

As future work, it would be interesting to extend the proposed metrics with pairwise index relations or even larger ones. Alternatively, the influence of ties of pairs of weights to the number of local optima should be studied, which, at first sight, introduce a high redundancy in the search space.

Acknowledgments. This work has been partially supported by the Saiotek and Research Groups 2013-2018 (IT-609-13) programs (Basque Government), TIN2010-14931 (Ministry of Science and Technology), COMBIOMED network in computational bio-medicine (Carlos III Health Institute), and by the NICaiA Project PIRSES-GA-2009-247619 (European Commission). Josu Ceberio and Leticia Hernando hold a grant from Basque Government.

References

1. Chenery, H.B., Watanabe, T.: International comparisons of the structure of production. Econometrica 26(4), 487–521 (1958)
2. Garcia, C.G., Pérez-Brito, D., Campos, V., Martí, R.: Variable neighborhood search for the linear ordering problem. Comput. Oper. Res. 33(12) (2006)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
4. Glover, F., Klastorin, T., Klingman, D.: Optimal weighted ancestry relationships. Management science report series. University of Colorado (1972)
5. Hernando, L., Mendiburu, A., Lozano, J.A.: An evaluation of methods for estimating the number of local optima in combinatorial optimization problems. In: Evolutionary Computation (2013)
6. Kemeny, J.G.: Mathematics without numbers. Daedalus 88, 577–591 (1959)
7. Leontief, W.: Input-Output Economics. Cambridge University Press (2008)
8. Martí, R., Reinelt, G.: The linear ordering problem: exact and heuristic methods in combinatorial optimization, vol. 175. Springer (2011)
9. Schiavinotto, T., Stützle, T.: The linear ordering problem: instances, search space analysis and algorithms. Journal of Math. Modelling and Algorithms (2004)