Are the artificially generated instances uniform in terms of difficulty?

Aritz Perez*, Josu Ceberio† and Jose A. Lozano*†
*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, ja.lozano@ehu.eus

***Abstract*—In the field of evolutionary computation, it is usual to generate artificial benchmarks of instances that are used as a test-bed to determine the performance of the algorithms at hand. In this context, a recent work on permutation problems analyzed the implications of generating instances uniformly at random (u.a.r.) when building those benchmarks. Particularly, the authors analyzed instances as rankings of the solutions of the search space sorted according to their objective function value. Thus, two instances are considered equivalent when their objective functions induce the same ranking over the search space. Based on the analysis, they suggested that, when some restrictions hold, the probability to create easy rankings is higher than creating difficult ones.**

In this paper, we continue on that research line by adopting the framework of local search algorithms with the best improvement criterion. Particularly, we empirically analyze, in terms of difficulty, the instances (rankings) created u.a.r. of three popular problems: Linear Ordering Problem, Quadratic Assignment Problem and Flowshop Scheduling Problem. As the neighborhood system is critical for the performance of local search algorithms three different neighborhood systems have been considered: swap, interchange and insert. Conducted experiments reveal that (1) by sampling the parameters uniformly at random we obtain instances with a non-uniform distribution 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 smoothness of the landscape induced by the neighborhood and on its size.

I. Introduction

In the field of evolutionary computation, it is common to use benchmarks of instances of a given problem in order to carry out a performance evaluation of existing and newly developed algorithms. When the final goal is to solve a specific real-world problem, real instances are used to perform 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 to evaluate the efficiency of the proposed algorithm under different scenarios.

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 the space of instances/problems/functions.

Recently, Ceberio et al. [6] showed that such equivalence does not hold for instances of the Linear Ordering Problem. Authors revealed that if the algorithm takes into account the objective value of solutions explicitly throughout the optimization process, then the space of functions is infinite. However, when the algorithms progress by making considerations of the type better-equal-worse (ordering type), then the number of functions that the algorithm distinguishes is finite, and, therefore, they can be grouped in terms of rankings. In the Linear Ordering Problem, authors concluded that (1) by sampling parameters of instances u.a.r, some rankings are observed more frequently than others, and (2) for the linear ordering problem and $n = 3$, the most frequent rankings present only one local optimum.

In this paper, we continue that work by adopting the framework of local search algorithms (LS). LS algorithms [7] are one of the simplest yet effective families of optimization algorithms. LS algorithms progress by moving from one solution to a neighboring solution that improves their quality. Different LS algorithms can be considered depending on the way in which neighboring solutions are selected. Among the selection criteria *best improvement* (BI) is one of the most used heuristics and it lies in selecting the neighboring solution that produces the highest improvement to the objective value of the current solution. Clearly, BI heuristic depends strongly on the neighborhood structure considered and, given a neighborhood, its behavior can be locally defined in terms of the pairwise (partial) rankings among the neighboring solutions.

In this work, we carried out an extensive experimental study on three popular problems in which we generated large amounts of instances sampling the respective parameters u.a.r.: *Linear Ordering Problem* [8], [9], *Quadratic Assignment Problem* [10] and *Flowshop Scheduling Problem* [11]. We measured the difficulty of the generated instances for BI algorithms under three different neighborhood structures: *swap*, *interchange* and *insert*. We define the difficulty of an instance under a certain neighborhood as the probability of BI of not reaching the global optimum when starting from a solution selected u.a.r. Using the notion of difficulty, it is possible to group the instances into equivalence classes, and perform an empirical analysis of their distribution.

Conducted experiments point out that (1) by sampling the parameters uniformly at random, we obtain instances with a non-uniform distribution, (2) the distribution of the difficulty for a given neighborhood strongly depends on the problem considered, and (3) given a problem the behavior of a neighborhood in terms of the difficulty is closely related to the smoothness of the instances induced by the neighborhood and, also, to the size of the neighborhood.

The remainder of the paper is organized as follows: in Section II the necessary background on combinatorial optimization, rankings and local search is introduced. In addition, a difficulty measure is defined from a BI perspective. In Section III, the permutation problems and the neighborhood structures considered in the work are detailed and the concept of roughness of an instance induced by a given neighborhood is presented. Afterwards, the experimental study is presented in Section IV. Next, a discussion is carried out in Section V. Finally, Section VI summarizes the main results of the paper and the future work lines.

II. Background

In this section, we introduce the concepts that are necessary to understand the main contributions of the paper.

A. Combinatorial optimization problems

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*. In this work, we will analyze three popular COPs over permutations: Linear Ordering Problem, Quadratic Assignment Problem and Flowshop Scheduling Problem (see Section III-A)

The aim in a COP is to, given an instance, find $x \in \Omega$ such that f is maximized (or minimized). From here on, for the sake of simplicity, we will consider only injective objective functions. In addition, 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.

B. Instances as rankings of solutions

Definition 1: Let f be an instance of a COP. The *ranking* induced by f is defined as:

$$\begin{aligned} r : \quad & \Omega \to n! \\ & x \mapsto r(x) \end{aligned}$$

where for any $x, y \in \Omega$, $r(x) < r(y)$ if and only if $f(x) > f(y)$, n is the size of x, and $n!$ is the set of all permutations of size n.

The algorithms whose behavior is completely determined by the ranking of the solutions induced by a particular instance f are called *ranking based optimization algorithms*. In other words, RO progresses exclusively by taking into account the information contained in a ranking over the search space. For instance, evolutionary algorithms that use tournament or ranking selection operators [12], local search algorithms such as tabu search [13], variable neighborhood search [14], iterated local search [15] or greedy randomized adaptive search [16] are of this type.

It is worth noting that the behavior of a particular RO over instances that induce the same ranking is the same. In addition, a RO can have the same behavior for instances belonging to different ranking classes (see [6] for more details). From here on, we will concentrate on rankings rather than on particular instances.

C. Local search on rankings

LS algorithms 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^{Ω} represents the power set of the domain Ω. In this work, we analyze COPs over permutations using three of the most popular neighborhoods: swap, interchange and insert.

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 conditioned 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 with a lower ranking, the solution with the lowest ranking, while the first improvement progresses by selecting a neighboring solution with a lower ranking. Both algorithms are said to converge to a (local) optimum solution when all the neighboring solutions are of a higher rank.

D. Difficulty

Next, we define the difficulty of an instance as a function of BI under a certain neighborhood.

Definition 2: Let k be an instance of a COP, $\mathbf{P} = (\Omega, f)$, and N a neighborhood. The *easiness/difficulty* of k, under N, is defined as the probability of reaching/*not* reaching the global optimum solution by using BI and by picking a solution of Ω at random.

Clearly, the difficulty is 1 minus the easiness, and the easiness of an instance can be measured as the size of the basin of attraction of the global optimum divided by the size of the search space. Since the different grades of difficulty can be seen as equivalence classes for instances, we will analyze the instances grouped by their difficulty/easiness. It is worth to remark that while the number of possible instances (rankings) is $\mathcal{O}(n!!)$, according to the definition of difficulty introduced

above (in terms of attraction basin size), i) the number of difficulty classes is $\mathcal{O}(n!)$, and ii) the average size of the difficulty classes is $\mathcal{O}(n!)$.

III. Problems and Neighborhoods

In the following, we introduce one by one the combinatorial problems for which the experimental analysis is carried out, and the different neighborhoods considered for the BI heuristic.

A. Permutation problems

In this work, we have focused exclusively on three permutation-based combinatorial optimization problems. These problems have the particularity that their solutions are codified naturally as permutations. A permutation is understood as a bijection σ of the items $\{1,\ldots,n\}$ onto $\{1,\ldots,n\}$, where $\sigma(i)$ denotes the item at position i.

Despite such similarity, as we will see in the following, the information codified in each problem is totally different. It is worth remarking that in the respective research works, it has been proved that the three are NP-*hard* problems [17].

The first problem considered in the work is the Linear Ordering Problem (LOP) [8], [9]. Given a matrix $B = [b_{ij}]_{n\times n}$ of numerical entries, the LOP 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_{kl} are the parameters of the problem. A particular assignment of values to these parameters constitutes an instance.

The second problem included in the study is the permutation Flowshop Scheduling Problem (FSP) [11] where n jobs ($i = 1,\ldots,n$) have to be scheduled on m machines ($j = 1,\ldots,m$) in such a way that a given criterion is minimized. A job consists of m operations and the j^{th} operation of each job must be processed on machine j for a given specific processing time without interruption. The processing times are fixed, non-negative values and every job is available at time zero. At a given time, a job can start on the j^{th} machine when its $(j-1)^{th}$ operation has finished on machine $(j-1)$, and machine j is free. A solution for the problem is codified as a permutation σ of length n where $\sigma(i)$ denotes the job at position i.

As the optimisation criterion, we considered the total flow time (TFT), which optimises the sum of the completion times of each job. Eq. 2 formalizes, given a solution σ, the correspoding TFT. Note that $c_{\sigma(i),m}$ stands for the completion time of job $\sigma(i)$ on machine m.

$$f(\sigma) = \sum_{i=1}^{n} c_{\sigma(i),m} \tag{2}$$

Being $p_{\sigma(i),j}$ the processing time required by job $\sigma(i)$ on machine j, the completion time of job $\sigma(i)$ on machine j can be recursively calculated as:

$$c_{\sigma(i),j} = \begin{cases} p_{\sigma(i),j} & i = j = 1 \\ p_{\sigma(i),j} + c_{\sigma(i-1),j} & i > 1, j = 1 \\ p_{\sigma(i),j} + c_{\sigma(i),j-1} & i = 1, j > 1 \\ p_{\sigma(i),j} + \max\{c_{\sigma(i-1),j}, c_{\sigma(i),j-1}\} & i > 1, j > 1 \end{cases}$$

Finally, we included the Quadratic Assignment Problem (QAP) [10]. In this problem, the goal is to allocate a set of facilities to a set of locations, with a cost function associated to the distance and flow between the facilities. The objective is to assign each facility to a location such that the total cost is minimized. Specifically, given two $n \times n$ numerical matrices $H = [h_{ij}]$ and $D = [d_{ij}]$, where h_{ij} is the flow between facility i and facility j, and d_{ij} denotes the distance between the location i and j, the goal is to find a permutation σ (where $\sigma(i)$ represents the facility allocated at position i), such that the function

$$f(\sigma) = \sum_{i=1}^{n}\sum_{j=1}^{n} h_{\sigma(i)\sigma(j)} * d_{ij} \tag{3}$$

is minimized. A particular assignment of values to the $D = [d_{ij}]$ and $H = [h_{ij}]$ parameters constitutes an instance.

B. Neighborhood systems

In local search optimization, a wide range of neighborhood structures have been proposed in order to deal with combinatorial problems. In what follows, we will describe the three most relevant structures to deal with permutation problems: *adjacent interchange* (from now on *swap*), *interchange* and *insert*.

Let us consider a permutation σ of size n. In what follows, we define the neighborhood structures:

- *Swap*: considers as neighbors all the solutions that can be obtained by performing any exchange of two items in consecutive positions. The neighborhood includes all the solutions at a Kendall's τ distance one, and its size is $n-1$.
- *Interchange*: considers as neighbors all the solutions that can be obtained by performing any exchange of the items in any two positions i and j. The neighborhood includes all the solutions at Cayley's distance one, and its size is $\binom{n}{2}$.
- *Insert*: it 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. The neighborhood includes all the solutions at Ulam's distance one, and its size is $(n-1)^2$.

C. Smoothness

As mentioned in Section II-A, each COP consists of a domain and an objective function that is characterized by a set of parameters (called instance). When computing the objective value of a given solution σ, either in LOP, FSP or QAP, only

a subset of the parameters is involved (see Equations 1, 2 and 3). At this point, if we consider a given neighborhood system, we note that the number of parameters that differ between neighboring solutions can be relevant in order to describe the average smoothness of the fitness landscape generated. To this number, we call it the (parameter) roughness of a COP induced by a neighborhood. For the sake of intuition, given a neighborhood, as the roughness of a COP decreases, on average, the smoothness of the objective function with respect to the neighborhood system tends to increases. Table I summarizes the parameter roughness for each problem-neighborhood pair considered in [18].

TABLE I
THE PARAMETER ROUGHNESS FOR EACH PROBLEM UNDER DIFFERENT NEIGHBORHOODS

	Swap	Interchange	Insert
LOP	1	$2\|i-j\|-1$	$\|i-j\|$
FSP	$m(n-i)$	$m(n-\min(i,j))$	$m(n-\min(i,j))$
QAP	$4(n-1)$	$4(n-1)$	$2n(\|i-j\|+1)-(\|i-j\|+1)^2$

With illustration purposes, let us consider the LOP and the interchange neighborhood. In order to compute the objective-value of any given solution σ of the LOP, we need to perform a sum of $(n(n-1))/2$ b_{ij} parameters. At this point, we perform an interchange operation on σ with indices i and j. In order to calculate the objective value of the newly created solution, it is not necessary to sum again over $(n(n-1))/2$ parameters, but it is possible to apply a correction provoked by the interchange. For this particular problem-neighborhood, the correction consists of removing $2|i-j|-1$ specific parameters, and adding the same number of parameters not considered previously.

At this point, it is worth to remark that the roughness of different COPs is not comparable since they differ in the number of parameters and in the manner that the parameters are employed in the computation of the objective function (see Section III-A). Nevertheless, as seen in Section IV, the roughness of a COP under different neighborhoods can be useful to explain the changes in the distribution of the difficulty of its instances.

IV. EXPERIMENTAL STUDY

This section is devoted to analyzing empirically the relation between instances of LOP, FSP and QAP with parameters generated u.a.r. and their difficulty under swap, interchange and insert neighborhoods.

The parameters have been generated as follows:

- LOP: the b_{ij} parameters are sampled i.i.d. according to an uniform distribution in the interval $[0,1]$ for $i,j \in \{1,...,n\}$.
- QAP: the h_{ij} and d_{ij} parameters are sampled i.i.d. according to an uniform distribution in the interval $[0,1]$ for $i,j \in \{1,...,n\}$.
- FSP: $m=n$ and the p_{ij} parameters are sampled i.i.d. according to an uniform distribution in the interval $[0,1]$.

In addition, we have included an artificial COP that does not have an associated objective function, i.e., it does not depend on any parameter set (we call rand. COP, *RCOP*). Its instances are directly given in terms of rankings of solutions. For this artificial COP, the rankings have been generated u.a.r. Due to the absence of parameters, the roughness of RCOP can not be measured under any neighborhood. However, we presume that RCOP produces rough landscapes because the neighborhoods have been randomly generated.

Not being restricted, we have considered a randomized neighborhood called rand. insert with size $(n-1)^2$. This neighborhood has a neighborhood graph equivalent to the *insert* neighborhood where the nodes have been randomly relabeled. Note that, due to random relabeling, it is not possible to obtain roughness measures for LOP, FSP and QAP.

We sampled $5 \cdot 10^5$ instances for each COP and for $n \in \{3,4,5,6,7\}$[1] The results obtained are summarized in Figs. 1 and 2. Fig. 1 shows the distribution of instances in terms of their easiness (probability of success or one minus difficulty) for $n \in \{6,7\}$[2], while Fig. 2 shows the evolution of the mean and mode of the easiness for $n \in \{3,4,5,6,7\}$.

Observed results point out that, in general, instances generated u.a.r. do not follow an uniform distribution from the scope of difficulty. Moreover, its distribution varies with respect to the neighborhood system and, also, to the problem. For instance, in LOP 74.0% and 58.8% of the instances generated have a single optimum under insert neighborhood for $n=6$ and $n=7$, respectively (see Figure 1).

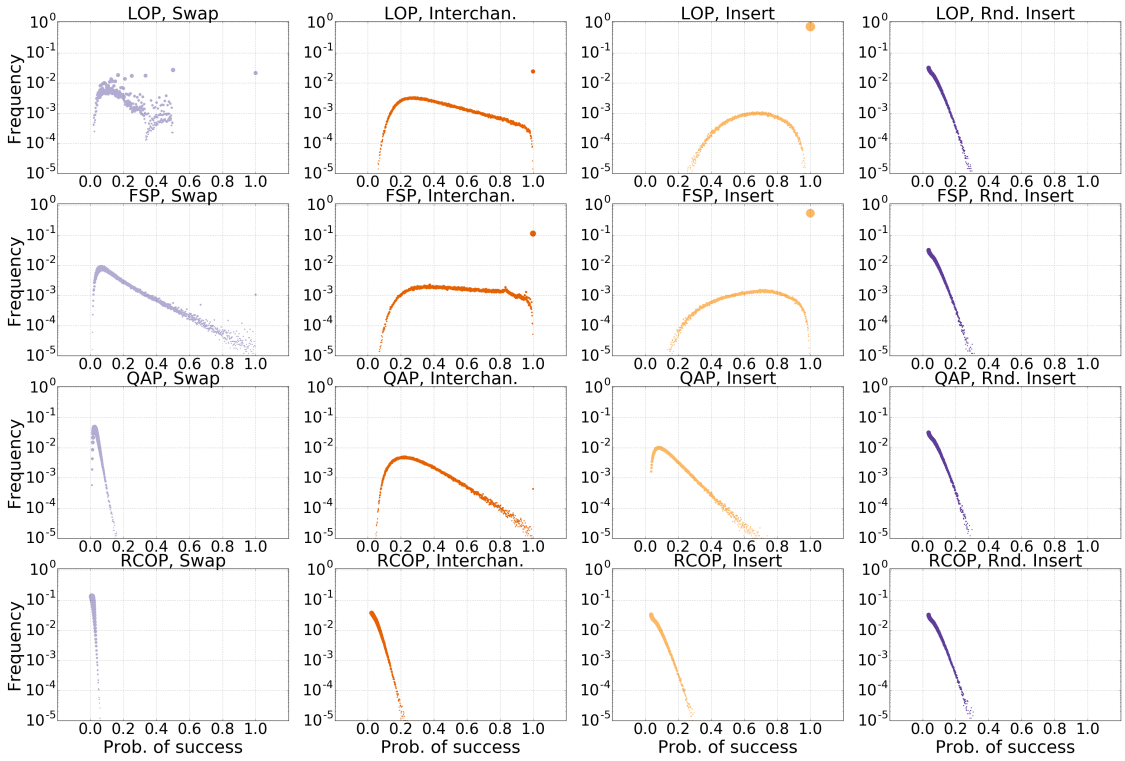
QAP generates the most difficult instances, on average, under the three neighborhoods (see Figure 2(a)). The instances generated for LOP and FSP problems are similar in terms of difficulty for the three neighborhoods (see Figure 1). The problems under the swap neighborhood are the most difficult. This is mainly motivated by the fact that it is the smallest neighborhood, with only $\mathcal{O}(n)$ neighbors, compared to the $\mathcal{O}(n^2)$ neighbors of interchange and insert. The difficulty of the LOP and FSP instances under the insert neighborhood is the lowest one, while for the QAP, the lowest difficulty is obtained under the interchange neighborhood.

As regards the scalability, it is possible to say that, on average, the difficulty of the instances for each problem and neighborhood increases with the size of the instance, n. However, in some cases the mode of the difficulty is 0 (easiness is 1), e.g., LOP and FSP with interchange and insert neighborhoods for $n \in \{3,4,5,6,7\}$ (see Figure 2(b)).

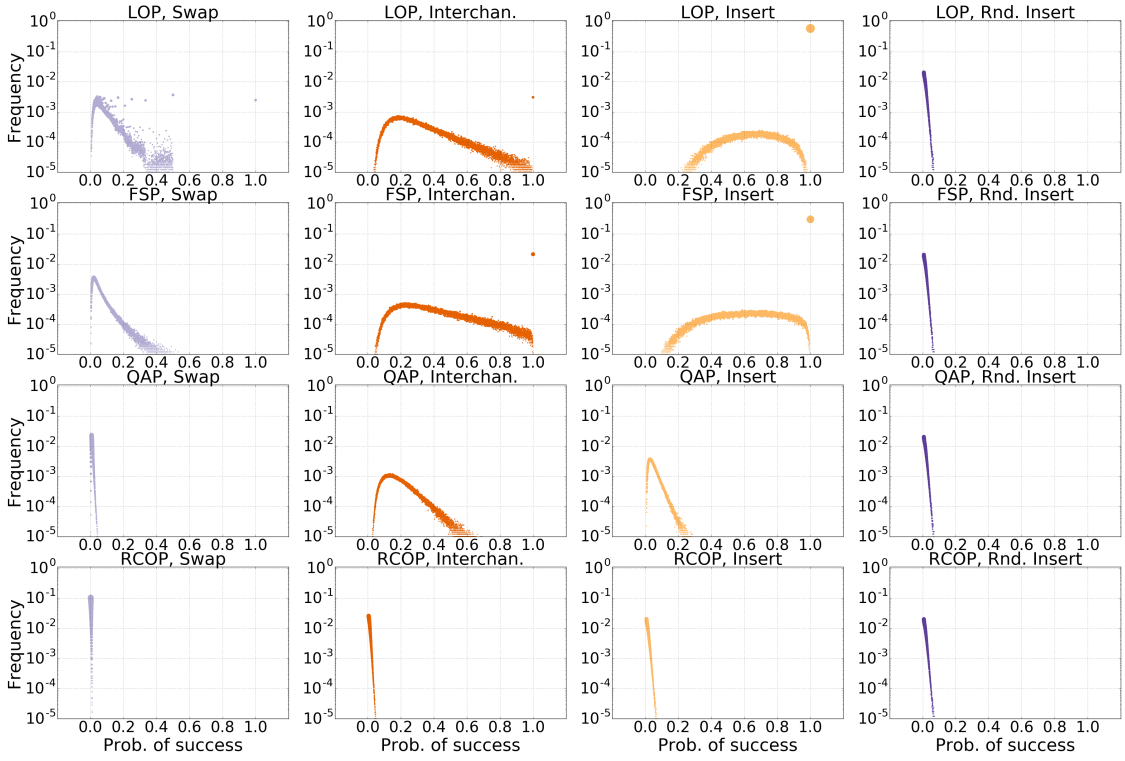
For each problem and the three neighborhoods (with the exception of swap in QAP), on average, instances generated u.a.r are much easier than instances generated for RCOP due to its rough landscape. Moreover, we observe the same difficulty distribution for RCOP under insert neighborhood than for LOP,

[1]Due to the size of the space of rankings, $n!!$, and the limited amount of computational resources, the analysis of difficulty could only be carried out for small size problems.

[2]The figures relative to $n \in \{3,4,5\}$ have been omitted for the sake of space, however, those shown in the paper are representative for $n \in \{3,4,5,6,7\}$.

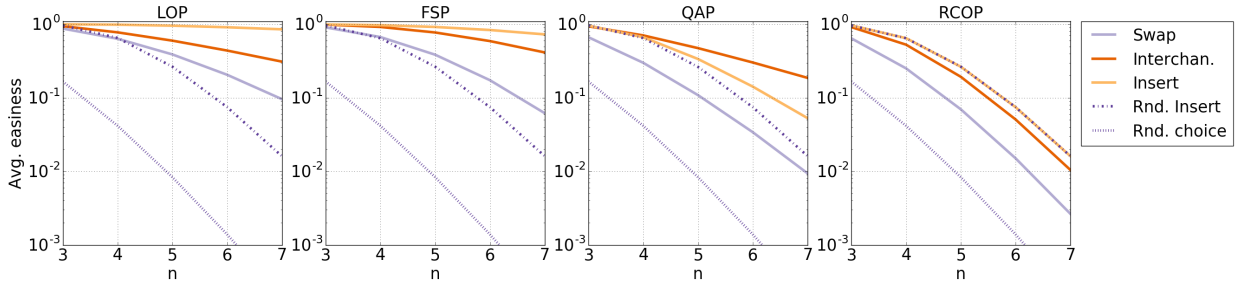


(a) $n = 6$

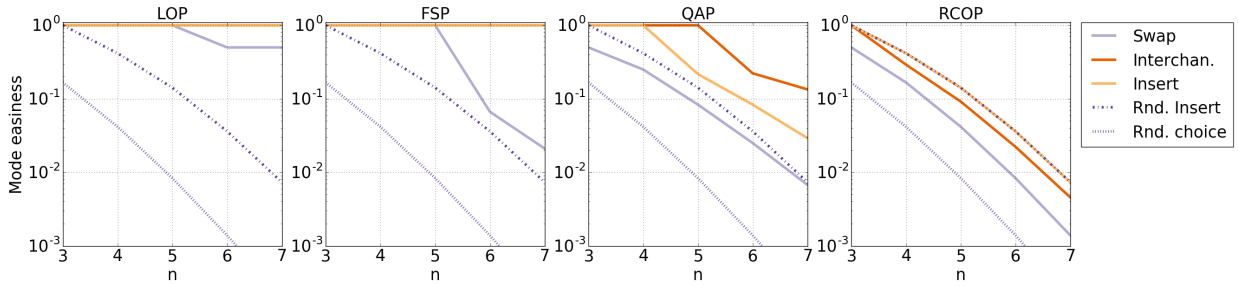


(b) $n = 7$

Fig. 1. The distribution of the easiness (probability of success) for instances generated u.a.r. generated instances for LOP, FSP, QAP and RCOP under swap, interchange, insert and *rand. insert* neighborhoods for $n \in \{6, 7\}$. Each point corresponds to a difficulty and its area is proportional to the number of instances generated with this difficulty.



(a) Mean easiness



(b) Mode Easiness

Fig. 2. A summary of the evolution of the easiness (average and mode) with respect to the size of the problem n for LOP, FSP, QAP and random COPS under swap, interchange, insert and random insert neighborhoods. We have included the percentage of times in which, taking a solution at random, we select the global optima.

FSP, QAP and RCOP under the *rand. insert* neighborhood (see Figure 1). Both observations suggest that the neighborhoods are able to exploit the information hidden in the generated instances of the COPs considered. Besides, results suggest that, when the instances have no information or the neighborhoods can not capture the hidden information, the distribution of the difficulty depends only on the topology of the neighborhood.

V. Discussion

The experiments suggest that the parameter roughness could influence the distribution of the difficulty of instances generated u.a.r. In what follows, we analyze the distribution of the difficulty for the three problems from the perspective of parameter roughness induced by the different neighborhoods.

For instance, the LOP under interchange and insert neighborhoods shows a similar distribution of the difficulty, being the difficulty under insert slightly lower. A possible explanation is that the roughness induced by the insert and interchange neighborhoods, and their respective sizes, are of the same order of magnitude. Thus, we recommend to use the insert neighborhood to deal with instances of LOP.

In FSP with interchange and insert neighborhoods, we observe the same phenomena than in LOP. Again, the roughness induced by interchange and insert are of the same order of magnitude. On the contrary, FSP instances under the swap neighborhood are more difficult. Even when the roughness induced by swap, interchange and insert are of the same order, the size of the swap neighborhood is only $\mathcal{O}(n)$, while the size of the interchange and insert is $\mathcal{O}(n^2)$. As a result, we recommend using the insert neighborhood when optimizing FSP instances.

In QAP, the instances are easier under interchange than under the insert neighborhood. In this case, interchange has a parameter roughness of $\mathcal{O}(n)$, while insert has a parameter roughness of $\mathcal{O}(n^2)$. Moreover, the instances are easier under interchange than under the swap neighborhood. Again, they have similar roughness, but the size of the swap neighborhood is smaller. Therefore, the interchange neighborhood is preferred to others when solving QAP instances.

As conclusion, it seems that, given a problem, the parameter roughness of a neighborhood and its size play an important role in the distribution of the difficulty of the generated instances. The provided evidences support the idea that, in order to select an appropriate neighborhood for dealing with instances of a particular COP, we could use a criteria based on the roughness induced by a neighborhood and its size.

In the remaining of this section, we analyze alternatives to the uniform sampling of parameters. In the experiments, we have seen that, when parameters are generated u.a.r., the distribution of the difficulty of the instances is not uniform. Moreover, even when artificial rankings are generated u.a.r., they do not produce a uniformly distributed difficulty. But, what is the effect of sampling the parameters of the instances i.i.d. according to another distribution?

In this sense, we tried to modify the average difficulty of the instances by sampling the parameters non-uniformly at random. For this purpose, we generated $2 \cdot 10^5$ parameters of instances for LOP, FSP and QAP i.i.d. according to a Beta distribution with $\alpha, \beta \in \{0.1, 1, 10\}$ for $n \in \{3, 4, 5, 6, 7\}$. In Fig. 3, we see the distribution of the difficulty for QAP under the insert neighborhood. Surprisingly, the obtained instances

have similar difficulty distribution to those generated u.a.r. ($\alpha = 1$, $\beta = 1$). The same behavior is observed for LOP, FSP and QAP under the swap, interchange and insert neighborhoods. It seems that the choice of the distribution used to generate the parameters i.i.d. does not have a meaningful impact on the distribution of the difficulty of the instances.

VI. Conclusions

In this work, we have analyzed empirically the difficulty of instances generated u.a.r for the Linear Ordering Problem, Flowshop Scheduling Problem and Quadratic Assignment Problem from a best improvement heuristic perspective using three different neighborhoods: swap, interchange and insert.

The conclusions can be summarized as: (1) the distribution of the difficulty is not uniform neither when the instances are generated u.a.r., nor when the artificial rankings are generated u.a.r., (2) depending on the neighborhood and the problem considered, the distribution of the difficulty can change dramatically, and (3) the distribution of the difficulty for instances generated u.a.r. seemingly depends on the roughness induced by the neighborhood and its size. In this sense, it seems to be possible to select the most appropriate neighborhood for a given COP based on these two criteria.

As future work, we will try to answer two interesting questions: i) Is it possible to control the difficulty of the randomly generated instances by means of a (non i.i.d) random sampling of their parameters? And, if the answer is positive, ii) how can we control the difficulty of the sampled instances?

Acknowledgments

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).

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 *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1645–1651.

[7] E. H. Aarts and J. K. Lenstra, Eds., *Local search in combinatorial optimization*. Princeton University Press, 2003.

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

[9] 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.

[10] T. C. Koopmans and M. J. Beckmann, "Assignment Problems and the Location of Economic Activities," Cowles Foundation for Research in Economics, Yale University, Cowles Foundation Discussion Papers 4, 1955.

[11] K. Baker, *Introduction to sequencing and scheduling*. Wiley, 1974.

[12] T. Back, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," in *Proceedings of the First IEEE Conference on Evolutionary computation (CEC 1994*, N. J. Piscataway, Ed. IEEE Neural Networks Council, 1994, pp. 57–62.

[13] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 4, no. 20, pp. 74–94, 1990.

[14] P. H. N. Mladenović, "Variable neighborhood search," *Computers & operations research*, vol. 11, no. 24, pp. 1097–1100, 1990.

[15] T. S. H. R. Lourenço, O. C. Martin, "Iterated local search," in *Handbook of metaheuristics*. Springer US, 2003, pp. 320–353.

[16] G. R. T. A. Feo, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 2, no. 6, pp. 109–133, 1995.

[17] M. R. Garey and D. S. Johnson, *Computers and intractability*. wh freeman New York, 2002, vol. 29.

[18] 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.

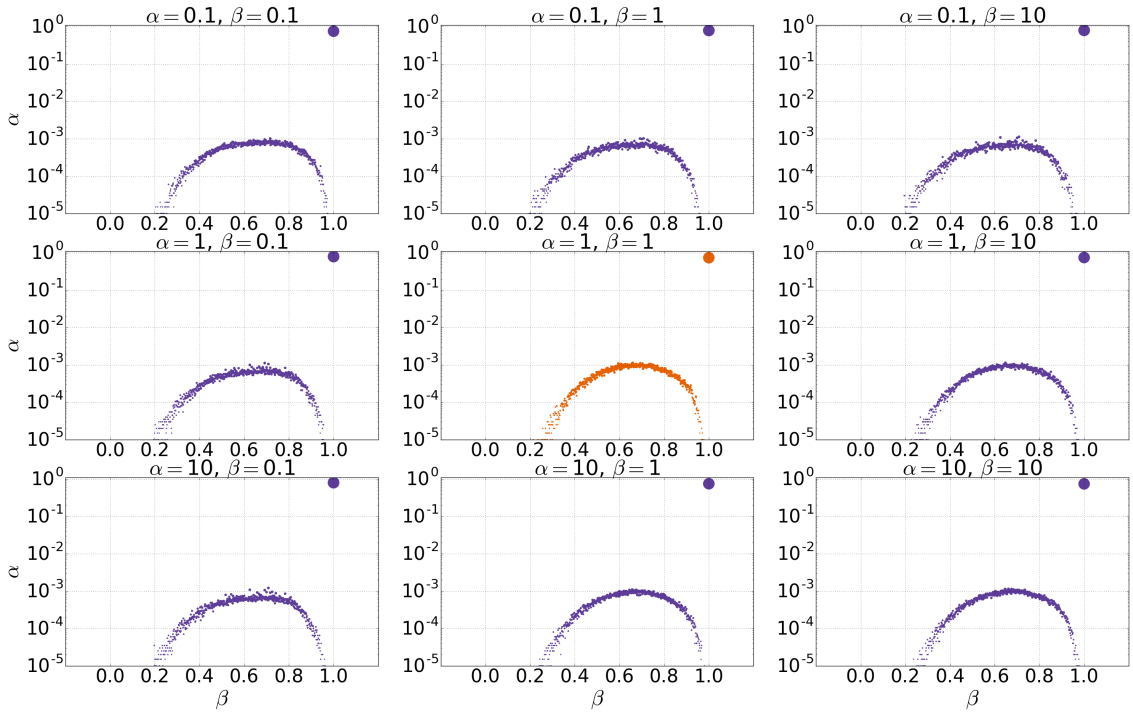


Fig. 3. The distribution of the easiness (probability of success) for instances i.i.d. according to a Beta distribution with parameters $\alpha, \beta \in \{0.1, 1, 10\}$ for LOP under the insert neighborhood for $n = 6$. Instances generated u.a.r. corresponds to $\alpha = \beta = 1$. Each point corresponds to a difficulty and its area proportional to number of instances generated with this difficulty.