Hybrid Heuristics for the Linear Ordering Problem

Erik Garcia, 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: erikg009@gmail.com, josu.ceberio@ehu.eus, ja.lozano@ehu.eus

***Abstract*—The linear ordering problem (LOP) is one of the classical NP-Hard combinatorial optimization problems. Motivated by the difficulty of solving it up to optimality, in recent decades a great number of heuristic and meta-heuristic algorithms have been proposed. Despite the continuous work on this problem, there is still room nowadays for designing strategies that beat the state-of-the-art algorithms, and take a step forward in terms of the quality of the obtained solutions.**

In this paper, two novel schemes are presented. The first algorithm consists of an iterated local search algorithm that carries out an organized exploration of the search space. The second scheme is an extension of the previous algorithm that, based on the properties of the LOP, proposes an exact procedure that allows us to improve the quality of the solutions systematically. Conducted experiments on one of the hardest LOP benchmarks (xLOLIB) show that 77 new best results were found out of 78 instances. The described strategies also provide innovative ideas for developing more advanced algorithms for solving the LOP.

I. Introduction

Studied for the first time in 1958 by Chenery and Watanabe [1], the Linear Ordering Problem (LOP) [2], [3] has become one of the most classical combinatorial optimization problems. In 1979, Garey and Johnson [4] classified it as NP-hard and showeds the difficulty in solving LOP instances up to optimality. Despite being formalized a long time ago, it has received recent attention in the research community due to its numerous applications in diverse fields, such as archaeology [5], economics [6], graph theory [7], machine translation [8] or mathematical psychology [9].

As a result, the literature contains a wide variety of papers that have dealt with the LOP in its different representations and approaches. Most of the proposed algorithms can be classified in three different strategies: exact, heuristic and meta-heuristic. Among the exact methods, the most meaningful include Branch and Bound [10], [11], Branch and Cut [12] and Cutting Plane algorithms [13], [14]. When it comes to solving combinatorial optimization problems, apart from the exact methods that always solve the problem to optimality, there are heuristic and meta-heuristics that supply a good, although not always optimal, solution. Pioneering works proposed constructive heuristics [1], [15], [16]. Such approaches were later outperformed by the advances produced in meta-heuristic optimization. Proof of this is the solutions for the LOP based on Local Search [17], [18], Genetic Algorithms [19], Tabu Search [20], Scatter Search [21], Variable Neighborhood Search [22], Ant Colony Optimisation [23], and recently Estimation of Distribution Algorithms [24].

According to a recent review of Marti et al. [25], the Memetic Algorithm (MA) and the Iterated Local Search (ILS) proposed by Schiavinotto et al. [26] and afterwards optimized by Ceberio et al. [3], are the algorithms that currently shape the state-of-the-art of the LOP. The MA is a hybrid algorithm which combines the canonical structure of a Genetic Algorithm with a high presence of local search procedures, either in the initialization of the population or in the evolutionary process itself. Alternatively, the ILS is a strategy that iteratively applies a local search algorithm to a single solution. When the process becomes trapped in a local optimal solution, the ILS applies a perturbation to the current solution, and continues with the optimization process until a termination criterion is satisfied. Both algorithms include an efficient implementation of a greedy local search algorithm.

Neither ILS nor MA explore the search space of solutions in an organized way. The exploration behavior of the ILS relies on the perturbation applied to the current solution. Similarly, the exploratory behavior of MA consists of the mutation of the solutions in the population by applying random modifications. Taking into account the factorial size of the search space of permutation problems, we think that the exploration behavior of ILS and MA can be further improved. Bearing that in mind, we propose an algorithm that explores the search space at different levels, starting from a more general exploration and delimiting slowly to local areas of the search space. This algorithm is called *Hybrid Exploration Algorithm* (HEA).

The second algorithm proposed in the paper is the *Sequential Exact Improvement* (SEI). This algorithm is based on the exact resolution of the LOP as a binary linear programming problem. When considering this type of approach, the exact solution can only be obtained for small instances ($n < 75$), and solving large instances (as in this work) is no longer feasible. Nevertheless, it is worth noting that the contribution of any item (or set of items) in the solution to the objective function is independent of the ordering of the previous and posterior items in the solution [3], [27]. This theoretical result permits the design of strategies that optimize the ordering of subsets of items (of size $n < 75$) in the solution, and guarantees that the overall quality of the solution cannot worsen.

The conducted experiments show that the presented algo-

rithms provide interesting procedures that allow us to outperform the best known results obtained for the LOP instances in xLOLIB.

The remainder of this paper is organized as follows. In Sect. II, we give a detailed description of the LOP, two different formalizations of the problem and some theoretical notes that will be used in Sect. III, where the two novel algorithms are presented. In Sect. IV, we carry out a thorough experimental analysis of the algorithms presented and compare the obtained results with the best known solutions. Finally, Sect. V sums up the main conclusions and raises some ideas for future work.

II. The Linear Ordering Problem

As pointed out in the introduction, the LOP was described for the first time in 1958, and throughout the following decades, a variety of representations have been published to formalize the problem. In what follows, we describe two possibilities that are used by the two algorithms proposed in the paper, respectively.

Given a square matrix $M = [m_{i,j}]_{i,j=1}^{n}$ of size n, the objective in the LOP is to find the joint permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ of rows and columns that maximizes the sum of the elements above the main diagonal. This objective can be formalized as maximizing the following objective function

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

The search space of solutions consists of all the permutations of the set of items $\{1, 2, ..., n-1, n\}$, and the number of solutions is $n!$.

A good point of the previous representation is that it is very intuitive, however, pioneering works tried to solve the problem by formulating it as a Binary Programming Problem [14]. This representation arises from the interpretation of the solutions of the LOP as directed acyclic complete graphs of n nodes. In this interpretation, the graph is formed by putting a directed edge from node i to node j if the item i is previous to the item j in the solution. Any solution of the LOP can be expressed as a unique directed acyclic complete graph and any directed acyclic complete graph can be interpreted as a unique solution.

Once the solutions are understood as graphs, the proposal of [14] is the following. The graph is represented with the binary variables x_{ij} of the problem which are the edges of the graph, 1 if there is an edge from node i to node j and 0 if there is no edge. So, the objective function to maximize is just $\sum_{i \neq j} x_{ij} m_{ij}$. The only remining aspects are the constraints needed to ensure that the binary variables form a directed acyclic complete graph, that is, a solution of the LOP.

The first constraint is to ensure the completeness of the graph, such that $x_{ij} + x_{ji} = 1$ for each different pair of nodes. The second and third constraints are to ensure that there are no cycles, such that $x_{ij} + x_{jk} + x_{ki} \leq 2$ and $x_{ik} + x_{kj} + x_{ji} \leq 2$ for each different triple of nodes.

Summarizing, the LOP as a binary linear programming problem can be written as follows:

$$\begin{aligned} \max : & \textstyle\sum_{i \neq j} x_{ij} m_{ij} & \\ & x_{ij} + x_{ji} = 1, & \forall i, j \in V_n, i < j \\ & x_{ij} + x_{jk} + x_{ki} \leq 2, & \forall i, j, k \in V_n, i < j, i < k, j < k \\ & x_{ik} + x_{kj} + x_{ji} \leq 2, & \forall i, j, k \in V_n, i < j, i < k, j < k \\ & x_{ij} \in \{0, 1\}, & \forall i \neq j \in V_n \end{aligned}$$

where V_n is the set of item/nodes $\{1, 2, \ldots, n\}$.

III. Hybrid Heuristic Algorithms

In this section, the contribution of the paper, the two hybrid heuristic algorithms, are presented.

A. Hybrid Exploration Algorithm

As noted previously, in LOP, as in many other permutation problems, the search space of solutions is $n!$ (all the permutations of size n), and thus, an exhaustive examination of all the solutions is not feasible beyond a small problem size n. As a consequence, most of the state-of-the-art designs try to find a balance between the exploration and exploitation abilities of the algorithms. In the case of the LOP, we think that strategies to balance that trade-off in the reference algorithms, ILS and MA [26], [3], can be further improved. On one hand, the exploration in ILS consists of a perturbation of the solution every time it reaches a local optimum. The idea is to escape from the local optimum and optimize other areas of the search space. On the other hand, MA is a population-based algorithm that executes a mutation operator that permits the addition of new random features to the solutions in the population. This procedure introduces diversity to the population and aims to explore new solutions.

Bearing this in mind, we present a simple meta-heuristic that faces the exploration of the search space by organizing the search at different levels: starting from a more broad exploratory behaviour and focusing it slowly to specific areas of the search space. This algorithm is called the *Hybrid Exploration Algorithm* (HEA). Note that even though in this work the aim is to optimize the LOP, the algorithm could be generalized to other problems where the solutions are represented as permutations or in general to other optimization problems with the correct adaptation.

According to the literature, using local search in the algorithm is almost mandatory to obtain high performances when solving the LOP. ILS is based on the local search and it explores the search space with perturbations that can be easily set to be larger or smaller. In addition, population-based algorithms, such as MA, are more likely to explore huge search spaces, therefore, we propose adapting the ILS to a population-based algorithm and configuring the perturbations to perform the search in an organized manner.

The pseudocode of HEA is presented in Algorithm 1. The algorithm starts by building the initial population of Pop_size. To that end, it runs a stochastic variant of Becker's constructive heuristic (see section III-A1) as many times as there are solutions in the population. Afterwards, the local search algorithm under the insert neighborhood is applied to each of the solutions.

Once the population has been initialized, the algorithm iterates on the next instructions. Firstly, the individuals of the population are moved (perturbed) with a certain number of swaps, and improved with a local search (under the insert neighborhood) to create an auxiliary population. Secondly, the best solutions of the newly created population and the former one are mixed, and the best Pop_size is chosen, while the rest of the solutions are discarded.

When a solution is perturbed considering a high number of swaps, it is expected that the obtained solution should be far from the one perturbed. Similarly, a low number of swaps results in a similar solution. Assuming this principle, we propose an algorithm that, at the beginning of the optimization, perturbs the solutions in the population with a high number of swaps in order to locate the regions with the best solutions in the search space, and, later, progressively decreases the number of swaps to explore in detail the chosen regions.

The number of random swaps starts with $0.45n$ and decreases at a rate of 5% each time until $0.05n$[1]. In addition, for each number of swaps, the perturbation and improvement steps are repeated until there is no improvement in the highest value of the population n_stop consecutive times.

Algorithm 1 Hybrid Exploration Algorithm (HEA)

```
Pop ← {}
for i = 1,...,Pop_size do
    π ← ProbabilisticGreedyConstructive(k)
    π' ← InsertLocalSearch(k)
    Pop ← Pop ∪ π'
end for
for i = 0.45, 0.40, ..., 0.1, 0.05 do
    cont ← 0
    while cont<n_stop do
        New_Pop ← Swap(n × i, Pop) //Random swaps
        New_Pop ← InsertLocalSearch(New_Pop)
        Pop ← Pop ∪ New_Pop //Remove worst
        if no improvement then
            cont ← cont+1
        else
            cont ← 0
        end if
    end while
end for
σ ← Pop[0] //Best solution
return σ
```

1) Probabilistic Greedy Constructive: The constructive heuristic used in HEA is a probabilistic version of the best known constructive algorithm for the LOP: the constructive of Becker [16]. This heuristic builds a solution by adding items one by one to it. At each time, the remaining items are ranked by a quotient

$$q_i = \frac{\sum_{i \neq j} m_{ij}}{\sum_{i \neq j} m_{ji}}, \text{ where } j \in \{\text{remaining items}\}$$

which is the ratio between the contribution (sum of the parameters in the row) and non-contribution (sum of the parameters in the column) to the solution. As the contribution and non-contribution of the remaining items depend only on the remaining items, those items already added to the solution are not taken into account. At every iteration of the algorithm, the item with the largest quotient is added to the solution.

It is worth noting that the heuristic of Becker adds the item with the maximum quotient value to the construction at each time, so it always builds the same solution, i.e., the algorithm is deterministic.

The probabilistic version we propose consists of adding to the solution one of the best k remaining items with a probability based on their quotient value. If the sum of the best k items is $d = \sum_{\text{best } k} q_i$, then the probability to add one of these best k items is $p_i = q_i/d$. Every time that the sum of parameters in the column of an item is zero, this is added to the solution automatically.

Algorithm 2 Probabilistic Greedy Constructive

```
R = {1, 2, ..., n} Initialize the set of remaining items
σ ← ()
while R is not empty do
    For each i ∈ R compute q_i = (Σ_{i≠j} m_ij) / (Σ_{i≠j} m_ji)
    d ← Σ_{best k} q_i
    Chose one of the best k with probability p_i = q_i/d
    Append chosen i to σ
    Delete chosen i from R
end while
return σ
```

This procedure permits different solutions to be built every time the algorithm is run (with $k > 1$). The pseudocode of the algorithm is described in Algorithm 2.

B. Sequential Exact Improvement

In general, when the Binary Linear Programming (BLP) representation is used to solve the LOP, the time needed to solve standard size problems is not affordable. Schiavinotto et al. [26] let an exact method run for one week for a matrix of 250 without getting any results. Although the exact solution cannot be obtained beyond certain values of n, due to the characteristics of the LOP, it is possible to use the BLP approach for improving the quality of solutions by solving submatrices of the problem. This is an innovative way of improving the solutions of the LOP that cannot be found in the literature, and can complement any algorithm already published for the LOP.

For a given solution σ, the contribution of any item at position i is independent of the ordering of the items ordered in $\{1, \ldots, i-1\}$ positions, and also independent of the ordering of the items in $\{i+1, \ldots, n\}$ positions [3]. Not limited to

[1]These parameters were deduced from a set of preliminary calibration experiments.

that, the same idea can be extended to sets of items. So, the contribution of the items at positions from i to j is independent of the ordering of the items in $\{1,\ldots,i-1\}$ positions, and items in $\{j+1,\ldots,n\}$. As a result, if an optimization method is applied to the submatrix related to some contiguous items in the solution, based on the previous statements, it is guaranteed that the overall quality of the new solution can only improve, or remain equal if that subset of items is already optimal.

Let us consider, with illustrative purposes, the LOP instance in Fig. 1. Given the solution $\sigma = (2,3,1,4,5)$, if the contiguous numbers $(3,1,4)$ are considered, the exact solution of that submatrix inside the red square in Fig. 1 is $(3,4,1)$. In addition, we see that the contribution of item 2 (inside the

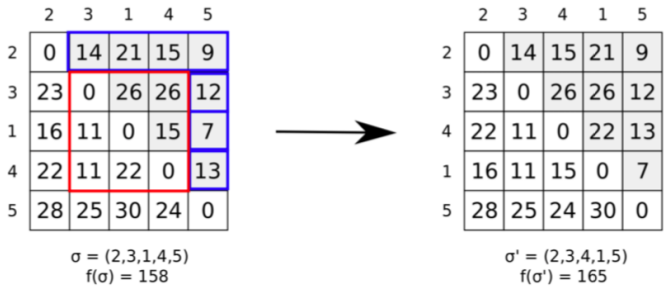

Fig. 1. Exact method applied to the contiguous subset of items $(3,1,4)$ to improve the current solution.

blue rectangle) before and after the optimization of the subset of items is the same. There is no contribution of item 5 as it is the last one, but the contributions of the posterior item, if they had existed, would have also remained the same. Only the contribution of the reordered items changes.

The previous remark on the LOP permits the improvement of any solution by applying optimization methods to submatrices of the problem. In this case, we propose a strategy to optimize contiguous submatrices in the solution following the BLP approach. In this sense, currently, there is also a huge variety of really competitive optimization solvers which have already implemented exact methods to solve these binary linear programs. Accordingly, taking that into account, in this work the commercial solver CPLEX has been used.

In order to provide intuition on the idea of solving contiguous matrices, we run some preliminary experiments with CPLEX on matrices of size 150 and 250 of the xLOLIB benchmark [26], and compared the results obtained with the best known results. From the conducted experiments, we conclude that:

- the algorithms based on solving continuous submatrices without any previous optimization are not competitive enough. So it is desirable to provide a good solution to the exact solver.
- Unless the size of the submatrix is large enough, the exact procedure does not contribute at all in the improvement of the solution.
- The time needed by the CPLEX to solve an instance of the LOP can be drastically reduced when the given instance is a good solution.

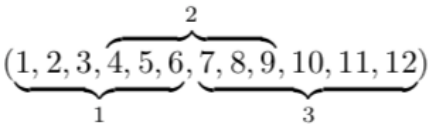

Fig. 2. Example of overlapping of submatrices of size 6 on the solution $(1,2,3,4,5,6,7,8,9,10,11,12)$

In order to see the reduction of computational time required by CPLEX when a good solutions is provided, we conducted the following experiment. This experiment consists of solving matrices with the CPLEX, and once the solution is obtained, the matrices are reordered with the optimum solution obtained, and solved again with the CPLEX. Next, the execution times for solving the initial matrix and the reordered matrix are compared. Here we are reordering the matrices with the optimum solution (the CPLEX does not need to make any changes), but it can be considered quite similar to reordering them with a good solution that is not the optimum one, even if some small changes are needed. We used 10 matrices of size 50, 60 and 70, and this was built taking random rows and columns from some random instances of size 150 of xLOLIB. The results are provided in Table I.

TABLE I
THE TIMES TO SOLVE EACH OF THE 10 MATRICES OF SIZES 50, 60 AND 70 WITH CPLEX. THE FIRST LINE OF EACH SIZE IS THE TIME TAKEN TO SOLVE THE INITIAL MATRIX AND THE SECOND LINE IS THE TIME TAKEN TO SOLVE THE MATRIX REORDERED WITH THE OPTIMUM SOLUTION.

n	Time(s)									
50	3	1	26	28	36	40	74	1	1	2
	1	1	10	19	11	23	27	1	1	2
60	499	509	308	45	5	7	124	6	3	4
	221	68	121	26	4	4	31	4	2	3
70	646	1855	3998	300	984	6185	1146	996	182	1378
	854	677	2437	599	163	5245	551	306	107	1047

In the view of the results, we can conclude that when reordering the matrix with a good solution, the reduction of the time needed to solve it is significant (in the great majority of the instances).

Therefore, based on what has been exposed in previous paragraphs, we propose the *Sequential Exact Improvement* algorithm (SEI). This procedure is implemented to be used as a post-optimizer of another algorithm, i.e., HEA. For the sake of maximizing the probabilities of improving the solution given by HEA, we propose a sequential resolution of overlapped submatrices. For example, given the solution $(1,2,3,4,5,6,7,8,9,10,11,12)$ of Fig. 2, the algorithm would optimize first the submatrix from position 1 to 6, secondly, the items from positions 4 to 9, and finally from positions 7 to 12. The idea of sequential overlapping submatrices permits any item to be moved to its optimal position. The process is repeated until no improvement is obtained. The pseudocode of SEI is presented in Algorithm 3.

Algorithm 3 Sequential Exact Improvement

```
Initial Solution: a good solution (for example, HEA)
Set the size of the submatrices and the overlapping between them
i ← 0
cont ← 0
while cont<n_submatrices do
    i ← remainder(i/n_submatrices)+1
    Solve the i^th submatrix
    if no improvement then
        cont ← cont+1
    else
        cont ← 0
    end if
end while
return σ
```

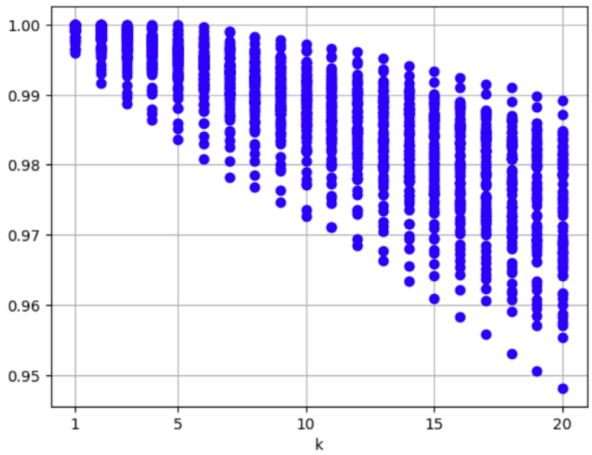

Fig. 3. The normalized fitness average values on the 78 instances of the xLOLIB for each value of k from 1 to 20.

IV. Experimental Study

With the aim of evaluating the validity of the proposed strategies, in this section we present a broad range of experiments. Particularly, the proposed algorithms are run on the xLOLIB benchmark of instances and the obtained results are compared with the best known results reported in the literature. This set of instances is composed of 39 instances of size 150 and 39 instances of size 250, and were generated by Schiavinotto et al. [26] through sampling uniformly at random the elements of the LOLIB instances. Currently, this benchmark constitutes the most challenging benchmark published on the LOP.

A. Parameter Tuning

The first step consists of setting up the two algorithms, that is, tuning their parameters.

1) Hybrid Exploration Algorithm: As regards HEA, there are three parameters to set: the number of items k to consider in the probabilistic greedy constructive, the size of the population (Pop_size) and the number of consecutive non-improvements iterations with the same best value of the population (n_stop).

In order to gain insights into the k parameter, for each instance of xLOLIB we constructed 1000 solutions, with k ranging from 1 to 20. Obtained fitness values were averaged and normalized for each k value. The results are presented in Fig. 3.

Bearing in mind Fig. 3, the best setting for the parameter is $k = 1$, as the best normalized average values are obtained for most of the instances (close to 1). However, as explained above, with $k = 1$ the algorithm becomes deterministic, and thus, since different solutions are needed to create an heterogeneous population, we set $k = 5$, as it provides values close to 1, but in the meantime, it has enough variance to build heterogeneous solutions.

In order to tune the population size and the stopping criterion, the other two parameters, 10 instances of the xLOLIB were randomly chosen, 5 instances of size 150 and another 5 of size 250. In addition, $\{15, 20, 25, 30\}$ population sizes and $\{25, 50\}$ values for non-consecutive improvements were considered. 10 repetitions of HEA were performed for each combination of both parameters. The results are summarized in Table II in terms of average results of the obtained fitness values and execution time of 10 repetitions.

Looking at the results in Table II, the stopping criteria of 25 non-consecutive improvements is in most of the instances the best option (in 8 instances out of 10). In relation to the size of the population, it is not so clear, however, it looks like larger sizes obtained better results (in 6 instances out of 10, the best population size is 30).

2) Sequential Exact Improvement: With respect to the set-up of the sequential exact improvement algorithm, the structure of the sequence needs to be set, that is, the size of the submatrices to solve by CPLEX. As mentioned in previous experiments, the size of the submatrices has to be big enough in order to make an improvement. It is worth noting that for submatrices of size 50-60, they are actually optimal. Unfortunately, the computation time grows exponentially with the size of the submatrices (see Table I). According to the previous experiments, solving submatrices of size 80 is affordable in the majority of the cases, but not systematically in a reasonable time span (the computational time needed is excessive). So, the size of the submatrices is set to 75 as it supposes affordable execution times.

Once the size of the submatrices is set, an overlapping structure for the submatrices in instances of size 150 and 250 must be provided. We propose using the structures shown in Fig. 4 for each instance size. Given the i^{th} submatrix, it overlaps the first half of items with the $(i-1)^{th}$ submatrix, and similarly, the second half of items is overlapped with items that share with the $(i+1)^{th}$ submatrix.

B. Performance Evaluation

Once the two algorithms has been set up, in the second step of the experimentation, the performance of both is measured

TABLE II

AVERAGE FITNESS VALUE AND AVERAGE EXECUTION TIME OF 10 RUNS FOR EACH INSTANCE WITH DIFFERENT POSSIBLE COMBINATIONS OF 15, 20, 25, 30 POPULATION VALUES AND 25, 50 STOPPING CRITERIA VALUES. IN BOLDFACE WE HIGHLIGHT THE RESULTS THAT HAVE THE BEST TRADE-OFF OF AVERAGES.

pop_size	Best Known	15		20		25		30	
n_stops		25	50	25	50	25	50	25	50
N-be75np_150	7172840	4508938	4508094	4508895	4510121	4509697	4509899	**4509849**	4510021
		38s	80s	49s	114s	62s	137s	**74s**	159s
N-stabu1_150	2874738	**318945**	318944	318934	318945	318947	318948	318935	318948
		34s	67s	44s	89s	58s	109s	69s	143s
N-be75eec_150	3480392	3157765	3155493	3157786	3158044	3157531	3157730	**3158948**	3158215
		42s	71s	54s	96s	60s	127s	**81s**	146s
N-be75tot_150	12287935	1625807	1626068	1625665	1626036	1626035	1625995	**1626286**	1626087
		39s	77s	51s	92s	63s	131s	**80s**	144s
N-be75oi_150	2246282	957787	**957903**	957797	957887	957846	957832	957806	957884
		37s	**80s**	50s	107s	66s	127s	76s	141s
N-be75oi_250	5907714	17806671	17811018	17804801	17807416	17810906	17814331	**17814316**	17815642
		314s	644s	432s	834s	553s	1095s	**659s**	1281s
N-be75eec_250	8881423	5908086	5908937	**5910149**	5910700	5909920	5910521	5909914	5910219
		316s	672s	**471s**	853s	525s	1048s	636s	1265s
N-be75np_250	17796212	11903284	11904173	11904034	11905017	11904535	11904615	**11904850**	11903899
		278s	570s	415s	781s	501s	1026s	**644s**	1167s
N-be75tot_250	30934111	25405846	**25411106**	25400288	25404980	25407159	25412144	25410188	25411427
		316s	**639s**	410s	862s	526s	1087s	704s	1283s
N-stabu1_250	7734436	3057834	3057769	3058613	3059100	3058980	3058908	**3059203**	3059695
		308s	531s	424s	831s	521s	1046s	**659s**	1176s

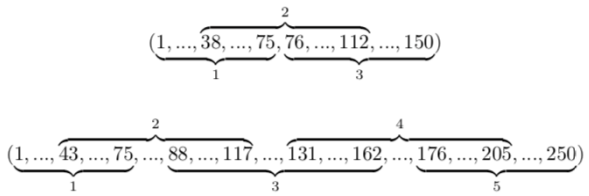

Fig. 4. Overlapping schemes of submatrices in instances of size 150 and 250.

and compared with respect to the best known results reported in the literature. Currently, the best known results for each instance of the xLOLIB are the maximal results obtained by Ceberio et al. [3] from 20 executions of the ILS$_r$ and MA$_r$ using a stopping criteria of $10000n^2$ evaluations.

We run the two algorithms, HEA and SEI, 10 repetitions on each instance from xLOLIB, and recorded the best and average fitness values of the obtained results. The results are presented in Table III. It is worth noting that, because of the nature of both algorithms and their stopping criterion, it makes not sense to stop the algorithm after a fixed number of function evaluations.

As can be seen, there are only a few best known solutions that are not outperformed by both algorithms. In fact, from 78 instances there is only one instance in which the best result is the one reported in the literature. Moreover, in 72 instances both algorithms obtained better best results than the state-of-the-art. Furthermore, another remarkable fact is that in 53 instances the obtained average value of both algorithms is better than the best known.

V. CONCLUSIONS & FUTURE WORK

In this paper, two hybrid heuristic algorithms were proposed for solving a classical combinatorial optimization problem: the linear ordering problem. The first algorithm consists of an iterated local search algorithm under the insert neighbourhood that carries out an organized exploration of the search space. The second scheme is an extension of the previous algorithm that, based on the properties of the LOP, carries out an exact and sequential procedure that permits the quality of the solutions to be improved systematically. Conducted experiments on one of the hardest LOP benchmarks (xLOLIB) revealed an outstanding performance when compared to the existing best known solutions reported in the literature. Moreover, the experimentation concluded that there is still room nowadays for designing strategies that beat the state-of-the-art algorithms, and take a step forward in terms of quality of obtained solutions.

For future research, there are a number of lines that would be interesting to explore. In this sense, one of them is to carry out a fair comparison between the hybrid exploration algorithm, the ILS and the MA with the execution time of the hybrid exploration algorithm (it stops when it finishes, the execution time cannot be set). This experiment would determine whether the organized exploration of the search space is more efficient than ILS or MA. Even so, there is practically no difference between the results of ILS or MA with the stopping criteria of $1000n^2$ and $10000n^2$ evaluations, so this suggests that there is not going to be an improvement, even if the stopping criteria (or execution time) is increased.

Another interesting research line would be the reduction of the execution time of the CPLEX when the matrix is reordered with a good solution as this allows larger submatrices to be solved, and thus, better results to be obtained. Moreover, in this paper we carried out a sequential procedure for applying CPLEX, other procedures nevertheless, such as a hierarchical one, could also be developed.

TABLE III
RESULTS OF HEA AND SEI ON XLOLIB BENCHMARK. VALUES IN BOLDFACE HIGHLIGHT THE BEST RESULTS THAT OUTPERFORMED THE BEST KNOWN REPORTED FOR THAT INSTANCE.

	$n = 150$					$n = 250$				
		HEA		SEI			HEA		SEI	
Instance	*Best Known*	Best	Average	Best	Average	*Best Known*	Best	Average	Best	Average
N-be75eec	3480392	**3480968**	3479751	**3482828**	3481407	8881423	**8892995**	8891153	**8896595**	8891954
N-be75np	7172840	**7182660**	7172569	7174000	7171680	17796212	**17822266**	17811829	**17822571**	17815625
N-be75oi	2246282	**2246465**	2246002	**2246853**	2246782	5907714	**5911731**	5910181	**5912445**	5910183
N-be75tot	**12287935**	**12287935**	12281721	12287707	12275247	30934111	**30985366**	30972938	**30998382**	30973390
N-stabu1	2874738	**2875732**	2874011	**2875732**	2873920	7734436	**7743135**	7738402	**7747599**	7740798
N-stabu2	4326696	**4327108**	4326312	**4327538**	4326382	11496181	**11505756**	11499498	**11516097**	11505425
N-stabu3	4508566	**4510435**	4509880	**4510445**	4510398	11895591	**11909920**	11903383	**11909724**	11907205
N-t59b11xx	3234445	**3239045**	3235360	**3239550**	3237133	8402430	**8409505**	8403831	**8411760**	8408517
N-t59d11xx	1461924	**1462418**	1462131	**1462418**	1462004	3837704	**3842547**	3838587	**3843003**	3838247
N-t59f11xx	1542532	**1543733**	1540892	**1543733**	1541859	3984558	**3995865**	3992428	**3995868**	3991806
N-t59n11xx	318967	318951	318946	318951	318948	823598	**824974**	824596	**825023**	824523
N-t65b11xx	6448366	**6454792**	6452102	**6455180**	6453499	17248262	**17267344**	17261143	**17275227**	17261948
N-t65d11xx	3556350	**3559347**	3559129	**3559347**	3558405	9342499	**9352734**	9349893	**9353593**	9350913
N-t65f11xx	3158431	**3159326**	3157831	**3159326**	3158124	8406064	**8417742**	8411076	**8414863**	8411313
N-t65l11xx	253245	253224	253142	**253417**	253168	665926	**666820**	666548	**666915**	666412
N-t65n11xx	550411	**550893**	550720	**550893**	550729	1429041	**1430863**	1429693	**1430816**	1429579
N-t69r11xx	11853137	**11855957**	11852263	**11855957**	11852299	31781332	**31790415**	31777777	**31822418**	31776753
N-t70b11xx	9645823	9644816	9637934	**9645830**	9637001	25387082	**25415047**	25410179	**25416607**	25412311
N-t70d11xn	5823471	**5825947**	5825338	**5825947**	5825375	15195896	**15206075**	15201365	**15211056**	15204745
N-t70d11xx	6172834	**6173935**	6170208	**6174178**	6168231	16037941	**16042913**	16035002	**16043251**	16036264
N-t70f11xx	5149944	**5150097**	5147372	**5150097**	5147518	13575965	**13588034**	13580097	**13598786**	13588571
N-t70l11xx	436807	**436862**	436832	**436862**	436836	1112228	1112058	1111447	**1114166**	1112006
N-t70n11xx	948721	**948913**	948879	**948913**	948776	2443584	**2445125**	2442707	**2444720**	2443509
N-t74d11xx	9381843	**9396044**	9392875	**9396044**	9393381	24399174	**24432273**	24422273	**24445112**	24430106
N-t75d11xx	9637128	**9642140**	9638921	**9642140**	9639000	25013216	**25045329**	25033516	**25048727**	25032804
N-t75e11xx	41529895	**41570193**	41568469	**41570193**	41570193	106636934	**106912584**	106823295	**106912899**	106807717
N-t75k11xx	1541226	**1541596**	1540908	**1541596**	1541462	4092119	**4094732**	4092379	**4094468**	4092300
N-t75n11xx	1742437	**1743094**	1742710	**1743094**	1743094	4522276	**4529853**	4526019	**4529018**	4526330
N-tiw56n54	837257	**837945**	837412	**837945**	837479	2097731	**2099727**	2099079	**2099727**	2098696
N-tiw56n58	1155078	**1155392**	1154351	**1155392**	1154526	2902539	**2906727**	2905200	**2906751**	2906046
N-tiw56n62	1626118	**1626528**	1626170	**1626495**	1626243	4139627	**4145747**	4143253	**4144996**	4143041
N-tiw56n66	2107453	**2107619**	2107134	**2107619**	2107487	5366742	**5370955**	5368831	**5371157**	5369152
N-tiw56n67	2372706	**2372945**	2372678	**2372945**	2372930	6319305	**6326441**	6323465	**6326881**	6325307
N-tiw56n72	4135204	**4135689**	4133551	**4135289**	4133068	11146034	**11156640**	11150552	**11153587**	11150412
N-tiw56r54	957753	**957966**	957801	**958060**	957854	2385800	**2387734**	2386997	**2388072**	2387360
N-tiw56r58	1219012	**1219295**	1218141	**1219295**	1218563	3057839	**3060388**	3058388	**3060787**	3058506
N-tiw56r66	1940681	**1940755**	1940717	**1940755**	1940724	4943468	**4948268**	4947254	**4948886**	4947314
N-tiw56r67	2056039	**2057237**	2055787	**2056665**	2055385	5287940	**5293055**	5289702	**5293543**	5291981
N-tiw56r72	2821686	**2823758**	2821757	**2823758**	2822299	7445814	**7452983**	7448311	**7457217**	7450081

ACKNOWLEDGMENTS

This work has been partially supported by the Research Groups 2013-2018 (IT-609-13), BERC 2018-2021, and ELKARTEK programs (Basque Government), the projects TIN2016-78365-R (Spanish Ministry of Economy, Industry and Competitiveness) and Severo Ochoa Program SEV-2017-0718 (Spanish Ministry of Economy, Industry and Competitiveness).

REFERENCES

[1] H. B. Chenery and T. Watanabe, "International comparisons of the structure of production," *Econometrica*, vol. 26, no. 4, pp. 487–521, October 1958.

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

[3] 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.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[5] F. Glover, T. Klastorin, and D. Klingman, *Optimal weighted ancestry relationships*, ser. Management science report series. Business Research Division, Graduate School of Business Administration, University of Colorado, 1972.

[6] W. Leontief, *Input-Output Economics*. Cambridge University Press, 2008.

[7] I. Charon and O. Hudry, "A survey on the linear ordering problem for weighted or unweighted tournaments," *4or*, vol. 5, no. 1, pp. 5–60, Mar. 2007.

[8] R. Tromble and J. Eisner, "Learning linear ordering problems for better translation," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, ser. EMNLP '09, 2009, pp. 1007–1016.

[9] J. G. Kemeny, "Mathematics without numbers," *Daedalus*, vol. 88, pp. 577–591, 1959.

[10] R. Kaas, "A branch and bound algorithm for the acyclic subgraph problem," *European Journal of Operational Research*, vol. 8, no. 4, pp. 355 – 362, 1981.

[11] 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.

[12] M. Grötschel, M. Jünger, and G. Reinelt, "A cutting plane algorithm for the linear ordering problem," *Operations research*, vol. 32, no. 6, pp. 1195–1220, 1984.

[13] J. Mitchell and B. Borchers, "Solving real-world linear ordering prob-

lems using a primal-dual interior point cutting plane method," *Annals of Operations Research*, vol. 62, no. 1, pp. 253–276, 1996.

[14] ——, "Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm," in *High Performance Optimization*, ser. Applied Optimization, H. Frenk, K. Roos, T. Terlaky, and S. Zhang, Eds. Springer US, 2000, vol. 33, pp. 349–366.

[15] H. Aujac, "La hiérarchie des industries dans un tableau des échanges interindustriels," *Revue économique*, vol. 11, no. 2, pp. 169–238, 1960.

[16] O. Becker, "Das helmstädtersche reihenfolgeproblem – die effizienz verschiedener näherungsverfahren -," in *Computers Uses in the Social Sciences, Berichteiner Working Conference*, Vienna, 1967.

[17] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell system technical journal*, vol. 49, no. 1, pp. 291–307, 1970.

[18] S. Chanas and P. Kobylanski, "A new heuristic algorithm solving the linear ordering problem," *Computational Optimization and Applications*, vol. 6, no. 2, pp. 191–205, 1996.

[19] I. Charon and O. Hudry, "Lamarckian genetic algorithms applied to the aggregation of preferences," *Annals of Operations Research*, vol. 80, no. 0, pp. 281–297, 1998.

[20] M. Laguna, R. Marti, and V. Campos, "Intensification and diversification with elite tabu search solutions for the linear ordering problem," *Computers & Operations Research*, vol. 26, pp. 1217–1230, 1999.

[21] V. Campos, F. Glover, M. Laguna, and R. Martí, "An experimental evaluation of a scatter search for the linear ordering problem," *Journal of Global Optimization*, vol. 21, no. 4, pp. 397–414, 2001.

[22] C. G. Garcia, D. Pérez-Brito, V. Campos, and R. Martí, "Variable neighborhood search for the linear ordering problem," *Computers & Operations Research*, vol. 33, no. 12, pp. 3549 – 3565, 2006.

[23] C. Chira, C. M. Pintea, G. C. Crisan, and D. Dumitrescu, "Solving the linear ordering problem using ant models," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1803–1804.

[24] J. Ceberio, A. Mendiburu, and J. A. Lozano, "The plackett-luce ranking model on permutation-based optimization problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 494–501.

[25] R. Martí, G. Reinelt, and A. Duarte, "A benchmark library and a comparison of heuristic methods for the linear ordering problem," *Comput. Optim. Appl.*, vol. 51, no. 3, pp. 1297–1317, Apr. 2012.

[26] 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.

[27] J. Ceberio, A. Mendiburu, and J. A. Lozano, "A note on the boltzmann distribution and the linear ordering problem," in *Conference of the Spanish Association for Artificial Intelligence*, 2016.