Technical Report

UNIVERSITY OF THE BASQUE COUNTRY
Department of Computer Science and Artificial Intelligence

The Linear Ordering Problem Revisited

Josu Ceberio, Alexander Mendiburu,
Jose A. Lozano

January, 2014

San Sebastian, Spain
www.ehu.es/ccia-kzaa
hdl.handle.net/10810/4562

The Linear Ordering Problem Revisited

Josu Ceberio[a], Alexander Mendiburu[b], Jose A. Lozano[a]

[a]*Intelligent Systems Group*
Department of Computer Science and Artificial Intelligence
University of the Basque Country UPV/EHU
200018 Donostia, Spain. Tel. +34 943018012
[b]*Intelligent Systems Group*
Department of Computer Architecture and Technology
University of the Basque Country UPV/EHU
200018 Donostia, Spain

Abstract

The Linear Ordering Problem is a popular combinatorial optimisation problem which has been extensively addressed in the literature. However, in spite of its popularity, little is known about the characteristics of this problem. This paper studies a procedure to extract static information from an instance of the problem, and proposes a method to incorporate the obtained knowledge in order to improve the performance of local search-based algorithms. The procedure introduced identifies the positions where the indexes cannot generate local optima for the insert neighbourhood, and thus global optima solutions. This information is then used to propose a restricted insert neighbourhood that discards the insert operations which move indexes to positions where optimal solutions are not generated.

In order to measure the efficiency of the proposed *restricted insert neighbourhood* system, two state-of-the-art algorithms for the LOP that include local search procedures have been modified. Conducted experiments confirm that the restricted versions of the algorithms outperform the classical designs systematically. The statistical test included in the experimentation reports significant differences in all the cases, which validates the efficiency of our proposal.

Keywords: Combinatorial optimisation, linear ordering problem, local optima, insert neighbourhood

1. Introduction

The Linear Ordering Problem (LOP) is a classical combinatorial optimisation problem which has received the attention of the research community since it was studied for the first time by Chenery and Watanabe (1958). Garey and Johnson (1979) demonstrated that the LOP is an *NP-hard* problem, thereby evidencing the difficulty of solving

Email addresses: josu.ceberio@ehu.es (Josu Ceberio), alexander.mendiburu@ehu.es (Alexander Mendiburu), ja.lozano@ehu.es (Jose A. Lozano)

the LOP instances up to the optimality. However, due to its numerous applications in diverse fields such as archeology (Glover et al., 1972), economics (Leontief, 2008), graph theory (Charon and Hudry, 2007), machine translation (Tromble and Eisner, 2009) or mathematical psychology (Kemeny, 1959), we can find a wide variety of papers that have dealt with the LOP by means of exact, heuristic and metaheuristic strategies.

Among the exact methods, the most meaningful include Branch and Bound (Kaas, 1981; Charon and Hudry, 2006), Branch and Cut (Grötschel et al., 1984) and Cutting Plane algorithms (Mitchell and Borchers, 1996, 2000). These methods, as Schiavinotto and Stützle (2004) highlighted, behave competitively for instances from specific benchmarks with up to a few hundred columns and rows, however their computation time increases strongly with the size of the instances, and thus, it is not possible to solve large instances in a reasonable time span. Beyond the exact proposals, pioneering works proposed constructive heuristics (Chenery and Watanabe, 1958; Aujac, 1960; Becker, 1967). Such approaches were later outperformed by the advances produced in metaheuristic optimisation. Proof of this are the solutions based on Local Search (Kernighan and Lin, 1970; Chanas and Kobylanski, 1996), Genetic Algorithms (Charon and Hudry, 1998), Tabu Search (Laguna et al., 1999), Scatter Search (Campos et al., 2001), Variable Neighborhood Search (Garcia et al., 2005), Ant Colony optimisation (Chira et al., 2009), and recently Estimation of Distribution Algorithms (Ceberio et al., 2013).

According to a recent review of Martí et al. (2012), the Memetic Algorithm (MA) and the Iterated Local Search (ILS) proposed by Schiavinotto and Stützle (2004), 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 initialisation of the population or in the evolutionary process itself. On the other hand, the ILS is a strategy that iteratively applies a local search algorithm to a single solution. When the process gets trapped in a local optimum solution, the ILS applies a perturbation to the current solution, and continues with the optimisation process until a termination criterion is satisfied. Both algorithms include an efficient implementation of a greedy local search algorithm with the insert neighbourhood designed specifically to solve the LOP.

As seen for most of the combinatorial optimisation problems, the hardness of solving a specific instance is not only limited to the size of this, but also to other additional parameters, unknown in most cases. In this regard, the community has tried to better understand the characteristics of the LOP that determine the difficulty of the instances, and similarly, has tried to identify the features that could be useful to guide the algorithms throughout the optimisation process. In this sense, Schiavinotto and Stützle (2004) sketched out the properties that could somehow characterise the hardness of the LOP instances. The authors defined the *sparsity*, *variation coefficient (VC)*, *skewness* and *fitness distance correlation* as measures of instance hardness, and showed that real-life instances, which are apparently more difficult than artificial ones, present significant differences in *sparsity*, *VC* and *skewness* with respect to random benchmark instances. Nevertheless, the relation between the mentioned properties, and the suitability of the MA and the ILS to solve the LOP is not straightforward.

On the same research line, Betzler et al. (2011) published a detailed work on the parameterised complexity for intractable median problems, and particularly on the Kemeny ranking problem, which can be seen as a subclass of LOP. Although the parameterised complexity studied in the cited work is of great relevance, the analysis of Betzler

et al. (2011) stands on a specific property of the Kemeny, which does not hold for the general LOP, and thus, the extension of the parameterised complexity to the LOP is not straightforward.

The aforementioned works and the absence of a detailed work that performs an in-depth analysis of the LOP motivated this paper. In this work we study the properties that the optimal solutions of the LOP hold in the framework of local search algorithms, placing special emphasis on the position where the indexes are placed, and identifying the role of the associated matrix entries of the instance in the generation of local optima.

The paper is divided into two parts: first, we provide a detailed description of the LOP, introducing definitions and theorems that study the structure of the problem with respect to the optimality of the solutions in the context of local search algorithms. Particularly, we emphasise the influence that the position of the indexes that compound a solution have when generating local optima solutions. As a result of the theoretical study, a restricted version of the insert neighbourhood is proposed. This neighbourhood discards specific insert operations that involve moving indexes to positions at which they cannot generate local optima solutions. The theoretical analysis demonstrates that these insert operations are never the operations that most improve the solution in the neighbourhood.

The second part of the paper is devoted to demonstrate the validity of the *restricted insert neighbourhood*. In this sense, we develop a *restricted* version of the two best performing algorithms for the LOP: the MA and the ILS. Experimental results show that the restricted versions of the algorithms outperform the classical designs in the 90% and 93.3% respectively, obtaining the same results for the rest of the executions.

The remainder of the paper is organised as follows: in the next section, the definition of the LOP is described. In Section 3, the structural analysis of the LOP is introduced placing special emphasis on the contribution of the indexes to the objective function. Next, in Section 4 the optimality of the LOP solutions is described in the context of local search algorithms, and in particular for the insert neighbourhood system. Section 5 is devoted to investigating the basis for the restricted insert neighbourhood system. In order to demonstrate the validity of the introduced analysis, a complete experimental study is introduced in Section 6. Finally, some conclusions and ideas for future work are presented in Section 7.

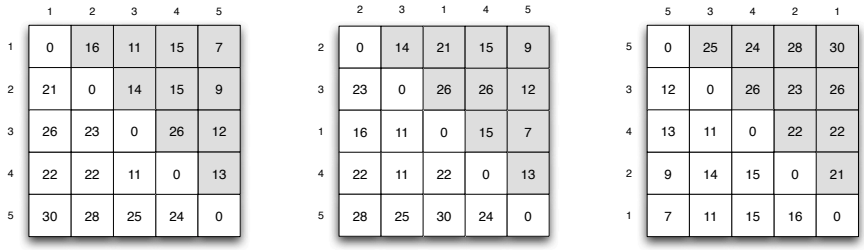
2. The Linear Ordering Problem

Given a matrix $B = [b_{kl}]_{n \times n}$ of numerical entries, the linear ordering problem 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 LOP 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[1]. This representation of the LOP is also known as the *triangulation problem of input-output matrices*. Although alternative representations of the problem can be found in Martí and Reinelt (2011) and Charon and Hudry (2007), due to the theoretical simplicity and readability of the exposed approach, in the remainder of the paper the *triangulation* representation will be considered.

Example 2.1. Let us introduce an example for a $n = 5$ LOP instance which will be used throughout the paper[2]. In Fig. 1, three different solutions, e, σ and σ^* are described. The initial matrix is represented by the identity permutation $e = (1, 2, 3, 4, 5)$ (see Fig. 1a), and its fitness, $f(e)$, is 138. The solution $\sigma = (2, 3, 1, 4, 5)$ introduces a different ordering of the indexes that provides a better solution than e (see Fig. 1b), $f(\sigma)$ is 158. The optimal solution for this example is given by $\sigma^* = (5, 3, 4, 2, 1)$ (see Fig. 1c), with fitness $f(\sigma^*) = 247$.



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

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

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

Fig. 1: Three different solutions of a $n = 5$ instance.

3. Analysis of the problem

In this section, we analyse the LOP by explaining the association between the indexes in σ and the arrangement of the b_{kl} entries of the matrix B. In addition, we describe the fitness variation that provokes changing the position of an index within σ, and the role of the b_{kl} entries in this regard. As necessary background to understand the latter content of the paper, in the following list we outline some meaningful properties of the LOP that define the association between the indexes in σ, and the b_{kl} entries in the B matrix.

For any permutation of indexes σ of size n and a matrix B of size $n \times n$:

- Every index $\sigma_i = k$, $i = 1, \ldots, n$, has associated $2(n-1)$ entries of B: $n-1$ from row k and $n-1$ from column k.

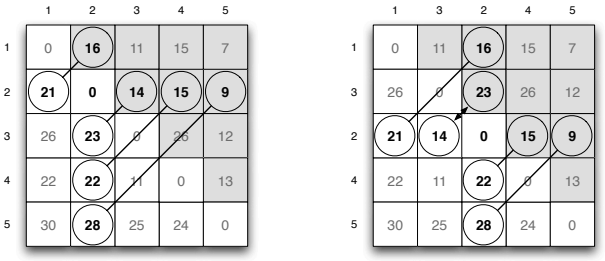
[1] From now on, σ will denote any permutation in $\mathbb{S}_n$, and e will stand for the identity permutation $(1, 2, \ldots, n)$ of size n. In addition k and l will denote the indexes within a permutation σ, and i, j and z will be used to identify the positions of σ.

[2] This example was extracted from Martí and Reinelt (2011).

- The set of associated entries of every index $\sigma_i = k$, $i = 1, \ldots, n$, can be organised in pairs, i.e. every entry in row k, $b_{k\sigma_j}$ (where $j = 1, \ldots, n$), has a pair in column k, $b_{\sigma_j k}$, symmetrically located with respect to the main diagonal.
- All the pairs of entries associated to index $\sigma_i = k$, $\{b_{k1}, b_{1k}\}, \ldots, \{b_{kn}, b_{nk}\}$, remain associated to this index regardless of its position and the position of the rest of the $n - 1$ indexes.
- Every entry $b_{\sigma_i\sigma_j}$ is associated to two indexes, σ_i and σ_j.
- For every pair $\{b_{\sigma_i\sigma_j}, b_{\sigma_j\sigma_i}\}$ of entries, one entry is always located above the main diagonal, and the other entry is located below, thereby bounding the best fitness contribution of this pair to $\max\{b_{\sigma_i\sigma_j}, b_{\sigma_j\sigma_i}\}$ in the best case, and to $\min\{b_{\sigma_i\sigma_j}, b_{\sigma_j\sigma_i}\}$ in the worst case.

In the remainder of the section, these characteristics and some extra definitions will be detailed with the help of illustrative examples.

Example 3.1. In this example, two different solutions are introduced, $e = (1, 2, 3, 4, 5)$ in Fig. 2a, and $\sigma = (1, 3, 2, 4, 5)$, in Fig. 2b. In both figures, the entries associated to the index 2 are highlighted in bold. We see that in spite of the different ordering, in both solutions the set of the entries associated to the index 2 are the same i.e. $(21, 14, 15, 9, 16, 23, 22, 28)$. Moreover, even though the position of index 2 is different in e and σ ($e_2 = 2$ and $\sigma_3 = 2$), the pairwise relation of the associated entries remains unchanged (see the circled indexes in Fig. 2).



(a) $e = (1, 2, 3, 4, 5)$, $f(e) = 138$. (b) $\sigma = (1, 3, 2, 4, 5)$, $f(\sigma) = 148$.

Fig. 2: Two different solutions for an instance of $n = 5$. Circled entries linked with edges denote the pairs of entries associated to index 2.

Checking the location of the associated entries of index 2 in σ, we note that the pair $(14, 23)$ has exchanged its positions in σ, being now 14 below the main diagonal and 23 above it. Prior to studying the implications that the movement of an index has in the fitness, in the following lines we first introduce a new term: the *contribution* of an index to the fitness function.

When index $k = 1, \ldots, n$ is ranked at position i in σ, i.e. $\sigma_i = k$, the contribution of index k to the objective function is given by the sum of the entries of column k in the rows

$\{\sigma_1, \ldots, \sigma_{i-1}\}$ and the sum of the entries of row k in the columns $\{\sigma_{i+1}, \ldots, \sigma_n\}$. That is to say, the previous $i-1$ indexes $\{\sigma_1, \ldots, \sigma_{i-1}\}$ and the posterior $n-i$ indexes $\{\sigma_{i+1}, \ldots, \sigma_n\}$ determine the contribution of the index k to the objective function. Formally, it is expressed as

$$c(\sigma, i) = \sum_{j=1}^{i-1} b_{\sigma_j \sigma_i} + \sum_{j=i+1}^{n} b_{\sigma_i \sigma_j} \tag{2}$$

Back to Example 3.1, due to the exchange of the pair $(14, 23)$, the contribution of index 2 has varied from 54 (16 + 14 + 15 + 9) in e (Fig. 2a) to 63 (16 + 23 + 15 + 9) in σ (Fig. 2b). In the case of index 3, its contribution has also increased, since the pair of $(14, 23)$ is associated to both indexes, 2 and 3. Inversely, in the cases of the indexes 1, 4 and 5, their contribution does not change from e to σ.

If we look carefully at Eq. 2, we realise that the contribution of index $\sigma_i = k$ is not actually determined by the specific ordering of the indexes in the previous and posterior positions of i, but it is determined by their grouping in those two sets of positions. As shown in Example 3.1, the contribution of the indexes 1, 4 and 5, does not change from e to σ since the grouping of the rest of the indexes into the previous and posterior sets of positions associated to the indexes 1, 4 and 5 was the same.

Proposition 3.1. *Given a solution σ, the contribution of the index $\sigma_i, i \in \{1, \ldots, n\}$, to the objective function $c(\sigma, i)$, is independent of the ordering of the previous indexes $\{\sigma_1, \ldots, \sigma_{i-1}\}$ and of the ordering of the posterior indexes $\{\sigma_{i+1}, \ldots, \sigma_n\}$.*

Example 3.2. This example illustrates how the contribution of index 3 is *independent* to the ordering of the previous and posterior sets of indexes. In Fig. 3a, the contribution of index 3, $c(e, 3)$ is 63 as a result of the sum (11+14+26+12). If we check the contribution of the index 3 in σ (see Fig. 3b), we see that it also sums 63, even though the indexes $\{1, 2\}$ and $\{4, 5\}$ have swapped their positions.

	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) $e = (1, 2, 3, 4, 5)$, $c(e, 3) = 63$.

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

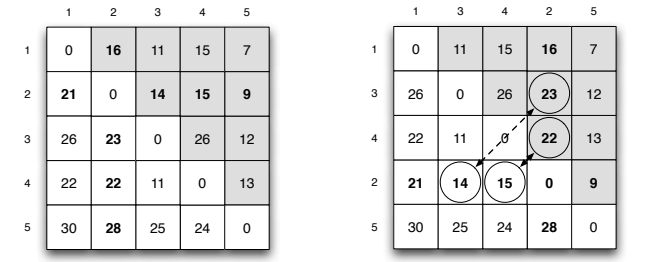
(b) $\sigma = (2, 1, 3, 5, 4)$, $c(\sigma, 3) = 63$.

Fig. 3: Illustration of the effect of swapping indexes at positions 1,2 and 4,5 with respect to the contribution of index 3 to the objective function.

In Proposition 3.1, we saw that the contribution of index k to the objective function is independent of the ordering of the indexes in the previous and posterior sets. But, what happens if an index $\sigma_j = l$ is moved from the previous set of indexes of σ_i to the

posterior set of indexes? Contrarily to the previous case, the contribution of the index σ_i, $c(\sigma, i)$, does not hold Proposition 3.1. At this point, it is worth remembering that each pair of entries $\{b_{\sigma_i\sigma_j}, b_{\sigma_j\sigma_i}\}$ in the matrix is associated to two indexes, σ_i and σ_j, and thus, any exchange of location of σ_i by definition affects the contribution to the fitness function of σ_i and σ_j. In fact, moving σ_i to position j, affects the contribution of all the indexes located between positions i and j. The example below illustrates the fitness variations produced by the movement of an index.

Example 3.3. Fig. 4a and Fig. 4b show matrix B according to solutions $e = (1, 2, 3, 4, 5)$ and $\sigma = (1, 3, 4, 2, 5)$. In this example we analyse the implications of moving index $e_2 = 2$ to position 4. Due to this modification, indexes 3 and 4, are shifted one position to the left, thus changing their contribution to the fitness function. Particularly, we observe that the pairs $\{14, 23\}$ and $\{15, 22\}$ associated to the indexes 2-3 and 2-4, have exchanged their positions. Therefore, the contribution of index 3, $c(e, 3)$ changes from



(a) $e = (1, \mathbf{2}, 3, 4, 5)$
$c(e, 2) = 54$, $c(e, 3) = 63$, $c(e, 4) = 69$.

(b) $\sigma = (1, 3, 4, \mathbf{2}, 5)$
$c(\sigma, 2) = 70$, $c(\sigma, 3) = 72$, $c(\sigma, 4) = 76$.

Fig. 4: Illustration of the effect of moving index 2 from position 2 to position 4, to the contribution of the indexes 2, 3 and 4 to the objective function. Numbers in bold denote the entries associated to index 2. Circled pairs of entries highlight exchanged entries.

63 (11 + 14 + 26 + 12) to 72 (11 + 26 + 23 + 12). Similarly, the contribution of index 4 changes from 69 (15 + 15 + 26 + 13) to 76 (15 + 26 + 22 + 13). And as regards index 2, its contribution also changes from 54 (16 + 14 + 15 + 9) to 70 (16 + 23 + 22 + 9). Note that the variation in the fitness contribution of index 2, is the sum of the variations of indexes 3 and 4.

4. The insert neighborhood and local optimality

As previously mentioned in the introduction, many of the most successful algorithms proposed for solving the LOP are partially or totally based on local search procedures. For that reason, we adopted the framework of local search algorithms in order to identify and extract meaningful information that could be used to improve their performance. It is well known that local search algorithms start from an initial solution and iteratively try to replace the current solution by a better one in a previously defined neighbourhood system (Blum and Roli, 2003). Among the different neighbourhood systems proposed in the literature for the LOP, most of the works (Schiavinotto and Stützle,

2004; Garcia et al., 2005) clearly point to the *insert* neighbourhood system as the best performing. For that reason, this is the system considered in this paper.

In what follows, we start by introducing some basic definitions about the insert neighborhood system and the local optimality of solutions.

Definition 4.1. Two solutions σ and σ' are *neighbors* under the insert neighborhood (N_I) if σ' is obtained by moving an index of σ to a different place. It is formally defined as

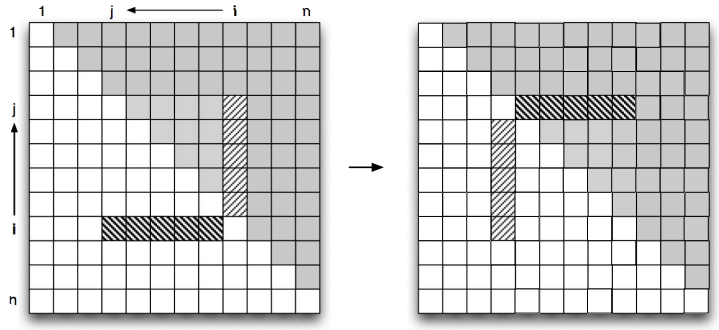
$$\sigma' \in N_I(\sigma) \Leftrightarrow \exists i, j \in \{1, \ldots, n\}, i \neq j \text{ s.t. } \begin{cases} (\sigma'_z = \sigma_z,\ z < i \wedge z > j) \wedge \\ (\sigma'_z = \sigma_{z+1},\ i \leq z < j) \wedge & \text{when } i < j \\ \sigma'_j = \sigma_i \\ \\ (\sigma'_z = \sigma_z,\ z < i \wedge z > j) \wedge \\ (\sigma'_z = \sigma_{z-1},\ i < z \leq j) \wedge & \text{when } i > j \\ \sigma'_i = \sigma_j \end{cases}$$

When an insert operation is performed, that is to move index σ_i to position j, some entries in the upper triangle are exchanged with their respective pairs in the lower triangle, as we showed in Section 3. In the following, we distinguish two different insert operation scenarios, and we highlight in each case the specific entries that are exchanged:

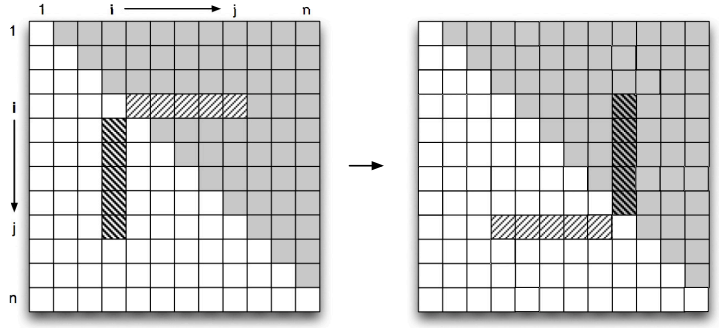
- $i > j$ (see Fig. 5a). The entries $\{b_{\sigma_j\sigma_i}, b_{\sigma_{j+1}\sigma_i}, \ldots, b_{\sigma_{i-1}\sigma_i}\}$ inside the upper triangle (light line pattern cells) are moved to positions $\{(j+1, j), (j+2, j), \ldots, (i, j)\}$ in the lower triangle, and the entries $\{b_{\sigma_i\sigma_j}, b_{\sigma_i\sigma_{j+1}}, \ldots, b_{\sigma_i\sigma_{i-1}}\}$ outside the upper triangle (dark line pattern cells) are moved to positions $\{(j, j+1), (j, j+2), \ldots, (j, i)\}$ in the upper triangle. The objective value of σ' (neighbour of σ), $f(\sigma')$, can be calculated by summing the objective value of $f(\sigma)$ and the difference of the entries exchanged, that is:
$$f(\sigma') = f(\sigma) + \sum_{z=j}^{i-1} (b_{\sigma_z\sigma_i} - b_{\sigma_i\sigma_z}) \tag{3}$$

- $i < j$ (see Fig. 5b). The entries $\{b_{\sigma_i\sigma_{i+1}}, b_{\sigma_i\sigma_{i+2}}, \ldots, b_{\sigma_i\sigma_j}\}$ inside the upper triangle (light line pattern cells) are moved to positions $\{(j, i), (j, i+1), \ldots, (j, j-1)\}$ in the lower triangle. Alternatively, the entries $\{b_{\sigma_{i+1}\sigma_i}, b_{\sigma_{i+2}\sigma_i}, \ldots, b_{\sigma_j\sigma_i}\}$ outside the upper triangle (dark line pattern cells) are moved to positions $\{(i, j), (i+1, j), \ldots, (j-1, j)\}$ in the upper triangle. Similarly to the previous case, the objective value of σ', $f(\sigma')$ can be calculated as:
$$f(\sigma') = f(\sigma) + \sum_{z=i+1}^{j} (b_{\sigma_i\sigma_z} - b_{\sigma_z\sigma_i}) \tag{4}$$

As Fig. 5 illustrates, the pairs of entries in line pattern cells (dark and light), are the only entries that are exchanged because of the insert operation. The rest of the entries remain on the same side of the main diagonal as they were before the insert operation. As described in Section 3, every σ_i has associated $\{b_{\sigma_i 1}, b_{1\sigma_i}\}, \ldots, \{b_{\sigma_i n}, b_{n\sigma_i}\}$ pairs of entries, and so we introduce the term *vector of differences* $(b_{\sigma_i 1} - b_{1\sigma_i}, \ldots, b_{\sigma_i n} - b_{n\sigma_i})$ where each value in the vector describes the fitness variation that a specific pair of



(a) $i > j$. Before and after the insert operation.



(b) $i < j$. Before and after the insert operation.

Fig. 5: The exchange of entries produced by moving index σ_i to position j. Two different scenarios are considered: $i > j$ and $i < j$. Figures on the left illustrate the cells implicated in the insert operation. Figures on the right show the new arrangement of the entries because of the insert operation. Cells in grey denote the entries of the matrix that sum to the objective function.

entries produces when it is exchanged because of a movement in σ_i. In what follows, we will see how the vector of differences associated to each index will be essential in order to determine if an index generates local optima solutions at a given position.

Definition 4.2. A solution σ^* is a *local optimum* for the insert neighbourhood if all neighbouring solutions σ have a lower fitness value.

$$\forall \sigma \in N_I(\sigma^*) \;\; f(\sigma^*) \geq f(\sigma)$$

Therefore, in the insert neighbourhood system, a solution is considered local optima, if and only if among all the possible insert operations, there is no movement that outperforms the current solution. Taking into account what was exposed Definition 4.1, given a solution σ and the neighbouring solution σ' obtained moving index k from position, $f(\sigma')$ can be computed by recalculating the fitness contribution of the index k in the new position, and summing the variation to $f(\sigma)$. This way, we state that a solution

is local optima in the insert neighbourhood if any insert operation performed over σ_i, being $i = 1, \ldots, n$, does not increase the contribution of σ_i.

Example 4.1. Let us consider the entries associated to the index at position 2, $\sigma_2 = 2$ with $c(\sigma, 2) = 54$. Fig. 6 illustrates the vector of differences $(b_{\sigma_1 2} - b_{2\sigma_1}, \ldots, b_{\sigma_n 2} - b_{2\sigma_n})$ of index 2 when performing all the possible insert operations. Besides, the contribution of index 2 is also given for each case.

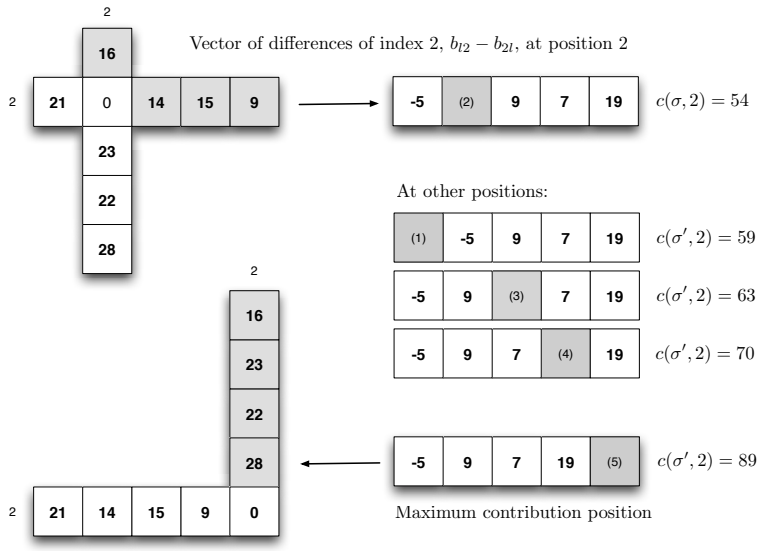


Fig. 6: The vector of differences of index 2, and the contribution of this index to the objective function for each possible insert operation given a specific σ.

As we can see, the insert operation that moves index 2 to position 5 is the one that maximises the contribution of this index, from 54 to 89. Note that in order to find the position at which the contribution of an index k is maximised, we need to find that arrangement of the vector of differences associated to index k such that the sum of the values in the positions $\{1, \ldots, i-1\}$ is maximised, and the sum of the values in the positions $\{i+1, \ldots, n\}$ is minimised[3].

The property below studies the arrangement of the vector of differences of each index when σ is a local optimum solution.

Property 4.1. Given a local optimum solution σ^* for the insert neighbourhood, then for every index σ_i^*, $i = 1, \ldots, n$, all the partial sums of the differences between the associated

[3]Note that the vector of differences is calculated by subtracting the entries in the column from the entries in the row

entries located before i are positive:

$$\sum_{j=i-1}^{z} (b_{\sigma_j^*,\sigma_i^*} - b_{\sigma_i^*,\sigma_j^*}) \geq 0, \qquad z = i-1, \ldots, 1 \tag{5}$$

and all the partial sums of the differences between the associated pairs located after i are negative:

$$\sum_{j=i+1}^{z} (b_{\sigma_j^*,\sigma_i^*} - b_{\sigma_i^*,\sigma_j^*}) \leq 0, \qquad z = i+1, \ldots, n \tag{6}$$

Back in Example 4.1, it can be observed in Fig. 6 that only the insert operation that moves index 2 to position 5 organises the vector of differences in the way that complies with Eq. 5 and 6: all the partial sums from position 4 to 1 are positive: $19 \geq 0$, $7 + 19 \geq 0$, $9 + 7 + 19 \geq 0$, $-5 + 9 + 7 + 19 \geq 0$.

Due to the vector of differences induced by the ordering of the remaining indexes in σ, index 2 only complies with Eq. 5 and 6 at position 5. However, changing the ordering of σ, the ordering of the vector of differences changes too, and thus, Eq. 5 and 6 might not longer hold. For illustrative purposes, let us consider a solution $\sigma' = (5, 3, 4, 2, 1)$ that has been obtained exchanging the position of the indexes 1 and 5 in σ. The vector of differences associated to index 2, according to σ', is $(19, 9, 7, *, -5)$[4] which indeed complies with Eq. 5 and 6. Moreover, in the case of σ', index 2 complies with the previous equations at either position 4 or position 5.

Nonetheless, there exist positions at which index 2 cannot generate a local optimum independently of the position of the remaining indexes, as is the case of the positions 1, 2 and 3. Looking at the vectors of differences associated to index 2 in Fig. 6 in positions 1, 2 and 3, it can be seen that no ordering of the values in the vector complies with Eq. 5 and 6.

In view of this property, in the next section, we propose a more efficient insert neighbourhood where we discard those insert operations that move indexes to positions at which they cannot generate local optima solutions independently of the remainder indexes.

5. The restricted insert neighborhood

In the previous section, we described the properties that the indexes within a solution σ need to comply with, in order for σ to be a local optima solution (Property 4.1). The next obvious step would consist of identifying the specific positions where the indexes generate local optima solutions. An in-depth analysis in this sense, however, suggests that such an approach is NP-hard, since for each index at each position that complies with Property 4.1, we need to check (n-1)! permutations. Nevertheless, there are some positions at which indexes do not generate a local optima regardless of the ordering of the rest of the indexes, and contrary to the previous case, to detect those positions is straightforward.

[4] $*$ denotes the position at which index 2 was placed, the position 4.

Based on the vector of differences associated to the indexes, in this section we analyse the basis for discarding the positions at which indexes cannot generate local optima solutions. As a result of the analysis, we propose the *restricted insert neighbourhood*, which discards some insert operations that move indexes to the positions where local optima solutions are not generated.

In order to illustrate the process of identifying the positions where an index cannot generate a local optima, we start studying the trivial cases, i.e. the boundary cases at which an index k is located either first or last:

- In order to discard the first position for index k, $\sigma_1 = k$, we need to demonstrate that no arrangement of the vector of differences associated to index k complies with Eq. 6. In order to prove that index k does not comply with that condition, independently of the order of the rest of indexes, it is enough to see that Eq. 6 is false when $\sigma_1 = k$ and $z = n$. Note that the result of the sum operator in the previous equation is independent of any ordering of the $\{\sigma_2, \ldots, \sigma_n\}$ indexes, since the full vector of differences $(b_{\sigma_1 k} - b_{k\sigma_1}, \ldots, b_{\sigma_n k} - b_{k\sigma_n})$ is considered in the sum as a result of $z = n$.

- Similarly to the previous case, in order to discard the last position for index k, $\sigma_n = k$, we need to demonstrate that no arrangement of the vector of differences associated to index k complies with Eq. 5. In order to prove that index k does not comply with that condition, independently of the order of the rest of indexes, it is enough to see that Eq. 5 is false when $\sigma_n = k$ and $z = 1$.

Beyond the boundary cases, in order to state whether an index k does not generate a local optima at a given position i, $\sigma_i = k$, we need to check that a partition of indexes does not exist which locates a group of values of the vector of differences in the positions $\{1, \ldots, i-1\}$ and another group in $\{i+1, \ldots, n\}$, such that the arrangement of the vector complies with Eq. 5 and Eq. 6 at the same time.

In this regard, we propose a simple algorithm that starts sorting in descending order the vector of differences associated to the index k. Since our aim is to discard positions, the second step consists of checking whether the most favourable partitioning of the vector of differences complies with Eq. 5 and Eq. 6 when $\sigma_i = k$. The allocation procedure consists of placing the largest value in the vector at position $i-1$, second largest at position $i-2$ and so on. On the other hand, the lowest value is placed at position $i+1$, the next lowest at $i+2$, following the same procedure as for the largest values in the vector of differences. Let us denote as σ' the solution induced by the new ordering of the vector of differences. Then, index k does not generate a local optima solution at position i if the following equation:

$$\sum_{z=i-1}^{1} (b_{\sigma'_z k} - b_{k\sigma'_z}) < 0 \text{ or } \sum_{z=i+1}^{n} (b_{\sigma'_z k} - b_{k\sigma'_z}) > 0 \tag{7}$$

is true.

Extending this procedure to the whole set of indexes in σ for all the positions, we calculate a binary matrix, called *restrictions matrix R*, where the entries with 0 represent the positions at which indexes do not generate a local optima solution. Algorithm 1 summarises the pseudocode of the proposed algorithm.

Algorithm 1 Algorithm to calculate the *restrictions matrix R*

```
1: Input: The matrix B of entries.
2: for k = 1 to n do
3:    diffVector = CalculateVectorDifferences(k,B);
4:    sortedDifferences = SortDescendingOrder(diffVector);
5:
6:    for i = 1 to n do
7:       beforeSum=0;
8:
9:       for z = i − 1 to 1 do
10:         beforeSum+=sortedDifferences[z];
11:      end for
12:      afterSum=0;
13:
14:      for z = i + 1 to n do
15:         afterSum+=sortedDifferences[z];
16:      end for
17:
18:      if beforeSum≥ 0 and afterSum≤ 0 then
19:         R[k][i] = 1;
20:      else
21:         R[k][i] = 0;
22:      end if
23:   end for
24: end for
25: Output: The restrictions matrix R.
```

Example 5.1. Fig. 7 illustrates the algorithm we propose to identify the positions where an index, in this example index 2, cannot generate a local optima solution. As can be

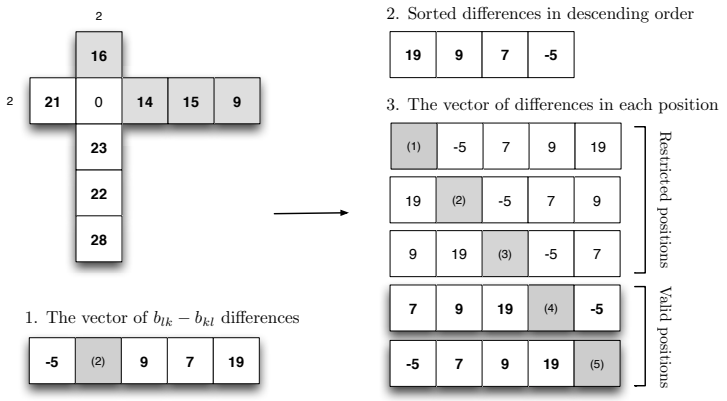


Fig. 7: The vector of differences associated to index 2, the sorted vector of differences, and the arrangement of the differences in order to identify the positions for which index 2 cannot generate local optima solutions.

observed in Fig. 7, a solution is local optima, if and only if index 2 is ranked at positions 4 or 5, since in the rest of the positions, even with the most favourable arrangement of the vector of differences, Eq. 7 is not complied. Extending the application of the algorithm applied over index 2 to the rest of the indexes, we then calculate the restrictions matrix R a binary matrix as shown in Fig. 8.

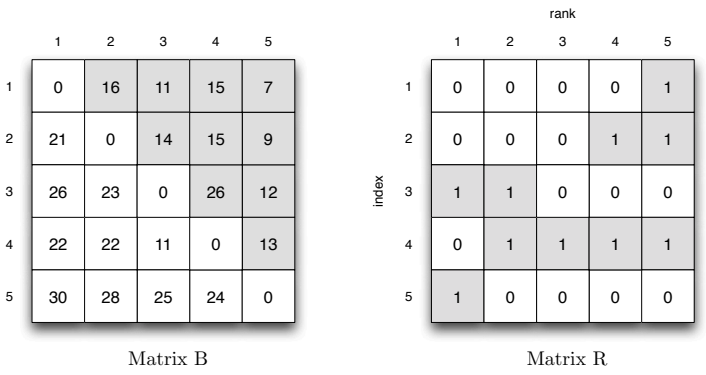


Fig. 8: Figure on the left shows the $n = 5$ instance used throughout the paper, and the corresponding restrictions matrix R is introduced on the right. Each position i, k of the matrix R indicates whether index k can generate a local optima when located at i; 1 if true, and 0 if not.

In the view of the restrictions matrix in Fig. 8, solving the toy LOP example used throughout the paper is trivial, since some indexes can generate local optima solutions only at one position, as is the case of indexes 1 and 5. However, once these indexes are fixed, index 2 and 3 are left with a position, 2 and 4. And so, index 4 must be placed in position 3. The global optimum solution of the example is $\sigma^* = (5, 3, 4, 2, 1)$.

For non-toy instances, the number of restricted positions is lower, but still significant. For this reason we propose to improve the performance of local search based methods by introducing a restricted version of the insert neighbourhood, called the *restricted insert neighbourhood*. This neighbourhood discards the insert operations that move indexes to positions at which they do not generate local optima solutions.

At first glance, it is obvious that using the restricted insert neighbourhood will not necessarily outperform the classical greedy local search method. It is well known that, sometimes, bad solutions can lead the algorithm to more promising areas of the search space than fitter solutions. However, in the case of the LOP, we discovered that the insert operation that is chosen in a greedy local search, the one under which the largest improvement is given, is never a restricted operation according to the matrix R. The theorem below formalises this result.

Theorem 5.1. *Given a non local optima solution σ, for every index σ_i, $i = 1, \ldots, n$, the insert movement that maximises its contribution to the fitness function is not given in a restricted position of σ_i.*

Proof. In order to demonstrate that the theorem is held, we will prove that the inverse scenario cannot be true, i.e., let us assume that there exists an insert operation that

moves an index σ_i to a restricted position j which improves the solution the most. This means that the maximum contribution of the index σ_i is given at position j. If the maximum contribution position of index σ_i is at a restricted position, then Property 4.1. should hold, implying that Eq. 7 is true for σ_i at position j. Therefore, position j cannot be restricted to index σ_i if it is the maximum fitness contribution position. □

As a result of Theorem 5.1, the restricted insert neighbourhood, is a subsystem of the insert neighbourhood, which has two meaningful properties:

- Given a solution σ and the restrictions matrix R, the size of the restricted insert based neighbourhood $N_R(\sigma)$ is reduced to

$$|N_R(\sigma)| = (n-1)^2 - \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} 1_{[R[i][j]==0]}$$

- Due to the reduction of the neighbourhood, the greedy local search that implements the restricted insert neighbourhood will perform fewer operations to reach a specific solution than the local search with the standard insert neighbourhood.

- Given a maximum number of evaluations, the greedy local search that implements the restricted insert neighbourhood will explore more solutions in the search space than that with the standard insert neighbourhood. As a consequence, the restricted version of the local search will presumably outperform the classical version.

6. Experimentation

In order to demonstrate the improvement of using the *restricted insert neighbourhood*, we have applied our neighbourhood proposal to the best performing algorithms proposed in the literature: Memetic Algorithm and Iterated Local Search (proposed by Schiavinotto and Stützle (2004)). Both algorithms include in their procedure an optimised implementation of a greedy local search algorithm for the LOP. The goal in this experimentation is to analyse the improvement obtained using the restricted insert neighbourhood instead of the classical insert neighbourhood.

Due to the lack of challenging benchmarks in the literature, Schiavinotto and Stützle (2004) proposed a new benchmark, the *extended* LOLIB (xLOLIB), which was generated by randomly sampling the instances of the LOLIB benchmark. Particularly, they generated 39 instances of size 150 and 39 instances of size 250. In addition to these instances, we generated an extra benchmark to include in this experimentation, xLOLIB2, with 200 instances of sizes 300, 500, 750 and 1000 (50 instances of each size) following the same procedure used for generating the xLOLIB benchmark.

Both source codes, MA and ILS, were obtained from the authors, and so the restricted insert neighbourhood was directly implemented on the original code (written in C), adding only the necessary code to calculate the restrictions matrix and implement the restricted insert neighbourhood. The experimentation was conducted on a cluster of 20 nodes, each of them equipped with two Intel Xeon X5650 CPUs and 48GB of memory.

In order to analyse the contribution of the restricted insert neighbourhood as fair as possible, we ran the original implementations of MA and ILS, and their *restricted* versions MA_r and ILS_r for three different maximum numbers of evaluations, $1000n^2$, $5000n^2$ and $10000n^2$ (n denotes the size of the instance). The evaluation numbers were set without performing any previous experimentation.

Each algorithm-instance pair was run 20 times and the average fitness of the best solutions obtained was calculated. Due to the large size of the results-tables obtained from the conducted experiments, the results have been summarised in Table 1 divided in three groups according to the different stopping criterion and the size of the instances. Besides, the results have been presented as the number of times the restricted versions of the algorithms beat the classical versions. Next to the results, within parentheses, we show the number of instances for which the same results on both versions, restricted and non-restricted, were obtained[5]. Remember that the restricted versions will at least equal the results of the classical proposal.

Table 1: Comparison of the results of MA with MA_r, and ILS with ILS_r for the xLOLIB and xLOLIB2 benchmark LOP instances (278 instances). The results have been grouped with respect to the maximum number of evaluations used, i.e. $1000n^2, 5000n^2$ and $10000n^2$. The results summarise the number of instances for which the average results of 20 repetitions of the restricted version of the algorithms (MA_r and ILS_r) outperformed the average results of the classical version of the algorithms (MA and ILS). Numbers in parentheses denote the number of instances for which the average results for both algorithms, the classical and the restricted, were equal.

$1000n^2$ evals.	xLOLIB		xLOLIB 2				Total
	150	250	300	500	750	1000	
MA_r vs. MA	35 (4)	31 (8)	39 (11)	43 (7)	41 (9)	37(13)	226 (52)
ILS_r vs. ILS	37 (2)	37 (2)	49 (1)	48 (2)	50 (0)	50(0)	271 (7)

$5000n^2$ evals.	xLOLIB		xLOLIB 2				Total
	150	250	300	500	750	1000	
MA_r vs. MA	37 (2)	39 (0)	50 (0)	49 (1)	44 (6)	44(6)	263 (15)
ILS_r vs. ILS	38 (1)	36 (3)	50 (0)	45 (5)	46 (4)	47(3)	262 (16)

$10000n^2$ evals.	xLOLIB		xLOLIB 2				Total
	150	250	300	500	750	1000	
MA_r vs. MA	39 (0)	34 (5)	43 (7)	50 (0)	50 (0)	49(1)	265 (13)
ILS_r vs. ILS	33 (6)	37 (2)	46 (4)	42 (8)	43 (7)	45(5)	246 (32)

In the view of the results, the restricted algorithms MA_r and ILS_r outperform the classical implementations in almost all the evaluated cases. With respect to MA_r, it outperforms MA in 81.2%, 94.6% and 95.3% of the instances for $1000n^2, 5000n^2$ and $10000n^2$ maximum numbers of evaluations. Similarly, ILS_r outperforms the ILS in 97.4%, 94.2% and 88.4% of the instances.

In order to assess whether there exist statistical differences among the results, we applied a nonparametric Wilcoxon test to the average results obtained by the pair MA - MA_r, and the pair ILS - ILS_r for each size of the instances. A level of significance $\alpha = 0.05$

[5]Supplementary results, original source codes, instances, and extended material of the experiments can be obtained from http://www.sc.ehu.es/ccwbayes/members/jceberio/LOP.html.

was set. The statistical test reported significant differences between the algorithms for all the sets of instances and for the three maximum number of evaluations. The p-values obtained for the pair MA - MA$_r$ were in the worst cases 1.23×10^{-06}, and 5.6×10^{-07} for ILS - ILS$_r$.

7. Conclusions & Future Work

In this paper we introduced a detailed theoretical study of the LOP in the context of local search algorithms. Based on this study we presented a method that allows to extract static information about the problem and incorporate it to improve the performance of local search algorithms. Particularly, we developed a method to detect the positions where the indexes cannot appear in local optima solutions of the insert neighbourhood. As a result of this study, an improved version of the insert neighbourhood system, called *restricted insert neighbourhood* was proposed, in which the insert operations that lead indexes to restricted locations are discarded.

In order to demonstrate the efficiency of the restricted insert neighbourhood, we applied this neighbourhood to the best performing state-of-the-art algorithms for LOP: Memetic Algorithm and Iterated Local Search. Average fitness values of the best solutions from 20 repetitions of MA, ILS, MA$_r$, and ILS$_r$ were calculated for three different maximum numbers of evaluations on a benchmark of 278 instances.

Conducted experiments showed that the restricted version of the algorithms systematically outperforms the classical versions. From 278 instances tested (xLOLIB - 78 instances) and (xLOLIB2 - 200 instances), MA$_r$ outperformed MA in 90% of the cases, and ILS$_r$ improved ILS in 93.3%. The Wilcoxon statistical test confirmed the behaviour reported by the experiments, indicating that MA$_r$ and ILS$_r$ are significantly better than the classical version of the algorithms for all the studied sets of instances.

Studying a problem and extracting information that can be used to guide the optimisation process of an algorithm is an interesting task that, in most of the cases, requires thorough research. In this work, we just scratched over the surface of the linear ordering problem with the *restricted insert neighbourhood*, and therefore, there are many issues that deserve deeper analysis. An easy extension of the theoretical study presented in this paper is that of the relative ordering of the indexes in which some adjacent orderings of indexes cannot generate local optima solutions due to the associated entries that share the consecutive indexes.

Finally, how to exploit the extracted knowledge in the most advantageous way is another challenging task that, in this case, has been addressed by discarding solutions within a neighbourhood, and proposing a restricted version of it. However, we find other interesting applications of the restrictions matrix such as the implementation of constructive heuristics to initialise metaheuristic procedures, or the implementation of guided mutations within Evolutionary Algorithms.

Acknowledgments

The authors gratefully acknowledge R. Martí and A. Duarte for providing essential material for this work, and T. Schiavinotto and T. Stützle for providing their implementation of the Memetic Algorithm and Iterated Local Search algorithms. 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), 2011-CIEN-000060-01 (Gipuzkoako Foru Aldundia) and CRC-Biomarkers Project 6-12-TK-2011-014 (Bizkaiko Foru Aldundia). Josu Ceberio holds a grant from Basque Government.

Aujac, H., 1960. La hiérarchie des industries dans un tableau des échanges interindustriels. Revue économique 11 (2), 169–238.

Becker, O., 1967. Das helmstädtersche reihenfolgeproblem – die effizienz verschiedener näherungsverfahren -. In: Computers Uses in the Social Sciences, Berichteiner Working Conference. Vienna.

Betzler, N., Guo, J., Komusiewicz, C., Niedermeier, R., Jul. 2011. Average parameterization and partial kernelization for computing medians. Journal of Computer and System Sciences 77 (4), 774–789.

Blum, C., Roli, A., Sep. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35 (3), 268–308.

Campos, V., Glover, F., Laguna, M., Martí, R., 2001. An experimental evaluation of a scatter search for the linear ordering problem. Journal of Global Optimization 21 (4), 397–414.

Ceberio, J., Mendiburu, A., Lozano, J. A., 2013. The Plackett-Luce Ranking Model on Permutation-based Optimization Problems. In: IEEE Congress on Evolutionary Computation. pp. 494 – 501.

Chanas, S., Kobylanski, P., 1996. A new heuristic algorithm solving the linear ordering problem. Computational Optimization and Applications 6 (2), 191–205.

Charon, I., Hudry, O., 1998. Lamarckian genetic algorithms applied to the aggregation of preferences. Annals of Operations Research 80 (0), 281–297.

Charon, I., Hudry, O., 2006. A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments. Discrete Applied Mathematics 154 (15), 2097 – 2116.

Charon, I., Hudry, O., Mar. 2007. A survey on the linear ordering problem for weighted or unweighted tournaments. 4or 5 (1), 5–60.

Chenery, H. B., Watanabe, T., October 1958. International comparisons of the structure of production. Econometrica 26 (4), 487–521.

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

Garcia, C. G., Pérez-Brito, D., Campos, V., Martí, R., 2005. Variable neighborhood search for the linear ordering problem. Computers & Operations Research 33 (12), 3549 – 3565.

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

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

Grötschel, M., Jünger, M., Reinelt, G., 1984. A cutting plane algorithm for the linear ordering problem. Operations research 32 (6), 1195–1220.

Kaas, R., 1981. A branch and bound algorithm for the acyclic subgraph problem. European Journal of Operational Research 8 (4), 355 – 362.

Kemeny, J. G., 1959. Mathematics without numbers. Daedalus 88, 577–591.

Kernighan, B. W., Lin, S., 1970. An Efficient Heuristic Procedure for Partitioning Graphs. The Bell system technical journal 49 (1), 291–307.

Laguna, M., Marti, R., Campos, V., 1999. Intensification and diversification with elite tabu search s26olutions for the linear ordering problem. Computers & Operations Research 26, 1217–1230.

Leontief, W., 2008. Input-Output Economics. Cambridge University Press.

Martí, R., Reinelt, G., 2011. The linear ordering problem: exact and heuristic methods in combinatorial optimization. Vol. 175. Springer.

Martí, R., Reinelt, G., Duarte, A., Apr. 2012. A benchmark library and a comparison of heuristic methods for the linear ordering problem. Comput. Optim. Appl. 51 (3), 1297–1317.

Mitchell, J., Borchers, B., 1996. Solving real-world linear ordering problems using a primal-dual interior point cutting plane method. Annals of Operations Research 62 (1), 253–276.

Mitchell, J., Borchers, B., 2000. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In: Frenk, H., Roos, K., Terlaky, T., Zhang, S. (Eds.), High Performance Optimization. Vol. 33 of Applied Optimization. Springer US, pp. 349–366.

Schiavinotto, T., Stützle, T., 2004. The linear ordering problem: instances, search space analysis and algorithms. Journal of Mathematical Modelling and Algorithms.

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