A diversity-aware memetic algorithm for the linear ordering Problem

Lázaro Lugo · Carlos Segura · Gara Miranda

Received: 10 January 2022 / Accepted: 25 September 2022 / Published online: 16 October 2022 © The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
The Linear Ordering Problem (LOP) is a very popular NP-hard combinatorial optimization problem with many practical applications that may require the use of large instances. The Linear Ordering Library (LOLIB) gathers a set of standard benchmarks that are widely used in validating solvers for the LOP. Among them, xLOLIB2 collects some of the largest and most challenging instances in the current literature. In this work, we present new best-known solutions for each of the 200 complex instances that comprise xLOLIB2 and for the other 93 instances of the benchmarks with smaller sizes. This important advance in the field of LOP has been made possible due to the development of a novel memetic algorithm (MA) that was designed by taking into account some of the weaknesses of state-of-the-art LOP solvers. In particular, one of the keys to success is that the novel proposal allows for a gradual shift from exploration to exploitation. This is done using the novel Best Non-Penalized (BNP) replacement strategy. BNP selects the survivors by taking into account the quality, the Spearman’s footrule distance, the stopping criterion, and the elapsed period of execution simultaneously. The novel diversity-aware proposal is called the memetic algorithm with explicit diversity management (MA-EDM) and extensive comparisons against state-of-the-art techniques provide insights into the reasons for the superiority of MA-EDM.

Keywords  Linear ordering problem · Diversity-aware optimizers · Memetic algorithms · Combinatorial optimization

1 Introduction
The Linear Ordering Problem (LOP) is a classical combinatorial optimization problem that arose in the field of economics. It has applications in other research areas, such as machine learning and logistics. The problem was introduced in 1936 by Leontief [1] and formalized in 1958 by Chenery and Watanabe [2], but it was not until decades later that Garey and Jonhson [3] demonstrated that LOP is an NP-hard problem. Although the problem has multiple applications and formulations, it is quite common to define it as a matrix triangulation problem.

Given a matrix $M_{n \times n} = (m_{ij})$, the triangulation problem is to determine a simultaneous permutation $\sigma$ of the rows and columns of $M$ such that the sum of the entries above the main diagonal is maximized (or equivalently, the sum of subdiagonal entries is minimized) [4]. Therefore, the LOP can be formulated as finding a permutation $\sigma$ that maximizes the following equation:

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

Several exact and approximate solvers have been proposed for this combinatorial optimization problem [5]. The interest in this problem has continued over the years, resulting in research compilations such as [4]. Until now, different algorithms have been devised that yield very promising solutions for the LOP instances addressed in [4]. Among these proposals, memetic algorithms (MAs) have proven to be very effective in the search for improved solutions to the LOP
As described in the next section, most trajectory- and population-based state-of-the-art approaches to the LOP rely on restarting procedures or other mechanisms to regain diversity or to abandon oversampled regions in an effort to avoid stagnation and premature convergence. In addition, they usually include problem-dependent information to exploit the particularities of the problem’s structure. The performance of schemes based on recovering diversity has been improved in several large-scale problems by methods that explicitly manage diversity [7], and methods that relate the amount of diversity to the elapsed period of execution and the stopping criterion have been particularly successful [8]. However, this last design principle has not been adopted in current LOP solvers. The hypothesis of this research is that there is still a big room for improvement in the design of LOP solvers for large instances, and thus, the solutions that are used as a reference when validating LOP solvers might be significantly enhanced. Knowing near-optimal values is important to properly measure the capabilities of different solvers. Thus, this research aims to design a novel long-term solver that is capable of obtaining new best-known solutions for the most challenging instances of the LOP. Given the promising behavior of MAs and the lack of population-based LOP solvers that establish a dependency between the amount of diversity maintained in the population and the elapsed execution period and the stopping criterion, a novel MA that adopts this design principle is applied.

The novel diversity-aware memetic algorithm proposed in this research is called the memetic algorithm with explicit diversity management (MA-EDM). MA-EDM combines an effective local search method and well-tested permutation-based genetic operators with a novel diversity-aware survivor selection strategy, the Best Non-Penalized replacement strategy, which aims to gradually shift from exploitation to exploration as evolution progresses. This is done using a replacement method that considers the quality, the Spearman’s footrule distance, the stopping criterion, and the elapsed period of execution simultaneously. As shown in the experimental validation, a very large number of new best-known solutions were attained for the most popular instances used in the literature. This is quite an important achievement, especially considering that these benchmarks have been available for decades and that many different kinds of methodologies have been used to handle them. It is also worth noting that our proposal does not incorporate some of the problem-dependent knowledge about the structure of the problem that is taken into account in the most recent LOP solvers. Thus, this research also shows that by applying proper general metaheuristic design principles, simpler and more effective LOP optimizers can be developed.

In the following section, the state-of-the-art in the LOP area is summarized by introducing a chronological description of the most promising approaches and their main features. Afterwards, our novel diversity-aware memetic algorithm is described in detail. Finally, the experimental analysis is presented, and the main conclusions are drawn. The experimental validation takes into account the best-performing metaheuristics proposed in the literature with the aim of showing that the advances are not only due to the fact that long-term executions are carried out. The advantages of MA-EDM are quite remarkable. Experiments to shed some light on the reasons behind the superiority of MA-EDM are also included.

## 2 Literature review

A wide variety of approaches to deal with the LOP have been proposed [5], including exact methods like branch-and-bound or cutting planes, specific heuristics and multiple metaheuristic strategies. In recent years, metaheuristics have gained more attention, especially for dealing with the most recent benchmarks, which include quite complex and large instances [6,9]. The most relevant metaheuristics developed for the LOP are described briefly below.

In 1999, Laguna et al. [10] proposed a Tabu Search (TS) approach which incorporates strategies for search intensification and diversification. Two different definitions of basic neighborhoods and path relinking are introduced for search intensification, while the diversification relies on ad-hoc short-term and long-term components. The short-term component is focused on modifying items that have been changed infrequently. Meanwhile, the long-term component is used to perform restarts by generating solutions that are distant from the best solutions found so far. This approach improved—at the time—the state-of-the-art heuristics [11].

As an alternative, Campos et al. [12] proposed a Scatter Search scheme that takes into account diversity in two of its components. First, in order to generate initial diverse solutions, GRASP constructions were designed with a greedy function that combines attractiveness and diversity. Second, the reference set selects half of its members by quality and half of its members by maximizing a measure of diversity. Regarding the improvement method, it is also based on the two neighborhoods proposed in [10], while an ad-hoc combination method based on votes is used.

Later, Schiavinotto and Stützle [9] performed a search-space analysis of the available LOP instances. Depending on certain characteristics of the matrices, such as variability, number of zeros or symmetry, they realized that those based on real data are very different from random ones. Moreover, the instances [13] were too small to pose a real challenge to state-of-the-art metaheuristic approaches, and even to exact algorithms, so an additional set of large, random, real-life like instances—denoted as eXtended LOLIB (xLOLIB)—was generated. The most complex instances present a low
correlation between fitness and distance to optimal solutions, which suggests that exploring distant regions is important to performance. As a result, an MA that includes an Iterated Local Search (ILS) for intensification and an adaptive restart strategy for diversification was proposed. For the local search, the neighborhoods were defined through the interchange or insert moves, and the swap operation of the current permutation. In addition to these neighborhoods, the sorting-reversing procedure introduced by Chanas and Kobylanski [11] was also applied. Since the performance of an MA may depend strongly on the crossover operator, four different operators were tested. The most promising results were obtained with Cycle Crossover (CX) and Order Crossover (OB). The experimental study showed that the MA resulted in a much more robust performance; in particular, the restarting mechanism was key to the success.

Trajectory-based heuristics were further advanced by developing variants of Variable Neighborhood Search (VNS) [14]. Several variants that include different intensification and diversification approaches were proposed. Experimental analysis shows that the performance of VNS improves when combined with short-term memory TS and diversification schemes. While proper solutions were attained in short-term executions, the solutions were not as promising as those output by MAs.

In 2012, Martí, Reinelt and Duarte [4] published an essential survey in the field of the LOP. Their work offers a comprehensive study of the main existing heuristic and metaheuristic approaches for the LOP at the time. For the heuristic methods, the authors studied some classical construction algorithms and some improvement algorithms based on inserts—moves or exchanges—or on local search. The experimental study also included some multi-start methods [11], as well as all the metaheuristics already introduced here: Tabu Search [10], GRASP [12], Scatter Search [12], Genetic Algorithms [9], MA [9], ILS [9], and VNS [14]. The experimental study also included a Simulated Annealing based on the noising algorithm by Charon and Hudry [15,16]. For their analysis, the authors also compiled all the existing—and widely used in related literature—LOP benchmarks to release a new version of the Lolib library. Their statistical analysis demonstrated that MA and ILS [9] constitute the most promising approaches.

Subsequently, the potential of ILS was explored further. Two general frameworks, based on the ILS and the Great Deluge Algorithm respectively, were proposed. The main novelty relies on exploring the neighborhood in a more efficient way [17,18] with the aim of increasing the amount of iterations that might be evolved.

An alternative to reduce the computational cost of local search-based approaches was proposed by Ceberio, Mendi-buru, and Lozano [6]. They conducted a theoretical analysis of neighborhoods that allowed them to identify specific positions where the indexes might generate local optima solutions. This way, a restricted insert neighborhood is considered in conjunction with the best performing algorithms [4], MA and ILS [9], to generate the MA$r$ and ILS$r$ methods. The comparisons of MA$r$ and ILS$r$—at the level of solution quality but especially in terms of execution times—demonstrated the superiority of these approaches. In order to go further in their testing, the authors generated an extra set of more complex instances, known as xLOLIB2, that were also integrated in LOLIB. xLOLIB2 is currently the set of largest instances that is most discussed in the related literature.

In order to further improve the exploration behavior of ILS and MA, the Hybrid Exploration Algorithm (HEA) was devised, which is a population-based metaheuristics based on ILS [19]. Similarly to the proposal put forth in this paper, HEA moves the balance from exploration to exploitation during the execution. This is done through a perturbation with a dynamic strength. Note that differently to our proposal, this adaptation does not take into account the stopping criterion or elapsed period. Additionally, a post-optimizer based on exactly solving submatrices of the instance by dealing with the LOP as a binary linear programming problem is applied. New best-known solutions were set to 77 out of the 78 instances from the xLOLIB [9].

Qian et al. [20] studied the properties of block insertions and proposed three new local search schemes that iteratively improve the LOP solutions by moving blocks of nodes. Among such local search schemes, the first-strategy outperforms the others in terms of computational time, so it was embedded into some state-of-the-art metaheuristics, obtaining the so-called MA$b$ and ILS$b$. This proposal is especially relevant for short-term optimizers.

Finally, with the aim of exploiting some structural properties of the LOP, Santucci and Ceberio [21] designed a new approach, denoted by CD-RVNS, which involves the following components: a VNS algorithm based on restricted neighborhoods, a randomized heuristic construction procedure that builds-up an LOP solution precedence by precedence, and a destruction procedure that removes precedences from an LOP solution with the aim of producing a new starting point for the construction procedure. VNS considers two different neighborhoods, insert and interchange, by switching between them when the search is trapped in a local optimum. During the construction and destruction procedures, an LOP solution is represented as a set of pairwise precedences, thus differing from the classical linear representation of permutations, which is indeed used in the VNS phase. Dealing with both representations requires the implementation of conversion methods between representations. Although representing a linear ordering as a set of precedences requires more memory than encoding a linear permutation, such a representation allows constructive heuristics to smoothly build up the solutions, precedence by precedence, thus avoiding
drastic changes in the objective value.

In summary, it is important to note that most state-of-the-art proposals call attention to the importance of diversification with the aim of avoiding stagnation. However, recent design principles based on relating the diversity to the stopping criterion and the elapsed period have never been applied in LOP solvers. Additionally, most recent strategies have focused on employing specific features of the structure of the LOP and/or hybridizing metaheuristics with exact solvers to advance the results further, meaning that the complexity of solvers has increased drastically, with algorithms containing several thousand lines of code. Finally, trajectory-based approaches have generally been more successful in short-term executions, whereas in the long term, population-based approaches are usually more convenient. While MA reports most of the current best-known solutions, the trajectory-based ILS, and CD-RVNS solvers, currently offer some of the best-known solutions for several instances, so when validating new solvers, it is important to consider both kinds of metaheuristics.

Finally, there are some recent techniques that while have not yet been able to generate state-of-the-art results for the LOP, represent quite interesting research lines. In [22, a gradient search technique that operates on the space of probability distributions over LOP solutions was designed. This method scales properly, so it seems an interesting research line for very large instances. Algebraic Differential Evolution allowed the generation of new best-known solutions in a related permutation-based problem in [23]. Adapting this variant to the LOP might bring new research paths to this field.

3 Memetic algorithm with explicit diversity management: MA-EDM

Solving practical and complex problems through metaheuristics requires the design of robust and adaptive components. Over the years, several studies have proposed various techniques to address different kinds of drawbacks, e.g., premature convergence, oversampling of neutral networks, and stagnation in noisy functions. The use of hybrid methods that combine several of these techniques in proper ways has been particularly useful [24]. These hybrid methods have been successfully applied in a variety of fields such as multiobjective optimization, pattern recognition, vehicle routing problems and scheduling [25–27]. Moreover, hybrid methods have also been useful in dealing with the scalability of metaheuristics [28]. Among the hybrid methods, memetic algorithms are one of the most popular, and as discussed above, they have been successfully applied to the LOP. This work extends such algorithms to better deal with long-term executions.

Algorithm 1 Memetic Algorithm with Explicit Diversity Management.

Require: LOP instance, N (size of population), Crossover operator (CX or OB), Survivor selection strategy (BNP), Stopping criterion (time)  
1: Initialization: Generate an initial population \( P_0 \) with \( N \) individuals. Assign \( i = 0 \).  
2: Local Search: Perform a local search for every individual in \( P_0 \).  
3: Diversity Initialization: Calculate the initial desired minimum distance \( (D_0) \) as the mean distance among individuals in \( P_0 \).  
4: while the execution time has not finished do  
5: Mating Selection: Perform binary tournament selection on \( P_i \) in order to fill the mating pool with \( N \) parents.  
6: Variation: Apply the crossover in the mating pool to create the set \( O_i \) with \( N \) offspring.  
7: Local Search: Perform a local search for each individual in \( O_i \).  
8: Survivor Selection: Apply the survivor selection strategy (BNP) to create \( P_{i+1} \) by considering \( P_i \) and \( O_i \) as input.  
9: \( i = i + 1 \)  
10: end while  
11: Return best evaluated permutation.

Regarding the linear ordering problem, knowing near-optimal solutions for large instances is important for the analysis of the performance of LOP solvers. Given the hypothesis previously stated, this research is focused on developing an LOP solver that, in the long-term, is able to find new best-known values for LOP instances of high complexity. Diversity-aware strategies that gradually shift from exploration to exploitation as evolution progresses have excelled in combinatorial problems [8]. Thus, to attain our aim, a novel diversity-aware LOP solver, the memetic algorithm with explicit diversity management (MA-EDM) is proposed. MA-EDM was designed by considering some genetic operators that have already proven to be effective for the LOP. Algorithm 1 shows the pseudocode of our proposal. It is a first-generation MA, so a static improvement individual strategy is applied. Each component is described below in detail.

Our approach starts by initializing a population with \( N \) individuals randomly (Line 1). As stated previously, in most effective LOP solvers, the solutions directly encode a permutation, so this was the representation we selected. The initialization considers a uniform random generator, meaning that each permutation is equiprobable. Then, each solution is improved by applying a first-improvement stochastic hill-climber that considers the insert neighborhood (Line 2) [10]. Note that the neighborhood chosen for the local search plays a key role in the performance of these algorithms. Considering this, we select the insert neighborhood since it is one of the most successful neighborhood definitions [29]. Neighbors are generated by simply moving a number in the given permutation to an alternative position while shifting all the numbers in the intermediate positions.

Naive implementations of the applied local search evaluate the whole neighborhood in \( O(N^3) \). However, since the
best insertion position of a number can be calculated in $O(N)$, neighbors are visited more systematically to reduce the total cost to $O(N^2)$ [17]. Note that similar improvement strategies are applied in most state-of-the-art MAs [18,21]. It is also important to note that the stochastic behavior of the hill-climber is maintained because at each iteration, the order in which the numbers to move are considered is random. Once a number is selected, if its best insertion move improves the current solution, it is accepted. This is repeated until no neighbor improves the current solution. Note that the local search with the complexity reduction technique allows achieving local optima quite efficiently, even for the largest instances considered in our experimental validation. For example, in instances with $n = 1000$, the time to reach a local optimum, when starting from a randomly created solution, is less than one second. This means that it can be directly integrated into MAs without the need to consider alternative stopping criteria.

An important feature of our approach is that it considers diversity explicitly. This is done with the replacement strategy, which works by setting a minimum desired distance that is updated during the run. The details are given later, but similarly to [30], the distances that appear in the initial population are used to set the initial desired distance ($D_0$) (Line 3). Note that this has the advantage of adapting the execution automatically to the given instance [30], in the sense that usually, for the most complex instances where a higher diversity is required to reach high-quality solutions, the improved initial population tends to exhibit larger distances. $D_0$ is calculated as the mean distance among all the individuals in the initial population. In order to calculate $D_0$, a metric or a distance-like function must be selected. In this case, the Spearman’s footrule distance ($D_{sf}$) [31] is employed. The dissimilarity between two permutations $x$ and $y$ with $p$ elements is defined as follows:

$$D_{sf}(x, y) = \sum_{i=1}^{p} \| i - pos(x[i], y) \|$$

where $pos(v, perm[])$ is the index of the value $v$ in the permutation $perm$.

The Spearman’s footrule distance was selected because the position of a number establishes the amount of values that are summed from the associated column. Thus, summing the absolute distance between the positions for each number is meaningful. Note that this metric has a broader range than other typical metrics applied to permutation-based representations, such as the exact match distance, which provided important advantages for the Job Shop Scheduling problem [30]. Moreover, it can be calculated in $O(N)$, meaning that the impact on performance is low, which also justifies its selection. Any of the metrics described in [31] could also be used, and some of them might even yield better results. However, given the advantages of the Spearman’s footrule distance mentioned previously, and that this work considers a large amount of complex instances that require computationally expensive experiments, the analyses of alternative metrics is left for future work.

As in standard MAs, MA-EDM evolves a set of generations until a given stopping criterion is reached (Lines 4-10). To provide fair comparisons, and since very different types of proposals, including trajectory-based and population-based approaches, are considered in our validation, the stopping criterion was set according to time. At each generation, a set of $N$ parents is selected using binary tournaments (Line 5). Then, the variation phase is applied (Line 6), and the generated offspring are subjected to the same improvement strategy that was applied to the initial population (Line 7). In this case, the variation consists only of a crossover operator, and the mutation is omitted. The logic behind this decision is that in many cases, MAs obtain proper solutions even if mutation is not used [32]. The local search performs modifications that allow new alleles to appear, so the increased complexity of including an additional operator is not always justified [33]. This is especially true when applying diversity-aware strategies [30], because one of the other roles of mutation is to avoid the premature loss of diversity, but this is also one of the aims of the diversity management strategies. Finally, a diversity-aware replacement strategy is applied (Line 8). The replacement phase and the crossover operators applied in MA-EDM are detailed in the next subsections.

### 3.1 Best non-penalized survivor selection strategy

One of the main novelties of this work is the way to select survivors. This strategy is executed in the Line 8 of Algorithm 1. The main design principle behind BNP is similar to the one discussed in [8] and relies on introducing a dependency between the selection of survivors and the ratio between the elapsed execution period and the stopping criterion. Specifically, the aim is to promote a larger degree of exploration in the initial phases of the execution, and then to gradually shift towards exploitation as the end of the optimization process nears. Note that BNP was successfully applied in the optimization of the Job Shop Scheduling Problem [30]. However, to our knowledge, it has never been applied in the Linear Ordering Problem.

Algorithm 2 shows the pseudocode of BNP. It receives as its input the current population and offspring, and it outputs the population of the next generation. BNP is based on promoting both quality and diversity, and the second aim is attained by avoiding the selection of individuals that are too close to each other. The definition of too close is dynamic. Specifically, a threshold distance ($D$) that is used to identify too-close individuals is calculated with the formula given in Line 1. In this formula, $T_{current}$ is the number
Algorithm 2 Best Non-Penalized (BNP) Survivor Selection Strategy.

Require: Population \( P = \{p^1, ..., p^N\} \), Offspring \( O = \{o^1, ..., o^N\} \)
1: \( D = D_0 = \frac{T_{\text{total}}}{T_{\text{pop}}} \times D_0 \)
2: \( S_{\text{Eligible}} = P \cup O \)
3: \( S_{\text{Penalized}} = \emptyset \)
4: \( S_{\text{New Population}} = \emptyset \)
5: while \(|S_{\text{New Population}}| < N\) do
6: for individual \( \in S_{\text{Eligible}} \) do
7: \( \text{individual.dci} = DCI(\text{individual}, S_{\text{New Population}}) \)
8: if \( \text{individual.dci} < D \) then
9: \( S_{\text{Eligible}} = S_{\text{Eligible}} \setminus (\text{individual}) \)
10: \( S_{\text{Penalized}} = S_{\text{Penalized}} \cup (\text{individual}) \)
11: end if
12: end for
13: if \( S_{\text{Eligible}} \neq \emptyset \) then
14: \( \text{new_selected} = \arg \max_{x \in S_{\text{Eligible}}} x.o_{\text{obj}} \) // Ties are broken randomly
15: else
16: for individual \( \in S_{\text{Penalized}} \) do
17: \( \text{individual.dci} = DCI(\text{individual}, S_{\text{New Population}}) \)
18: end for
19: \( \text{new_selected} = \arg \max_{x \in S_{\text{Penalized}}} x.d_{\text{ci}} \) // Ties are broken randomly
20: end if
21: \( S_{\text{New Population}} = S_{\text{New Population}} \cup \{\text{new_selected}\} \)
22: end while
23: return \( S_{\text{New Population}} \)

of elapsed seconds from the start of the optimization process, and \( T_{\text{total}} \) is the stopping criterion (also in seconds). Thus, the initial distance (\( D_0 \)) previously discussed is reduced linearly, so that at the end of the optimization process, \( D \) is set to zero. This means that, as the evolution progresses, the promotion of diversity is reduced and, consequently, more importance is given to the quality. This transformation is performed gradually, resulting in a steady shift from exploration to intensification. BNP manages three sets of individuals. The eligible set (\( S_{\text{Eligible}} \)) contains individuals that have not yet been selected to survive and that are not too close (distance lower than \( D \)) to any of the solutions already selected to survive. The penalized set (\( S_{\text{Penalized}} \)) contains individuals that have not yet been selected to survive and whose distance to a survivor is lower than \( D \). Finally, the new population set (\( S_{\text{New Population}} \)) contains the survivors. Initially, \( S_{\text{Eligible}} \) is filled with the union of the current population and offspring (Line 2), whereas \( S_{\text{Penalized}} \) and \( S_{\text{New Population}} \) are empty (Lines 3 and 4). Finally, the process iteratively selects survivors until \( S_{\text{New Population}} \) is filled with \( N \) individuals (Line 5 to 22). At each iteration, the distance condition is checked for each individual in \( S_{\text{Eligible}} \) (Lines 6 and 7), and those that are too close to an already selected survivor are moved to \( S_{\text{Penalized}} \) (Lines 8 to 11). Note that in the pseudocode, \( DCI \) stands for Distance to Closest Individual, and is defined as follows: \( DCI(x, P) = \min\{\text{distance}(x, y) \mid y \in P\} \), where \( \text{distance}(x, y) \) refers to the Spearman’s footrule distance already discussed. In those iterations where \( S_{\text{Eligible}} \) is not empty, i.e. there are sufficiently distant individuals, the best among them (the one that maximizes the objective function) is selected to survive (Lines 13-14 and 21). The opposite case represents situations where the diversity is not as large as desired. Thus, the penalized solution with the longest distance to the closest survivor is selected (lines 16-21).

To further illustrate the replacement operator, Fig. 1 shows the behavior of BNP in three different scenarios: initial, intermediate and final phases of the run. Particularly, in each case, it considers that its input is a set of eight solutions (union of previous population and offspring) from which four solutions must be picked up as survivors. Each subfigure represents the status at the end of the BNP process. Note that the threshold distance to perform the penalties is reduced over time, meaning that each of the previous scenarios consider different threshold distances. In order to facilitate the illustration, this figure considers that BNP is applied to a continuous problem with two variables \((x_1 \text{ and } x_2)\). Each small circle with a number represents a solution and the value inside the circle represents its relative quality, meaning the solution tagged with a 1 is the best one, whereas the one tagged with a 8 is the worst-one. Solutions with a red circle are the survivors (\( S_{\text{New Population}} \)), the ones with a gray circle are the penalized ones (\( S_{\text{Penalized}} \)), whereas the ones with a black circle are not penalized nor selected (\( S_{\text{Eligible}} \)). Also note that the blue circle surrounding each survivor represents the region of the search space that is penalized because of such a selection. Note that, at each iteration of BNP, the Best Non-Penalized solution is selected until four or them are marked in red color. Due to the different threshold sizes, the survivors marked in subfigure (a) are more focused on diversity than on quality, whereas subfigure (c) represents a case with a higher selection pressure. In this particular example, it selects the best solutions even if it implies an important reduction of diversity. One of the keys of BNP is precisely to perform a smooth reduction of the threshold distance, with the aim of shifting gradually from exploration to exploitation.

3.2 Crossover operators

To select the crossover operators, the literature was extensively reviewed. Two of the most effective operators for the LOP are the Cycle Crossover (CX) [34] and the Order Crossover (OB) [35]. A clear winner is not identified in the related literature [9], so both of them are considered in this research.

The CX operator takes two parents as input and is based on the principle of inheriting values in its exact positions. Thus, each number in the offspring permutation shares its position with one of the two parents. An example of an application of the CX operator is shown in Fig. 2a. The parents are denoted
as $\rho_1$ and $\rho_2$, whereas the offspring is denoted as $o_1$. First, the offspring inherits the elements that share the same value in both parents, simply by copying them in their original position. These positions are marked with a blue rectangle and are shown with a red circle in the child $o_1$. Then, the remaining elements are used to establish cycles that have the following property: consecutive elements in the cycles share the same position in the parents. Note that, with this property, the cycles can be identified unequivocally. In the example shown in Fig. 2a, there are two cycles. The first one contains the numbers 4, 2, 7, 3 and 9 because the pairs (4, 2), (2, 7), (7, 3), (3, 9) and (9, 4) share the same positions in the parents. The second one contains the numbers 1 and 5. Then, for each of these cycles, a parent is randomly selected and all numbers in the cycle are copied from this parent. In the example, the first cycle is inherited from $\rho_2$ and the second cycle is inherited from $\rho_1$. In $o_1$ the values marked with violet circles correspond to the first cycle, while the ones with orange circles correspond to the second cycle. Note that to generate a second offspring, the process is repeated.

In contrast, the OB operator is concerned with preserving the partial order among elements. An example of an application of OB is shown in Fig. 2b. As in the previous case, $\rho_1$ and $\rho_2$ are the parents, whereas $o_1$ is the offspring. This operator starts by randomly selecting $\lceil n/2 \rceil$ positions of the vectors. In the illustration of OB, these positions are marked with a blue rectangle. The offspring inherits the values from $\rho_1$ in the unmarked positions. In the illustration, these values are 6, 2, 7 and 1. These numbers are shown with an orange circle to note that they are inherited from $\rho_1$. Subsequently, the numbers that have not been copied to $o_1$ are detected and sorted according to the order of appearing in $\rho_2$. Then, they are copied in the marked positions in this order. In the illustration, these numbers are 9, 4, 8, 3 and 5. They are shown with a violet circle to note that they are inherited by considering the order of $\rho_2$. Finally, note that to generate the second offspring, the roles of $\rho_1$ and $\rho_2$ are interchanged.

### 4 Experimental study

This section is devoted to validate the advances attained by MA-EDM. One of the main aims of this research is to obtain new best-known solutions for the most complex sets of instances, so that they can be used as new reference solutions. However, it is also important to show that such solutions are not attainable by state-of-the-art solvers when similar computational resources are granted. Thus, the set of experiments aims to provide new best-known solutions, compare MA-EDM against state-of-the-art solvers and shed some light on the reasons behind the superior performance of MA-EDM.

In the LOP field, several sets of instances with different dimensions, densities and degrees of complexity have been proposed and integrated in LOLIB. These sets of instances are widely used in the literature and are the standard for upgrading the state of the art in the LOP field [36]. The experimental validation considers the benchmarks sets in LOLIB [4], which are the following: IO [13], MB [37], SGB [38], RandB [4], RandA1 [10], RandA2 [12], Spec [4], xLOLIB [9] and xLOLIB2 [6]. Some of them, especially the four first sets of benchmarks, are relatively easy and are almost systematically solved by current solvers. Thus, they are useful just to check the robustness of new proposals. The rest of the benchmark sets contain some larger and more difficult cases, and even though some of them were proposed more than 30 years ago, they have yet not been solved to optimality. Thus, they are normally used to check new advances in the field.

Table 1 summarizes some of the most relevant features of the sets of instances included in LOLIB. In particular, it shows the number of instances that comprise each set, the sizes of the instances, the mean density of each package,
Table 1  Summary information of the sets of instances that comprise LOLIB

| Set     | Instances | Size(#Instances) | Density(%) | Known Optima |
|---------|-----------|------------------|------------|--------------|
| IO [13] | 50        | 44(31)           | 39.89      | 50           |
|         |           | 50(4)            |            |              |
|         |           | 56(11)           |            |              |
|         |           | 60(3)            |            |              |
|         |           | 79(1)            |            |              |
| MB [37] | 30        | 100(5)           | 48.96      | 30           |
|         |           | 150(10)          |            |              |
|         |           | 200(10)          |            |              |
|         |           | 250(5)           |            |              |
| SGB [38]| 25        | 75(25)           | 46.27      | 25           |
| RandB [4]| 90       | 40(20)           | 49.61      | 71           |
|         |           | 44(50)           |            |              |
|         |           | 50(20)           |            |              |
| RandA1 [10]| 100   | 100(25)          | 50.05      | -            |
|         |           | 150(25)          |            |              |
|         |           | 200(25)          |            |              |
|         |           | 500(25)          |            |              |
| RandA2 [12]| 75    | 100(25)          | 46.79      | 25           |
|         |           | 150(25)          |            |              |
|         |           | 200(25)          |            |              |
| Spec [4]| 37        | 11-36(8)         | 36.54      | 30           |
|         |           | 50-452(29)       |            |              |
| xLOLIB [9]| 78      | 150(39)          | 41.63      | -            |
|         |           | 250(39)          |            |              |
| xLOLIB2 [6]| 200    | 300(50)          | 39.13      | -            |
|         |           | 500(50)          |            |              |
|         |           | 750(50)          |            |              |
|         |           | 1000(50)         |            |              |

i.e., the percentage of nonzero entries in the matrices, and the number of instances in which the current best-known solution matches the current best-known upper bound. Note that the size of the instances is usually considered one of the most important features in determining the complexity of the instance. However, other features, such as the density, are also important. Note that in LOLIB, there is not a large difference in the densities of the different packages. Thus, xLOLIB2, which contains the largest instances, is usually considered the most difficult benchmark of LOLIB. For further discussion of the complexity of LOP instances, readers are referred to [39]. With respect to the number of known optima in each set, it is notable that only in the smallest instances is there a match between the best-known solutions and the best-known upper bounds. As shown below, the best-known solutions are improved in this paper for 293 instances. However, the best-known upper bounds are not reached in any of those instances. In fact, the distances between the upper bounds and the best-known solutions are quite large, so there is probably also room for improvement in terms of the upper bounds.

Regarding state-of-the-art solvers, the validation was conducted by considering both trajectory- and population-based metaheuristics. For the trajectory-based methods, the ILS, [6] and CD-RVNS [21] methods were used. Note that CD-RVNS yields better results than ILS, in most cases. However, there are some specific instances where ILS provides a highly effective search, so it was maintained in our validation. Regarding the population-based alternatives, MA, [9] is the current clear winner, and similarly to MA-EDM, it is also a Memetic Algorithm, so it was selected to perform the validation. Note that the source codes of all these methods are publicly available. In order to facilitate the replicability of our experiments, our source code is also freely available.¹

Four kinds of experiments are presented below. First, the tuning step of MA-EDM is described. Second, MA-EDM is compared against the selected state-of-the-art solvers. Third, some analyses that provide a better understanding of the differences between the dynamics of the populations of MA, and MA-EDM are detailed. Finally, an analysis to better understand the effect of the stopping criterion is included. Note also that in order to provide sound conclusions, and taking into account the stochastic behavior of the solvers involved, all our executions were repeated 30 times and proper statistical tests were applied. Specifically, the results

¹ https://github.com/carlossegurag/LOP_MA-EDM. In addition to the source code, the datasets generated and/or analyzed during the current study are available in this repository in the RawData directory. Moreover, this repository contains the best-known solutions generated for each instance in the BKS.zip file.
were compared with the *scmamp library* [40] and Scikit-learn [41], following some of the guidelines described in [42]. All the tests were applied assuming a significance level of 5%.

### 4.1 Parameterization

The parameterization of MA-EDM is a relatively simple task. The two only decisions that had to be made to perform the experimentation were the selection of the crossover operator and the population size. As mentioned earlier, regarding the crossover operator, it was clear that CX and OB had yielded the best overall results [9], so both of them were considered in the tuning phase. Regarding the population size, it is well known that adequate values usually depend on other components of the optimizer, such as the stopping criterion, in non-trivial ways [43,44]. Since diversity-aware techniques usually require long executions and they alter the dynamics of the population, using previously reported experiments to select the population size is not too meaningful, so we decided to test the following values: 25, 100, 200, 300, 400, 500 and 600.

In order to understand the implications of the parameterization on the results, four instances of the benchmark with the largest instances (xLOLIB2) were considered. Since xLOLIB2 consists of instances with four different sizes, one instance of each size was selected. They were *Nbe75eec300, Nbe75eec500, Nbe75eec750,* and *Nbe75eec1000.* The fourteen different configurations resulting from combining two crossover operators and seven population sizes were executed 30 times for each instance. The stopping criterion was set to 4 hours.

The results of each instance were analyzed independently. First, the Kruskal-Wallis test was applied as an omnibus test. In every case, significant differences appeared, so the Mann Whitney test with Hommel’s correction was used to identify the pairwise differences [45]. Additionally, to visualize the results, the Nemenyi test was applied. The critical difference plot of each instance is shown in Fig. 3. Note that in this kind of plot, the average rank position of each model is shown, and horizontal lines join schemes with no statistically significant differences. Each scheme is denoted by a string formed by the concatenation of the crossover operator and the population size. In particular, the ranking sorts the solvers in decreasing order, meaning that the solvers with lower values are preferred. As an example, in the *Nbe75eec300* instance, CX200 obtained a value close to 3.5, which means that when sorting the results for the different seeds of the 14 configurations tested, the mean ranking of CX200 was this value. When considering the global results, the advantage of the CX operator is quite clear. In the four instances, at least the six best configurations used the CX operator. Moreover, there was only one instance (*Nbe75eec500*) in which the results attained with the best configuration that relies on the OB operator were not significantly different from those reported by the best CX operator. Thus, the superiority of the configurations with the CX operator is considerable.

Regarding the population sizes, the conclusions are not as clear. First, an important asset of MA-EDM is that for configurations with the CX operator, most of the configurations with different population sizes appear linked, meaning that the results are not very sensitive to the population size. In fact, only the CX25 configuration is significantly different. Thus, there is quite a large range of admissible values. Since CX200 was the best ranked in two instances and its ranking difference was never larger than half of the critical difference, this configuration was selected to continue our validation. Note
Table 2 Parameterization applied in each state-of-the-art solver and MA-EDM

| Optimizer     | Parameterization          |
|---------------|---------------------------|
| ILSr          | $\epsilon = 0.0001, i_{\text{change}} = 7, n_{ni} = 750$ |
| MAr           | $N = 200$                 |
| CD-RVNS       | Parameter-less Algorithm  |
| MA-EDM        | $N = 200$, Crossover Operator=CX |

that selecting any of the other configurations in the 100 to 600 range would probably lead to similar conclusions.

Finally, note that while the Nemenyi critical difference plot is quite illustrative, the more advanced statistical tests Kruskal Wallis (as omnibus) and Mann Whitney test with the Hommel’s correction (as post-hoc) are more advisable [45]. When comparing CX200 to the remaining configurations with these tests, the same conclusions than the ones obtained with the Nemenyi test could be drawn.

### 4.2 Validation against the state of the art

One of the most important objectives of our research is to design a proper long-term solver, i.e., an optimizer that is able to profit from long-term executions by avoiding stagnation problems. In order to validate this aim, the three state-of-the-art optimizers selected and MA-EDM were executed with every instance of the nine benchmark sets by setting the stopping criterion to 4 hours. Table 2 shows the parameterization applied in each optimizer. Note that this is the parameterization recommended by the authors, but in the case of MAr, a parameterization study similar to the one presented above was carried out to set the population size, and the value $N = 200$ was selected. Each execution was repeated 30 times with different seeds.

In order to summarize our results, statistical tests similar to those from the previous experiment were applied. Specifically, the results attained for each instance with the four optimizers tested were compared by using the Kruskal Wallis test as an Omnibus test, followed by the Mann Whitney with the Hommel’s correction to detect pair-wise differences. Note that, for every instance, MA-EDM always attained an equal or larger mean and median than the rest of the methods. Table 3 shows, for each benchmark set, the number of instances where the differences between MA-EDM and each state-of-the-art method selected were statistically significant (column $\succ$) and not statistically significant (column $\succeq$). As we can note, the amount of instances where the results attained by MA-EDM are statistically superior is quite noticeable. Particularly, the superiority appeared in 439, 289 and 420 instances out of the 685 instances, when comparing MA-EDM to ILSr, MAr, and CD-RVNS, respectively. In the smallest benchmarks, not many cases with statistically significant differences appeared. However, as the complexity grows, the number of instances with statistically significant differences in favor of MA-EDM clearly increases. Particularly noticeable is the case of xLOLIB2, where significant differences appeared in the whole benchmark set.

Since xLOLIB2 contains the largest instances, eight instances were selected from this benchmark to show the results in more detail. They were two instances from each of the different sizes present in the benchmark set. Table 4 shows the floor of the mean objective function attained by each method, as well as the current Best-Known Solution (BKS). The improvement provided by MA-EDM is noticeable, and the differences with respect to the BKS are quite significant, which indicates that current solvers are not as effective with these large and complex instances, and it is precisely in the most complex cases where MA-EDM excels. Table 5 shows the amount of new BKS found by each method in the different benchmark sets. Note that when more than one method found the same new BKS, it is counted in all of them. The advantages provided by MA-EDM are quite clear. In fact, there was no instance where MA-EDM was exceeded in terms of the best or mean attained value.

In order to facilitate future validations, the supplementary material attached to this work contains the detailed results for the 685 instances considered over the course of this research. For each instance, the best and mean solution attained by each method is shown. In addition, the BKS is included. MA-EDM was able to set new bounds in 293 cases out of the 685 instances, and it was always able to attain the BKS, meaning that the increase in robustness is huge in comparison to current solvers.

### 4.3 Analysis of diversity

One of the most important distinguishing features of MA-EDM is the explicit management of the diversity present in the population. To understand the different dynamics of the populations of MA-EDM and MAr, four instances were selected and the intermediate results were saved. Figure 4 shows the trend in the population diversity during the run. Specifically, the diversity was calculated at different times as the mean of the Spearman’s footrule distances appearing in the population, and the results of 30 independent executions were averaged. The four instances tested showed the same trend. In MA-EDM, the reduction in diversity was almost linear, which is explained by the linear reduction in the desired distance that is considered in the replacement phase (see Line 1 in Algorithm 2). In contrast, MAr reduced its diversity quite quickly, maintaining a low but nonzero diversity during the whole run. Note that the amount of diversity oscillates in a range of low values. This is because MAr is a generational scheme that does not introduce any special mechanisms to promote diversity, meaning that there is a tendency to con-
Table 3: Summary of statistical tests: column \( > \) shows the number of instances where statistically significant differences appeared in favor of MA-EDM, and column \( = \) counts the cases with no significant differences.

|          | IO | MB | RandA1 | RandA2 | RandB | SGB | Spec | xLOLIB | xLOLIB2 |
|----------|----|----|--------|--------|-------|-----|------|--------|---------|
| MA-EDM vs ILS\(_r\) | 50 | 0  | 29     | 1      | 2     | 98  | 42   | 33     | 90      | 0       | 25     | 33    | 4     | 0     | 78    | 0     | 200 |
| MA-EDM vs MA\(_r\)  | 50 | 0  | 30     | 0      | 67    | 33  | 75   | 0      | 90      | 0       | 25     | 0     | 36    | 1     | 23    | 55    | 0     | 200 |
| MA-EDM vs CD-RVNS   | 50 | 0  | 30     | 0      | 13    | 87  | 51   | 24     | 90      | 0       | 25     | 31    | 6     | 0     | 78    | 0     | 200 |
| Total instances     | 50 | 0  | 100    | 0      | 75    | 90  | 75   | 90     | 25      | 37      | 37    | 38    | 78    | 200  |

Table 4: Floor of the mean objective function attained in eight selected instances from xLOLIB2.

| Instance/Algorithm | BKS     | ILS\(_r\) | MA\(_r\) | CD-RVNS | MA-EDM |
|--------------------|---------|-----------|----------|---------|--------|
| Ntiw56n54_300      | 2654641 | 2657882   | 2661516  | 2650252 | 2662901 |
| Nt75e11xx_300      | 145044286 | 145093367 | 145297161 | 144958876 | 145388485 |
| Nt65f11xx_500      | 29285209 | 29284570  | 29336499 | 29247869 | 29365313 |
| Nt75e11xx_500      | 374756997 | 374780909 | 375603945 | 374381639 | 375998123 |
| Nt65n58_750        | 20633749 | 20661491  | 20704523 | 20645181 | 20720453 |
| Nt65b11xx_750      | 129443488 | 129523720 | 129868029 | 129349693 | 130011100 |
| Nbe75ecc_1000      | 122183020 | 122395155 | 122743873 | 122255253 | 122852580 |
| Nt70f11xx_1000     | 177714930 | 177754220 | 178289425 | 177802295 | 178387875 |

Table 5: New BKS found in the experimentation process.

| Instance/Algorithm | ILS\(_r\) | MA\(_r\) | CD-RVNS | MA-EDM | # Instances improved |
|--------------------|----------|----------|---------|--------|---------------------|
| IO                 | 0        | 0        | 0       | 0      | 0                   |
| MB                 | 0        | 0        | 0       | 0      | 0                   |
| SGB                | 0        | 0        | 0       | 0      | 0                   |
| Spec               | 3        | 3        | 1       | 3      | 3                   |
| RandB              | 0        | 0        | 0       | 0      | 0                   |
| RandA1             | 2        | 10       | 4       | 38     | 38                  |
| RandA2             | 1        | 4        | 3       | 4      | 4                   |
| xLOLIB             | 0        | 16       | 5       | 48     | 48                  |
| xLOLIB2            | 0        | 0        | 0       | 200    | 200                 |
| Total of new BKS   | 6        | 33       | 13      | 293    | 293                 |

The mutation eventually reintroduces some amount of diversity, which explains the peaks. However, these peaks are low, meaning that MA\(_r\) performs the search by placing all the individuals in a relatively narrow region, and all the individuals are moved relatively close to one another. In comparison, MA-EDM initially keeps the individuals quite distant from one another, and as the execution progresses, they are gathered in the most promising regions located thus far.

To complement the above findings, the trend in the best solution evaluated during the search is shown in Fig. 5. As in the previous experiment, the mean of 30 executions is shown and, in this case, the trajectory-based methods are also included. First, note that for these long-term executions, population-based approaches are much more effective. Second, it is also clear that MA-EDM exhibits a slow convergence, and specifically, when compared to MA\(_r\), it shows a lower solution quality during most of the search. This is because during the initial phases, it promotes larger levels of exploration than MA\(_r\), and the intensification is delayed until the last phases. However, as the end of the search nears and the diversity decreases, intensification is promoted, and MA-EDM is able to profit from the promising regions identified and yield much better final quality values. Finally, it is also important to note that these results do not indicate that MA-EDM requires more than 3 hours to improve the results of MA\(_r\). If a more restrictive computational budget is applied, MA-EDM reduces its diversity more quickly because of the relation between the management of diversity and the stopping criterion set in the optimizer, so even when using shorter times, MA-EDM outperforms MA\(_r\). This is analyzed in the next section.
Fig. 4  Evolution of the diversity (mean Spearman’s footrule distance) during the run in four selected instances (mean of 30 executions). The gradual shift from exploration to exploitation in MA-EDM is clear.

(a) Nbe75ee300  
(b) Nbe75ee500  
(c) Nbe75ee750  
(d) Nbe75ee1000

Fig. 5  Evolution of the best objective function attained during the run in four selected instances (mean of 30 executions). As expected, the diversity-aware model excels in the last phase of the optimization process.

(a) Nbe75ee300  
(b) Nbe75ee500  
(c) Nbe75ee750  
(d) Nbe75ee1000

4.4 Effect of the stopping criterion

One of the distinguishing features of MA-EDM is that its internal behavior depends on the stopping criterion. In particular, for more restrictive computational budgets, MA-EDM reduces its diversity more quickly because of the relation between the management of diversity and the stopping criterion set in the optimizer. In the results presented in Fig. 5, MA-EDM outperformed MA_r near the end of the optimization process. MA-EDM delays the intensification phase, which explains this long time required by MA-EDM. However, shorter periods imply faster convergence, meaning that the time required to outperform the other optimizers might be shorter. In order to study this effect, similar executions were carried out with the same instances but using as stopping criterion several different values between 1 minute and 240
Fig. 6  Effect of the Stopping Criterion (mean of 30 executions) in four selected instances. MA-EDM outperforms the others methods with stopping criteria larger than 1 hour.

Fig. 6 shows, in the left part, the fitness attained by MA-EDM and the others optimizers for the different stopping criteria. In the smallest instances, MA-EDM requires a really short period (less than ten minutes) to outperform the results of $MA_r$. However, as the sizes of the instances increase, this time is also larger. In particular, in the largest instance, MA-EDM outperforms $MA_r$ when performing executions with a stopping criterion larger than 1 hour. Note that this is a much shorter value than the one presented in the previous subsection. Trajectory-based algorithms did not obtain high-quality results in any of the tested time periods, so they are not so promising for large instances, where population-based techniques excel.
The right part of Fig. 6 shows the trend in diversity for population-based approaches when considering the stopping criterion set to 60 minutes and 240 minutes. The behavior of MA-EDM is the expected one, i.e. the diversity is decreased linearly in both cases. Differently, the decrease in diversity in MA is independent of the stopping criterion, which explains the relative stagnation in its results, and the impressive advance achieved by MA-EDM.

5 Conclusions and future work

This paper represents an effort to further improve the best-known solutions for standard benchmarks used in the validation of LOP solvers. To attain this goal, a novel MA with explicit management of diversity is designed. This proposal is called the memetic algorithm with explicit diversity management (MA-EDM). The main novelty of MA-EDM is the incorporation of a survivor selection strategy that is based on avoiding individuals that are too close to each other, which is done dynamically and considers the Spearman’s footrule distance. The experimental validation clearly confirms the advantages provided by this design principle for the LOP in comparison to state-of-the-art solvers, which is the main contribution of this paper. New best-known solutions are attained in 293 well-known instances. Moreover, the advantages provided by MA-EDM are confirmed with proper statistical tests, and additional analyses are conducted to understand the reasons behind the important advances in MA-EDM in comparison to alternative MAs. The differences between the amount of diversity maintained in MA-EDM and in other MAs are noticeable, which explains the important improvements contributed by MA-EDM. Thus, this research confirms the importance of proper diversity management in LOP solvers, and establishes a significant amount of new best-known solutions that should be considered in future validations of LOP solvers.

We can identify some limitations of the study and proposal put forth in this paper, which provide some new opportunities for future work:

- One of the most important drawbacks of our approach is that high-quality solutions are only reached near the end of the optimization process. Introducing some alternative mechanisms that combine the intensification and exploration phases during the whole run might improve the anytime behavior of MA-EDM [46], which could allow its application in alternative conditions, where more is taken into account than the quality of the final solution.
- This research does not analyze alternative ways of calculating distances between solutions. Considering alternative metrics such as the exact match distance and the reversal distance seems quite promising.
- The densities of the instances considered in this paper are not very high or very low. It would be interesting to create additional instances with a variety of densities with the aim of analyzing the impact of density on performance.
- Since the neighborhood considered in this paper exhibits quadratic growth, dealing with matrices with several thousand rows and columns might drastically limit the number of generations evolved, which could considerably deteriorate the quality. Integrating some techniques to increase the scalability of our proposal and reduce the time required to attain high-quality solutions seems interesting.

Acknowledgements Authors acknowledge the support from “Laboratorio de Supercómputo del Bajío” through the project 300832 from CONACYT.

References

1. Leontief WW (1936) Quantitative input and output relations in the economic systems of the United States. Rev Econ Stat 18(3):105–125
2. Chenery HB, Watanabe T (1958) International comparisons of the structure of production. Econometrica 26(4):487–521
3. Garey MR, Johnson DS (1979) Computers and Intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., Philadelphia, USA
4. Martí R, Reinelt G, Duarte A (2012) A benchmark library and a comparison of heuristic methods for the linear ordering problem. Comput Optim Appl 51:1297–1317
5. Martí R, Reinelt G (2011) The linear ordering problem - exact and heuristic methods in combinatorial optimization, p. 169. Springer, Heidelberg Dordrecht London New York
6. Ceberio J, Mendiburu A, Lozano JA (2015) The linear ordering problem revisited. Eur J Oper Res 241(3):686–696
7. Črepinšek M, Liu S-H, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv 45(3):35–13533
8. Segura C, Hernández-Aguirre A, Luna F, Alba E (2016) Improving diversity in evolutionary algorithms: new best solutions for frequency assignment. IEEE Trans Evol Comput 21(4):539–553
9. Schiavinotto T, Stützle T (2004) The linear ordering problem: instances, search space analysis and algorithms. J Math Model Algor 3(4):367–402
10. Laguna M, Martí R, Campos V (1999) Intensification and diversification with elite tabu search solutions for the linear ordering problem. Comput Oper Res 26(12):1217–1230
11. Chanas S, Kobylański P (1996) A new heuristic algorithm solving the linear ordering problem. Comput Optim Appl 6:191–205
12. Campos V, Glover F, Laguna M, Martí R (2001) An experimental evaluation of a scatter search for the linear ordering problem. J Glob Optim 21:397–414
13. Grötschel M, Jünger M, Reinelt G (1984) A cutting plane algorithm for the linear ordering problem. Oper Res 32(6):1195–1220
14. García CG, Pérez-Brito D, Campos V, Martí R (2006) Variable neighborhood search for the linear ordering problem. Comput Oper Res 33(12):3549–3565
15. Charon I, Hudry O (2007) A survey on the linear ordering problem for weighted or unweighted tournaments. 4OR 5, 5–60
16. Charon I, Hudry O (2010) An updated survey on the linear ordering problem for weighted or unweighted tournaments. Ann Oper Res 175:107–158
17. Sakuraba CS, Yagiura M (2010) Efficient local search algorithms for the linear ordering problem. Intl Trans Op Res 17:711–737
18. Sanfect & Ceberio J, Lozano JA (2020) Journey to the center of the linear ordering problem. Expert Syst Appl 42(9):4432–4442
19. Garcia E, Ceberio J, Lozano JA (2019) Hybrid heuristics for the linear ordering problem. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1431–1438
20. Qian Y, Lin J, Li D, Hu H (2020) Block-insertion-based algorithms for the linear ordering problem. Comput Oper Res 115:104861
21. Santucci V, Ceberio J (2020) Using pairwise precedences for solving the linear ordering problem. Appl Soft Comput 87:105998
22. Santucci V, Ceberio J, Baiollo M (2020) Gradient search in the space of permutations: an application for the linear ordering problem, pp. 1704–1711. Association for Computing Machinery, New York, NY, USA
23. Baiollo M, Milani A, Santucci V (2020) Variable neighborhood algebraic differential evolution: an application to the linear ordering problem with cumulative costs. Inf Sci 507:37–52
24. Fernandes IF, Silva JrDM, Goldbarg EFG, Maia SM, Goldbarg MC (2020) A PSO-inspired architecture to hybridise multi-objective metaheuristics. Memetic Comp 12(3):235–249
25. Falcón-Cardona JG, Hernández Gómez R, Coello Coello CA, Castillo Tapia MG (2021) Parallel multi-objective evolutionary algorithms: a comprehensive survey. Swarm Evol Comput 67:100960
26. Sun L, Pan Q-K, Jing X-L, Huang J-P (2021) A light-robust-optimization model and an effective memetic algorithm for an open vehicle routing problem under uncertain travel times. Memetic Comp 13(2):149–167
27. Asadujjaman M, Rahman HF, Chakraborty RK, Ryan MJ (2021) A memetic algorithm for concurrent project scheduling, materials ordering and suppliers selection problem. Procedia Computer Science 192, 717–726. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021
28. Amaya JE, Cotta C, Fernández-Leiva AJ, García-Sánchez P (2020) Deep memetic models for combinatorial optimization problems: application to the tool switching problem. Memetic Comp 12(3):149–167
29. Sankhala S, Mendiburu A, Lozano JA (2020) Journey to the center of the linear ordering problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2020, pp. 201–209. Association for Computing Machinery, New York, NY, USA
30. Hernández Constantino O, Segura C (2021) A parallel memetic algorithm with explicit management of diversity for the job shop scheduling problem. Appl Intell, 1–13
31. Sevaux M, Sörensen K, et al. Permutation distance measures for memetic algorithms with population management. In: Proceedings of 6th Metaheuristics International Conference. MIC'05, pp. 832–838
32. Neri F, Cotta C, Moscato P (2011) Handbook of memetic algorithms. Springer, Berlin, Heidelberg
33. Tsai H-K, Yang J-M, Tsai Y-F, Kao C-Y (2004) An evolutionary algorithm for large traveling salesman problems. IEEE Trans Syst Man Cybern Syst 34(4):1718–1729
34. Mez P, Freisleben B (2000) Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Trans Evol Comput 4(4):337–352
35. Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold Company, New York, USA
36. Song J, Zhao H, Zhou T, Tao Y, Lü Z (2019) Solving the linear ordering problem via a memetic algorithm. In: Arai, K., Bhatia, R., Kapoor, S. (eds.) Proceedings of the future technologies conference (FTC) 2018, pp. 421–430. Springer, Cham
37. Mitchell JE, 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. Springer, Boston, MA, pp. 349–366
38. Knuth D (1993) The stanford graphbase: a platform for combinatorial computing. ACM Press, New York, USA
39. Pérez A, Ceberio J (2018) Creating difficult instances of the linear ordering problem. In: XIII Congreso Español en Metaheurísticas Y Algoritmos Evolutivos Y Bioinspirados, pp. 733–738
40. Calvo B, Santafé G (2016) scamp: statistical comparison of multiple algorithms in multiple problems. R Found 8(1):248–256
41. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M (2011) Édouard Duchesnay: Scikit-learn: machine Learning in Python. J Mach Learn Res 12(85):2825–2830
42. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Inf Sci 180(10):2044–2064
43. Eiben AE, Smith JE (2015) Introduction to evolutionary computing. Natural Computing Series. Springer, Berlin, Heidelberg
44. Zhang Y-A, Ma Q, Sakamoto M, Furutani H (2010) Effects of population size on the performance of genetic algorithms and the role of crossover. Artif life Robot 15(2):239–243
45. del Amo IG, Delta DA (2013) SRCS: a technique for comparing multiple algorithms under several factors in dynamic optimization problems. Metaheuristics for dynamic optimization. Springer, Berlin, Heidelberg, pp 61–77
46. Radulescu A, López-Ibáñez M, Stützle T (2013) Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In: Purshouse RC, Fleming PJ, Fonseca CM, Greco S, Shaw J (eds) Evolutionary multi-criterion optimization. Springer, Berlin, Heidelberg, pp 825–840

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.