Hybrid Linkage Learning for Permutation Optimization with Gene-pool Optimal Mixing Evolutionary Algorithms

Michal W. Przewozniczek
Dep. of Computational Intelligence
Wroclaw Univ. of Science and Techn.
Wroclaw, Poland
michal.przewozniczek@pwr.edu.pl

Marcin M. Komarnicki
Dep. of Computational Intelligence
Wroclaw Univ. of Science and Techn.
Wroclaw, Poland
marcin.komarnicki@pwr.edu.pl

Peter A.N. Bosman
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
peter.bosman@cwi.nl

Dirk Thierens
Utrecht University
Utrecht, The Netherlands
d.thierens@uu.nl

Bartosz Frej
Fac. of Pure and Applied Mathematics
Wroclaw Univ. of Science and Techn.
Wroclaw, Poland
bartosz.frej@pwr.edu.pl

Ngoc Hoang Luong
University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
hoangln@uit.edu.vn

ABSTRACT

Linkage learning techniques are employed to discover dependencies between problem variables. This knowledge can then be leveraged in an Evolutionary Algorithm (EA) to improve the optimization process. Of particular interest is the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) family, which has been shown to exploit linkage effectively. Recently, Empirical Linkage Learning (ELL) techniques were proposed for binary-encoded problems. While these techniques are computationally expensive, they have the benefit of never reporting spurious dependencies (false linkages), i.e., marking two independent variables as being dependent. However, previous research shows that despite this property, for some problems, it is more suitable to employ more commonly-used Statistical-based Linkage Learning (SLL) techniques. Therefore, we propose to use both ELL and SLL in the form of Hybrid Linkage Learning (HLL). We also propose (for the first time) a variant of ELL for permutation problems. Using a wide range of problems and different GOMEA variants, we find that also for permutation problems, in some cases, ELL is more advantageous to use while SLL is more advantageous in other cases. However, we also find that employing the proposed HLL leads to results that are better or equal than the results obtained with SLL for all the considered problems.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**;

KEYWORDS

Genetic Algorithm, Estimation-of-Distribution Algorithm, Linkage Learning, Model Building, Empirical linkage learning

ACM Reference Format:
Michal W. Przewozniczek, Marcin M. Komarnicki, Peter A.N. Bosman, Dirk Thierens, Bartosz Frej, and Ngoc Hoang Luong. 2021. Hybrid Linkage Learning for Permutation Optimization with Gene-pool Optimal Mixing Evolutionary Algorithms. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463152

1 INTRODUCTION

Many real-world problems that may be represented in a permutation-based manner are NP-hard [8, 12]. Genetic Algorithms (GAs) were shown to be effective in solving some of these problems and are frequently employed for this purpose [9, 16]. The model-based Evolutionary Algorithms (EAs) were shown to outperform the classic EAs that do not use problem decomposition techniques [3, 11, 13, 15, 22, 23]. However, the body of literature on model-based EAs for permutations is far smaller than for binary or real-valued spaces. Nevertheless, the recent advances reveal the significant potential brought by such optimizers, e.g., Generalized Mallows Estimation of Distribution Algorithm (GM-EDA) [5], Linkage Tree GOMEA (LT-GOMEA) for permutations [2], and the Parameter-less Population Pyramid for Permutation problems (P4) [37]. Therefore, in this paper, we take the next step in this promising direction.

LT-GOMEA and P4 employ SLL based on a Dependency Structure Matrix (DSM). Linkage learning based on a DSM is a part of various state-of-the-art GAs that were shown to be effective for binary [11, 15, 32] and non-binary [22] discrete optimization problems as well. Recently, Linkage Learning based on Local Optimization (3LO) was found to be a promising new approach to linkage learning [28]. 3LO does not work with a DSM and can be classified as an ELL technique rather than an SLL technique. A key advantage of 3LO is that it has been proven that it does not detect false linkages. However, this increase in linkage quality comes at a price: 3LO is computationally expensive. Consequently, based on results reported so far, SLL remains a better choice for some problems, especially those with many inter-gene dependencies [28].

Therefore, the main objectives of this paper are as follows. First, we propose a form of 3LO for permutation problems. Second, we hybridize this technique with an existing SLL technique for permutation problems to arrive at a novel HLL technique. Based on experiments using SLL and HLL within two state-of-the-art EAs, namely LT-GOMEA and P4, we show that using HLL is better or equal to using SLL for all considered problems. Similarly, we show that using HLL is also better or equal to using the proposed ELL for all considered problem-EA combinations with only one exception.

The remainder of this paper is organized as follows. In Section 2, we present related work. Sections 3 and 4 describe the proposed pbELL and linkage hybridization, respectively. The results of our experiments are presented in Section 5. Finally, the last section presents key conclusions and promising future work directions.

2 RELATED WORK

2.1 DSM-based linkage learning

In EAs, a DSM is a square matrix that represents a degree of dependence between variables (genes) [15]. Mutual information is frequently used to estimate a DSM from the evolving population [11, 28, 32, 35, 37, 38], which is defined as follows:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

where X and Y are random variables.

In the context of an EA, X and Y represent genes, and the probabilities $p(\cdot)$ are estimated using frequencies of (pairs of) gene values in the population. A DSM is often used as the basis of information upon which we identify higher-order dependency structures. For both methods considered in this paper, a clustering algorithm creates a Linkage Tree (LT) based on DSM. In an LT, nodes represent clusters of genes that are considered to be dependent on each other. Starting from the leaves that contain only one gene, higher nodes concatenate the clusters of genes represented by their two children-nodes. Finally, the root of an LT contains all genes. More information and examples considering the LT creation process may be found in [25, 28, 34]. Other means of computing a DSM from a population are possible as well. In particular, for permutation problems, a first proposal for doing so involved also a measure of adjacency of two genes in terms of the permutation being represented [2].

2.2 Linkage Learning based on Local Optimization

Linkage learning techniques employing statistical measures (such as presented in the previous subsection) to estimate dependencies between genes may be classified as statistical linkage learning (SLL) techniques. A significantly different approach, called Linkage Learning based on Local Optimization (3LO), was proposed in [28]. 3LO employs perturbations and a local optimization algorithm to empirically check if two genes are dependent. Therefore, it is classified as an ELL technique. Specifically, linkages are discovered on the basis of a single individual ($\boldsymbol{x} = [x_1, ..., x_n]$, where n is the problem size). For each gene m, a so-called *linkage scrap* is discovered. A linkage scrap is defined as follows.

$$\begin{aligned} &LScrap(\boldsymbol{x}, m) = \\ &\quad OR\Big(XOR\Big(\boldsymbol{x}, \boldsymbol{x}^{(m)}\Big), XOR\Big(opt(\boldsymbol{x}), opt\Big(\boldsymbol{x}^{(m)}\Big)\Big)\Big) \end{aligned} \tag{2}$$

where $\boldsymbol{x}^{(m)}$ is an individual $\boldsymbol{x}$ in which the mth gene is perturbed and $opt(\boldsymbol{x})$ is the individual $\boldsymbol{x}$ that results from optimization using First Improvement Hill Climber (FIHC) [11, 28]. The order of the genes as considered within FIHC optimization is chosen randomly but is the same for each *linkage scrap* discovery operation. In other words, the results of applying local optimization to individuals $\boldsymbol{x}$ and $\boldsymbol{x}^{(m)}$ are compared. Since FIHC with a given order works deterministically, information is obtained on how perturbation of the mth gene influences the optimization result. All genes of $opt(\boldsymbol{x})$ and $opt\Big(\boldsymbol{x}^{(m)}\Big)$ that differ together with the mth gene create a *linkage scrap* and are considered to be dependent.

A linkage learning procedure results in *perfect linkage* if it suffers neither from *false linkage* reporting nor *missing linkage* reporting. False linkage concerns reporting two genes that are independent to be dependent, whereas missing linkage concerns failing to report that two genes are dependent when in fact they are dependent. Contrary to most SLL methods, the procedure in 3LO is proven to never report *false linkage*. However, 3LO may still suffer from missing linkage [28].

In 3LO, *linkage scraps* are used to create a DSM-like matrix. Then, for each pair of genes, the number of times it is reported in a *linkage scrap* collection is counted and stored in the DSM. This DSM is then used to create an LT in the same way as in SLL. Since 3LO is computationally expensive, it is highly inefficient to use it in the same way that SLL techniques are used in EAs since they frequently update their linkage model (typically every generation). Therefore, a specific 3LO Algorithm (3LOa) dedicated to using 3LO was proposed in [28].

2.3 Linkage Diversity and Conditional Linkage

The authors of [35] stress that the SLL-based GA they have proposed successfully solves the Hierarchical-If-And-Only-If (HIFF) problem because it can perfectly recognize the underlying problem structure. However, as pointed out in [28, 33], for overlapping problems in general, a single model that exhibits *perfect linkage* described in groups of genes may not be enough to solve the problem efficiently. This is mainly due to the representation of linkages in terms of multiple groups (or clusters) of genes, either some linkages must be broken in order to be still able to obtain smaller linkage sets or very large linkage groups are required. Hence, the correct linkages are respectively not effectively or efficiently processed. In either case, this results in poor scalability of the EA in terms of finding the optimum as the problem length increases. Therefore, to go beyond the current state-of-the-art in linkage learning, new avenues should be explored. Along one avenue, recently, the concept of conditional linkages was proposed [4]. The conditional linkage allows representing overlapping dependencies, which was shown to be beneficial in case of overlapping problems with real-valued variables. However, the definition of the linkage model in that work was still pre-determined and not learned online, which is what is needed for successful black-box optimization, which is what we focus on here. The novel avenue in linkage learning that we consider in this paper is using more than one linkage model at the same time. With such a mixture of linkage models, any misrepresentations in one model may be alleviated through the use of another linkage model.

2.4 Modern evolutionary permutation-based optimization

For optimization problems in discrete Cartesian spaces, a candidate solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ can be directly represented using n discrete variables, i.e., $\boldsymbol{x} \in \times_{i=1}^{n} \mathbb{D}_i$, where $\mathbb{D}_i$ is the domain of variable i. For optimization problems in permutation spaces, a solution $\boldsymbol{x}$ is a permutation of $(1, 2, \ldots, n)$. A key issue using such a representation directly is that crossover operations almost always result in a solution that is not a permutation. One general-purpose solution to this issue is to make use of the random keys encoding. Permutations are then encoded by an n-dimensional vector $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ of n real-valued random keys, i.e., $\boldsymbol{r} \in \times_{i=1}^{n}[0, 1]$. The encoded permutation is found by sorting $\boldsymbol{r}$ in ascending order, such that $r_{x_1} < r_{x_2} < \ldots < r_{x_n}$. For example, $\boldsymbol{r} = (0.05, 0.62, 0.92, 0.80, 0.24)$ encodes $\boldsymbol{x} = (1, 5, 2, 4, 3)$. Any crossover in random-keys space always results in a valid encoding of a permutation.

GOMEA is a modern family of EAs which, at the core, revolve around the use of the Optimal Mixing (OM) variation operator. OM transforms an existing individual $\boldsymbol{r}$ in an iterative genetic-local-search-like manner. In the case of an LT model, OM goes through every node in the LT in random order. For each node, the variables identified in the node are copied from a donor individual $\boldsymbol{d}$ to $\boldsymbol{r}$. For instance, when individual $\boldsymbol{r} = (0.05, 0.62, 0.92, 0.80, 0.24)$ is recombined with donor $\boldsymbol{d} = (0.91, 0.14, 0.33, 0.45, 0.60)$ using the LT node $(1, 3, 4)$, it will be changed into $\boldsymbol{r} = (0.91, 0.62, 0.33, 0.45, 0.24)$. If the partially-altered individual is not worse than its previous state, the change is kept; otherwise, the change is undone. After all the nodes in the LT are traversed, the original individual has been transformed into a new individual that has an equal or better fitness value. Most commonly, the donor individual is selected (randomly) anew for each node in the LT. In that case, we speak of Gene-pool Optimal Mixing (GOM).

P4 [37] is a recently proposed version of the Parameter-less Population Pyramid (P3) [11] that was adjusted to solving permutation-based problems. Like P3, P4 employs a pyramid-like structured population rather than a classical vector (or set) structured population. A new individual is added in every iteration. Initially, the new (randomly generated) individual is added to the first level of the pyramid. Then, GOM is used to transform the solution, using the solutions from the same pyramid level as potential donors. If the individual is improved through GOM, it is added to the next level of the pyramid. This repeats until the solution is no longer improved or the top of the pyramid is reached. P3 and P4 share many mechanisms with LT-GOMEA [2]. Specifically, the same linkage learning techniques are used. Moreover, the use of GOM makes them part of the GOMEA family. Specifically for P4, further similarities with the permutation version of LT-GOMEA are random keys encodings and the incorporation of the random rescaling and re-encoding operators. The difference is that the parameterless version of LT-GOMEA scales its populations using an interleaved multi-start population-growing scheme [14], while P4 maintains multiple populations in the form of pyramid levels. In both cases, a separate linkage model is learned for each population, albeit that in P4 a new linkage model is learned every time a single new solution is added. As a consequence, P4 generates many more linkage trees during a single run than LT-GOMEA, which results in a more diverse set of linkage groupings, thus increasing the exploration capacity of the Optimal Mixing operator. Both GOMEA variants, were shown competitive to other state-of-the-art methods designed to solve permutation-based problems [2, 37].

3 EMPIRICAL LINKAGE LEARNING FOR PERMUTATION-BASED PROBLEMS

In this section, we propose a new linkage learning technique, inspired by 3LO [28]: Empirical Linkage Learning for Permutation-based Problems (pbELL). The motivation and intuition behind the pbELL proposition are as follows. First, pbELL (same as 3LO) should not propose any *false linkage* because, intuitively, a proper linkage model that is free of *false linkage* may be the key to effectiveness and efficiency. This intuition is confirmed by results obtained for 3LO on binary-encoded problems [28], by research on gray-box optimization [4, 36] - showing that the effectiveness of evolutionary methods can be increased significantly when knowledge concerning the problem structure decomposition is known a priori - and recent research concerning the relationship between linkage quality (i.e., the quality of problem decomposition supported by linkage) and the effectiveness of state-of-the-art evolutionary methods [26]. The second motivation behind proposing pbELL was to make its computational cost as low as possible. 3LO is so computationally expensive that it is not feasible to apply it in conjunction with the current state-of-the-art EAs that employ OM (e.g., LT-GOMEA [2], P3 [11], and DSMGA-II [15]). It was shown recently, for real-valued variables, that it is possible to achieve much more efficient ELL through fitness-based probing, leading to high-quality linkage models and much better results from GOMEA than when SLL methods are used [19]. Therefore, we wish to propose an ELL technique for permutation-based problems that can find high-quality linkage at a computational cost that is low enough to make pbELL applicable to state-of-the-art GAs like LT-GOMEA and P4.

Trying to capture the precise meaning of *dependence* or *linkage*, we propose the following approach. In the forthcoming description, we identify each gene with its initial number and use the language of permutations (rather than the language of random keys encoding), assuming that the domain of a fitness function fit is a set of permutations (rather than some Cartesian product). By a permutation of a finite set, we mean a bijective self-map of this set. Assuming that each individual is characterized by n genes, we will be mainly concerned with permutations of $\{1, ..., n\}$. Let $B = \{b_1, ..., b_r\}$ and $C = \{c_1, ..., c_s\}$ be a partition of $\{1, ..., n\}$, i.e., B and C are disjoint and $B \cup C = \{1, ..., n\}$. We will interpret B as a block of interesting genes and C as a context. We say that permutations π and ρ have the same order of B if $\rho^{-1}\pi$ is increasing on $\pi^{-1}(B)$ (or, equivalently, $\pi^{-1}\rho$ is increasing on $\rho^{-1}(B)$). For instance, permutations $\binom{1\ 2\ 3\ 4\ 5\ 6\ 7}{3\ 5\ 6\ 4\ 7\ 2\ 1}$ and $\binom{1\ 2\ 3\ 4\ 5\ 6\ 7}{1\ 2\ 3\ 7\ 4\ 5\ 6}$ have the same order of $\{3, 5, 6\}$.

Definition 1. *Genes from the block B are* independent *of the context C iff $fit(\pi) = fit(\rho)$ for any two permutations π and ρ which have the same order of B and of C. Because of the symmetry of the roles of B and C, we will also say simply that B and C are independent.*

Roughly speaking, this means that changes introduced exclusively in B are likely to change fitness, but transposing adjacent elements of B and C has no effect on fitness.

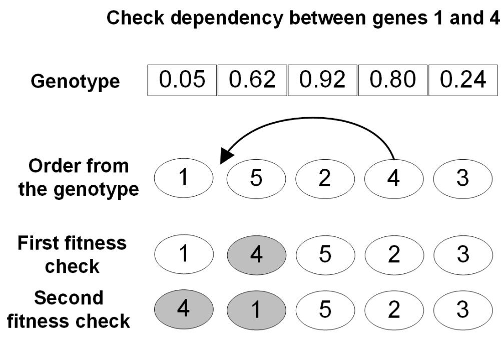


Figure 1: pbELL - linkage discovery for a single pair of genes

In Figure 1, we present the pbELL procedure of checking if two genes are dependent. The example considers a 5-gene permutation problem. First, the order resulting from the random keys values is obtained. Same as in 3LO, linkage is discovered on the basis of a single individual. To check if genes 1 and 4 are dependent, we modify the order resulting from the individual's genotype to make these genes adjacent, and we make the first fitness check. Then, we swap the order of the considered genes, and we do the second fitness check. If the fitness value of two fitness checks differs, then we find the two considered genes dependent. Otherwise, we do not report the dependency. The following theorem is a precise formulation that pbELL will never report a *false linkage.*

Theorem 1. *If B is a block independent of the context C then no two genes $i \in B$ and $j \in C$ will be reported dependent by pbELL.*

Proof. In the first step pbELL changes the order of genes to obtain a permutation π with adjacent i and j, i.e.,

$$\{\pi^{-1}(i), \pi^{-1}(j)\} = \{m, m+1\}.$$

Let τ_{ij} be the transposition of i and j, i.e., $\tau_{ij}(i) = j$, $\tau_{ij}(j) = i$, $\tau_{ij}(k) = k$ for k different than i and j. pbELL performs fitness check for π and $\tau_{ij}\pi$. We have

$$\pi^{-1}\tau_{ij}\pi(\pi^{-1}(i)) = \pi^{-1}(j)$$
$$\pi^{-1}\tau_{ij}\pi(\pi^{-1}(j)) = \pi^{-1}(i)$$

and

$$\pi^{-1}\tau_{ij}\pi(k) = k \text{ if } \pi(k) \notin \{i, j\}.$$

In other words, $\pi^{-1}\tau_{ij}\pi$ is the identity permutation with two adjacent elements flipped. The flipped elements belong to distinct sets $\pi^{-1}(B)$ and $\pi^{-1}(C)$, so $\pi^{-1}\tau_{ij}\pi$ is increasing both on $\pi^{-1}(B)$ and $\pi^{-1}(C)$. Hence, π and $\tau_{ij}\pi$ have the same order of B and C implying that $fit(\pi) = fit(\tau_{ij}\pi)$ and, consequently, dependence is not reported. □

pbELL will not report *false linkage* for any permutation-based problem type: relative, absolute, and neighbor. However, it is important to note that, like 3LO, pbELL may *miss* some linkage (i.e., pbELL may fail to detect some true gene dependencies) because the lack of independence may be revealed in permutations different from the one used by pbELL.

Table 1: Example of a pbELL-based DSM for five genes.

Gene number	1	2	3	4	5
1	x	**1**	**1**	0	0
2	**1**	x	**1**	0	0
3	**1**	**1**	x	**1**	0
4	0	0	**1**	x	**1**
5	0	0	0	**1**	x

The pbELL procedure described above is executed for all available pairs of genes. Since this operation is expensive, it is performed only during the initialization phase, and it does not change during the run of the EA.

The proposed pbELL may be found similar to Differential Grouping (DG) [20, 21, 30], which was also the inspiration for the recently proposed ELL method for the real-valued GOMEA [19]. DG is a problem decomposition technique dedicated to continuous search spaces. The core idea of DG is to perturb the genotype and analyze fitness changes triggered by the perturbation. Although pbELL behaves similarly, there are also significant differences. First, pbELL changes the original genotype to make the considered genes adjacent. Second, pbELL considers a specific perturbation designed to assure that the perturbation will only influence two selected genes. On the other hand, when DG perturbs the genotype, it may influence many genes. Moreover, its recent version, namely RDG [30], perturbs the whole groups of genes at once.

The number of fitness function evaluations (FFE) spent by pbELL is $n(n-1)$ (for each of the $\binom{n}{2}$ available gene pairs, we perform two fitness checks). Since pbELL is executed only once before the optimizer run, its computational cost seems reasonable.

The proposed pbELL returns information about whether two genes were found dependent or not. Therefore, we can represent this information in a DSM-like structure. An example of such a DSM is shown in Table 1. Note that genes 1, 2, and 3 are all dependent on each other. The situation is different for genes 3, 4, and 5. Although the pairs of genes (3,4) and (4,5) are dependent, no dependency was found between genes 3 and 5. Such linkage does not have to be incorrect. If we assume that the underlying problem structure is built from two blocks of dependent genes: (3,4) and (4,5), genes 3 and 5 are not directly dependent. Their dependency is determined only by gene 4. If the value of gene 4 is constant, then there is no dependency between genes 3 and 5.

The above example shows how pbELL-DSM may be used to discover the overlapping dependencies. The DSM entries take only values 0 or 1. To avoid any bias when using the DSM to construct higher-order linkage structures such as the LT, when pbELL is used, a small random value is added to each DSM entry.

4 HYBRIDIZED LINKAGE LEARNING

In this paper, we propose to hybridize our pbELL-DSM with an SLL based-approach as first published in [2]. We do so by adding the two DSM matrices. The entries of the SLL-DSM are all normalized to the range [0;1). Thus, in the proposed hybridization, the signal coming from the pbELL-DSM is dominating since it can only take the extreme values 0 and 1. The intuition behind the proposed hybridization approach is as follows. pbELL is useful in discovering if a particular pair of genes is dependent or not. However, it does not measure the strength of the dependency. This information is supplemented by predictive-DSM. When we hybridize pbELL-linkage with predictive-linkage, we do not add a random value to pbELL-DSM entries.

5 THE RESULTS

5.1 Test Problems

We consider three different problems. The first is the Permutation Flowshop Scheduling Problem (PFSP) [31] (also considered in the papers proposing LT-GOMEA [2] and P4 [37]). Each PFSP instance is defined by J jobs and M machines. Jobs are divided into operations that are processed on machines. Each machine can process only one operation type and only one operation at the same time. The main goal is to find a job-processing sequence π that minimizes the Total Flow Time (TFT) measure defined as follows:

$$TFT(\pi) = \sum_{i=1}^{J} c_{\pi(i),M} \tag{3}$$

where $c_{\pi(i),j}$ indicates the completion time of the ith job's operation on the jth machine. In this paper, we consider PFSP test cases using $\{100, 200, 500\}$ jobs and $\{5, 10, 20\}$ machines. The same test cases were considered in [2, 5, 37].

Another considered test problem is Linear Ordering Problem (LOP) [7]. LOP is NP-hard [6, 18] and is known for its many real-world applications [12]. The definition of LOP may be formulated using its graph representation [29]. Each LOP instance of size n can be represented by a directed graph $G = (V, E)$, where V ($|V| = n$) denotes a set of vertices and E indicates a set of edges. For each edge $(u, v) \in E$, a cost $c_{u,v}$ is given. When an edge between two vertices does not exist, then its corresponding cost is equal to 0. A sample problem solution is a permutation π of vertices, and its cost is defined by the following formula.

$$cost(\pi) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{\pi(i),\pi(j)} \tag{4}$$

In this paper, we consider the randomly generated instances proposed in [29][1]. We chose test cases of size 500 and all available densities d, i.e., $\{1\%, 5\%, 10\%, 50\%, 100\%\}$ where the density is the probability that there is an edge between any vertices pair.

The third considered problem is the ordering deceptive problem [17]. Ordering deceptive problems are built from m separable deceptive subproblems. The solution quality is the sum of subproblems values. Thus, contrary to PFSP and LOP, this problem is additively separable. Like binary deceptive functions [10], each subproblem has two optima (global and local), which are maximally distant. The subproblem was designed to attract optimizers to the local optima. We consider ordering deceptive problems consisting of $m \in \{10, 20, 40, 80, 160, 320\}$ subproblems of size four. The single subproblem values are dependent on relative ordering and are reported in [17].

5.2 Experiments setup

We consider two state-of-the-art model-based EAs dedicated to optimizing permutation-based problems, both from the GOMEA family – LT-GOMEA [2] and P4 [37]. Both methods are considered in four different versions:

- **Standard** that employs SLL proposed in [2]. Both LT-GOMEA and P4 were originally proposed in this form.
- **Empirical** that employs pbELL proposed in this paper.
- **Hybrid** that employs HLL.
- **Random**. In this version, the DSM entries are generated randomly at every method iteration. This version is employed to show the difference that is made by using linkage learning.

The source codes of LT-GOMEA[2] and P4[3] were obtained from the repositories of their authors. All the source codes were joined in one program and, if possible, share all the appropriate source code parts. The full source codes with the settings files and the detailed results of all the runs may be downloaded from GIThub[4]. Each experiment was executed 20 times. The statistical significance of the reported results was verified by the unpaired Wilcoxon test, we employ a significance level of 5%.

For PFSP, we have employed the same FFE-based stop condition as in [2, 5, 37]. For the LOP test cases, the situation was as follows. For the test cases of the length below 500 genes, all three considered LT-GOMEA and P4 versions that employed linkage learning (Standard, Empirical, and Hybrid) were finding results of the same quality. Therefore, we do not report these results in the paper. However, for 500-gene test cases, the number of FFE was not a reliable measure to terminate the computation – for these test cases, LT-GOMEA and P4 computation time was not linearly dependent on FFE. Some LT-GOMEA runs with the same FFE-based stop condition were finishing within few hours, but the other runs required days to finish. The reason for this situation was as follows. Depending on the run, LT-GOMEA may add very large populations sooner or later (it is dependent on the moment smaller populations are found useless and deleted). Computing DSM-based linkage for very large population sizes consumes no FFE, but it may be very time-consuming. Thus, the computational cost of FFE computation becomes similar or lower than the cost of other method activities. In such a situation, FFE is not a reliable computation load measure, which makes the FFE-based stop condition not reliable as well. More details about the non-linear relation between FFE and computation time, together with the fairness of the FFE-based stop condition, may be found in [24, 27]. Therefore, for the considered 500-gene LOP test cases, we have used a time-based stop condition set to 36 hours. All LOP-related experiments were executed on a DELL PowerEdge R7425 2xAMD Epyc 7601 256GB server. To make the comparison fair, the number of computation processes was equal to the number of available physical processor cores, no other resource-consuming processes were running. A similar experiment setup was employed in [28]. In Table 2, we report the median FFE spent on the whole experiment for each d (two experiments per each d). If the $GomeaFFE/P4FFE$ ratio is below 1, it indicates that for these test cases, LT-GOMEA-Hybrid quickly increased the population size of the largest population it maintained (i.e., the smaller populations are quickly found useless and deleted) and spent a significant amount of computation time on updating linkage. P4-Hybrid has

[1]http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/lop

[2]https://homepages.cwi.nl/~bosman/source_code.php

[3]https://github.com/przewooz/P4

[4]https://github.com/przewooz/hybbridLinkagePermut.git

Table 2: The comparison of median FFE average for LOP

d	P4-Hybrid		LT-GOMEA-Hybr.		Ratio	
1	1.42E+08	1.04E+08	2.25E+08	1.51E+08	0.63	0.69
5	1.11E+08	1.05E+08	1.71E+08	1.36E+08	0.65	0.77
10	1.14E+08	***1.09E+08***	1.49E+08	***9.16E+07***	0.77	***1.20***
50	1.13E+08	1.41E+08	2.00E+08	9.92E+07	0.56	1.43
100	1.10E+08	1.48E+08	1.31E+08	9.76E+07	0.84	1.51

outperformed LT-GOMEA-Hybrid for all considered LOP test cases, despite the lower or higher FFE per experiment (see Table 8 in Section 5.4). For one experiment (marked in bold), the results were not statistically significant. For the ordering deceptive problems, same as for LOP, we have employed the time-based stop condition. The computation time was 8 hours, which was enough to converge for all considered EAs.

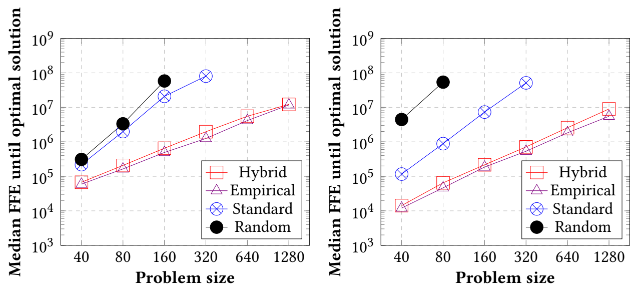
5.3 The Influence of Linkage Learning

In this section, we compare the effectiveness of all the considered LT-GOMEA and P4 versions that employ linkage learning (Standard, Empirical and Hybrid) with their versions that randomly generate linkage. The results for PFSP are reported in Table 3. For both considered methods, the *Standard*, and *Hybrid* versions outperform the version employing random linkage for all considered test cases.

The results of the comparison between *Random* and *Empirical* versions may seem surprising – the version that employs random linkage performs significantly better. However, taking into consideration the issue of linkage diversity [28] and the usefulness of conditional linkages [4] (see Section 2.3), these results seem intuitive. Due to the computational cost of pbELL, *Empirical* versions of LT-GOMEA and P4 employ the predetermined linkage model (linkage is discovered only once, before the optimization). Thus, in these versions, linkage remains the same for the whole run. PFSP is an overlapping problem (all genes are dependent on each other, although the strength of the dependency may differ). For such problems, using a single LT during a whole optimization process is unfavorable [25] and may prevent an EA from reaching high-quality results. P4 is less affected by the lack of linkage diversity than LT-GOMEA because it generates and maintains a more diverse population, which may partially alleviate the lack of linkage diversity.

For the considered LOP test cases (Table 4), the results obtained for P4 show that P4-Hybrid and P4-Standard outperform P4-Random for all test cases with $d < 100\%$. Similarly, P4-Empirical outperforms P4-Random for $d = 1\%$ and is less effective for $d = 100\%$. These results seem to be consistent with the results obtained for P4 for PFSP test cases. However, they also indicate that larger d decreases the advantage of linkage learning. This observation is confirmed by the results obtained for LT-GOMEA. For $d \geq 5\%$, LT-GOMEA-Random outperforms *Standard* and *Hybrid* versions. This situation is surprising because of the following reasons.

- If LT-GOMEA-Empirical performs better than LT-GOMEA-Random, then LT-GOMEA-Hybrid that is also using pbELL and pbELL is a dominating linkage (see Section 4) should outperform LT-GOMEA-Random as well.
- The results for P4 and LT-GOMEA differ significantly.
- The influence of linkage learning seems to depend on parameter d rather than on the problem type itself



(a) LT-GOMEA (b) P4

Figure 2: LT-GOMEA and P4 scalability for ordering deceptive problems (results with ≥ 50% optimal solutions found)

Explaining the above observations will be the objective of future research. Nevertheless, in Section 5.5, we propose an analysis that gives some insights into the reasons behind these surprising results.

In Figure 2, we present the scalability of all the considered LT-GOMEA and P4 versions for ordering deceptive problem. For this problem, all linkage learning versions outperform *Random* versions significantly. Such results are expected because to solve these problems successfully, it is necessary precisely to discover the subgroups of dependent genes [17]. The larger the problem size is, the less likely it is to point at least some of the existing gene groups by randomly generated linkage. Thus, linkage learning methods are significantly more effective in solving such problems.

5.4 Main Results

In Table 5, we present the comparison between all considered linkage learning versions for both considered methods for PFSP. For both methods, the *Standard* version outperforms the *Empirical* one significantly, which is expected in spite of the comparison with *Random* versions. LT-GOMEA-Hybrid and LT-GOMEA-Standard are equal for most of the considered test cases. However, for six test cases, the *Hybrid* version outperforms *Standard*. For P4, both versions seem equal.

For LOP, for both methods, the *Hybrid* version outperforms *Standard* for four out of ten test cases. The *Standard* version was found better than *Hybrid* only for one test case for LT-GOMEA. Thus, we may state that for LOP *Hybrid* version outperforms *Standard* for both considered methods. For P4, *Hybrid* and *Standard* versions outperform the *Empirical*. For LT-GOMEA, the *Empirical* version outperforms the *Standard* and *Hybrid* ones. These observations are consistent with the results presented in Table 6, although the results for LT-GOMEA remain surprising due to the reasons pointed out in the previous subsection.

Finally, for the ordering deceptive problems, as shown in Figure 2, *Hybrid* and *Empirical* versions significantly outperform all other for both methods. To successfully solve the ordering deceptive problems, it is necessary to discover gene dependencies precisely. The ordering deceptive problems are equivalent to problems built from deceptive function concatenations in binary search spaces. As shown in [26], SLL is fast and precise in decomposing such problems for binary search spaces. However, the results presented in this paper show that SLL (employed in *Standard* versions) does not have this advantage concerning the permutation-based problems. Therefore, for both methods, *Standard* versions scale significantly

Table 3: The comparison for the PFSP test cases between those versions of the considered methods that employ linkage learning and the versions with randomly generated linkage based on the *p*-values reported by the Wilcoxon Test

	LT-GOMEA			P4		
Test case group	Stand. vs Rand. Stand./eq./Rand.	Hybr. vs Rand. Hybr./eq./Rand.	Emp. vs Rand. Emp./eq./Rand.	Stand. vs Rand. Stand./eq./Rand.	Hybr. vs Rand. Hybr./eq./Rand.	Emp. vs Rand. Emp./eq./Rand.
100 jobs	**30** / 0 / 0	**30** / 0 / 0	0 / 0 / **30**	**30** / 0 / 0	**30** / 0 / 0	1 / **28** / 1
200 jobs	**20** / 0 / 0	**20** / 0 / 0	0 / 8 / **12**	**20** / 0 / 0	**20** / 0 / 0	0 / **18** / 2
500 jobs	**10** / 0 / 0	**10** / 0 / 0	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0	0 / **9** / 1

Table 4: The comparison for the LOP test cases between those versions of the considered methods that employ linkage learning and the versions with randomly generated linkage based on the *p*-values reported by the Wilcoxon Test

	LT-GOMEA Random vs...			P4 Random vs...		
Exp. no. (*d*)	Stand. dec.	Hybr. dec.	Emp. dec.	Stand. dec.	Hybr. dec.	Emp. dec.
1 (1)	*equal*	***Hybr.***	***Emp.***	***Stand.***	***Hybr.***	**Emp.**
2 (1)	***Stand.***	***Hybr.***	***Emp.***	***Stand.***	***Hybr.***	***Emp.***
3 (5)	*equal*	*equal*	*equal*	***Stand.***	***Hybr.***	*equal*
4 (5)	***Rand.***	*equal*	*equal*	***Stand.***	***Hybr.***	***Emp.***
5 (10)	***Rand.***	***Rand.***	*equal*	***Stand.***	***Hybr.***	*equal*
6 (10)	***Rand.***	*equal*	*equal*	***Stand.***	***Hybr.***	***Emp.***
7 (50)	***Rand.***	***Rand.***	*equal*	***Stand.***	***Hybr.***	*equal*
8 (50)	***Rand.***	*equal*	***Emp.***	***Stand.***	***Hybr.***	*equal*
9 (100)	***Rand.***	***Rand.***	***Emp.***	*equal*	*equal*	***Rand.***
10 (100)	***Rand.***	***Rand.***	***Emp.***	***Stand.***	***Hybr.***	*equal*

better than *Random* but are significantly outperformed by *Hybrid* and *Empirical.* The performance of both versions that employ pbELL is the same because pbELL is the key to precise linkage discovery for ordering deceptive problems.

In Table 7, we summarize the comparison between the different linkage learning versions. For both methods, the *Hybrid* version is better or equal to Standard for all considered problem types. In some cases (e.g., for ordering deceptive problems), Hybrid is as effective as the best of the two linkage learning techniques it hybridizes. However, for some method-problem combinations (e.g., for P4 applied to LOP or for LT-GOMEA applied to PFSP), the effectiveness of the *Hybrid* version is higher than *Standard* and *Empirical.* This shows that hybridization of various linkage learning techniques may lead to joining their pros rather than cons. Thus, this idea seems to be a highly promising research direction.

In Table 8, we present the direct comparison between the two most effective methods considered in this paper, namely LT-GOMEA-Hybrid and P4-Hybrid. For PFSP and LOP, we report the number of test cases for which a particular method reported statistically better results than the results of the other method. For ordering deceptive functions, we report the largest problem size, for which a method has found the optimal solution in at least 50% of the runs. For PFSP, LT-GOMEA significantly outperforms P4. The situation is the opposite for LOP. Such results are expected and confirm that P4-Hybrid is more suitable in solving LOP, while LT-GOMEA-Hybrid is more suitable in solving PFSP. The results for ordering deceptive problems show that no matter which method is used, they scale similarly (thanks to the employed linkage).

Due to paper size limitations, we the compare LT-GOMEA-Hybrid and P4-Hybrid with the competing methods only on the base of PFSP. We use the Average Relative Percentage Deviation (ARPD) [1, 5, 37]. As the competing methods we employ GM-EDA [5] and Random Key-based EDA (RK-EDA) [1]. The comparison given in Table 9 shows that both methods considered here are highly competitive to other state-of-the-art optimizers. Additionally, all results reported by the Hybrid versions of LT-GOMEA and P4 are of higher quality than the best solutions reported in [29] for LOP.

5.5 Results Discussion

The results reported in this paper show that the use of empirical linkage learning techniques is a promising research direction for permutation-based problems. We have also shown that linkage hybridization may significantly improve the results of two different GOMEA variants. The most significant difference between LT-GOMEA and P4 is in the way that they organize their adaptive population size and, as a consequence, in the number of diverse linkage trees they generate during the search. LT-GOMEA employs a classic population model, while P4 maintains a population that resembles a pyramid, and its size continuously increases during the method run.

The results obtained for LT-GOMEA solving LOP were surprising – for $d \geq 5\%$, the *Random* version outperformed *Standard* and *Hybrid*, although it was less effective than *Empirical.* The full explanation of this phenomenon requires further investigation. However, it seems that the nature of the considered test cases changes with the increase of *d.* The nature of the solved test case may depend on the test case rather than on the solved problem type. For instance, the LOP instances with $d = 100\%$ may be more similar in nature to PFSP instances with 100 jobs than LOP instances with $d = 5\%$. This issue requires further investigation and illustrates the importance of being able to apply landscape analysis techniques to guide the choice of the most suitable linkage learning technique for a specific problem instance.

6 CONCLUSION AND FUTURE WORK

In this paper, we have proposed pbELL, a new empirical linkage learning technique for permutation-based problems. pbELL was added to two state-of-the-art GOMEA optimization methods, namely LT-GOMEA and P4. For some of the considered problems, GOMEAs using pbELL were more effective than their original versions employing SLL. Finally, based on pbELL, we have proposed an HLL technique that joins linkage information obtained by pbELL and SLL. For all considered test problems and both GOMEA variants, the results obtained by the Hybrid version were better than or equal to the results obtained by the Standard one. Thus, we may state that this paper proposes two new GOMEA extensions,

Table 5: The influence of considered linkage learning techniques on the effectiveness of LT-GOMEA and P4 on the base of PFSP and the p-values reported by the Wilcoxon Test

	LT-GOMEA			P4		
Test case group	**Hybr. vs Stand. Hybr./eq./Stand.**	**Hybr. vs Emp. Hybr./eq./Emp.**	**Stand. vs Emp. Stand./eq./Emp.**	**Hybr. vs Stand. Hybr./eq./Stand.**	**Hybr. vs Emp. Hybr./eq./Emp.**	**Stand. vs Emp. Stand./eq./Emp.**
100 x 05	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0
100 x 10	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0
100 x 20	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0	1 / 9 / 0	**10** / 0 / 0	**10** / 0 / 0
200 x 10	3 / 7 /0	**10** / 0 / 0	**10** / 0 / 0	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0
200 x 20	0 / **10** / 0	**10** / 0 / 0	**10** / 0 / 0	0 / 9 / 1	**10** / 0 / 0	**10** / 0 / 0
500 x 20	3 / 7 / 0	**10** / 0 / 0	**10** / 0 / 0	1 / 8 / 1	**10** / 0 / 0	**10** / 0 / 0

Table 6: The comparison for the LOP test cases between those versions of the considered methods that employ linkage learning and the versions with randomly generated linkage based on the p-values reported by the Wilcoxon Test

		LT-GOMEA						P4					
Exp.		**Hybr. vs Stand.**		**Hybr. vs Emp.**		**Stand. vs Emp.**		**Hybr. vs Stand.**		**Hybr. vs Emp.**		**Stand. vs Emp.**	
no.	***d***	**equal**	**decision**	**equal**	**decision**	**equal**	**decision**	**equal**	**decision**	**equal**	**decision**	**equal**	**decision**
1	**1**	0.0001	***Hybrid***	0.0438	***Empirical***	0.0001	***Empirical***	0.0015	***Hybrid***	0.0001	***Hybrid***	0.0001	***Standard***
2	**1**	0.0034	***Hybrid***	0.0003	***Empirical***	0.0001	***Empirical***	0.0522	***Hybrid***	0.0124	***Hybrid***	0.5461	*equal*
3	**5**	0.0401	***Hybrid***	0.9702	*equal*	0.0674	*equal*	0.1615	*equal*	0.0001	***Hybrid***	0.0001	***Standard***
4	**5**	0.9405	*equal*	0.0028	***Empirical***	0.0025	***Empirical***	0.8813	*equal*	0.0002	***Hybrid***	0.0001	***Standard***
5	**10**	0.7938	*equal*	0.0036	***Empirical***	0.0169	***Empirical***	0.3507	*equal*	0.0002	***Hybrid***	0.0001	***Standard***
6	**10**	0.0859	*equal*	0.9702	*equal*	0.1354	*equal*	0.9553	*equal*	0.0003	***Hybrid***	0.0006	***Standard***
7	**50**	0.0137	***Standard***	0.0001	***Empirical***	0.0028	***Empirical***	0.0006	***Hybrid***	0.0001	***Hybrid***	0.0004	***Standard***
8	**50**	0.0479	***Hybrid***	0.0001	***Empirical***	0.0001	***Empirical***	0.0032	***Hybrid***	0.0001	***Hybrid***	0.0004	***Standard***
9	**100**	0.1672	*equal*	0.0001	***Empirical***	0.0001	***Empirical***	0.7369	*equal*	0.0001	***Hybrid***	0.0003	***Standard***
10	**100**	0.1790	*equal*	0.0001	***Empirical***	0.0002	***Empirical***	0.5755	*equal*	0.0009	***Hybrid***	0.0001	***Standard***

Table 7: The comparison of considered linkage learning techniques

	LT-GOMEA			P4		
	PFSP	**LOP**	**Dec.**	**PFSP**	**LOP**	**Dec.**
Hybrid	**Best**	Med.	**Best**	**Best**	**Best**	**Best**
Stand.	Med.	*Worst*	*Worst*	**Best**	Med.	*Worst*
Empir.	*Worst*	**Best**	**Best**	*Worst*	*Worst*	**Best**

Table 8: LT-GOMEA-Hybrid and P4-Hybrid comparison

	LT-GOMEA-Hybrid	**equal**	**P4-Hybrid**
PFSP	**23**	37	0
LOP	0	1	**9**
Deceptive	1280	N/A	1280

Table 9: The ARPD-based comparison between Hybrid versions of LT-GOMEA and P4 with other methods (the results of RK-EDA and GM-EDA are taken from [1] and [5], respectively)

Testcase	**LT-GOMEA-H**	**P4-H**	**RK-EDA**	**GM-EDA**
tai20-20-0	**0.00**	**0.00**	0.22	0.65
tai20-20-1	**0.00**	**0.00**	0.28	0.29
tai50-20-0	0.47	**0.42**	1.81	1.76
tai50-20-1	**0.25**	0.34	1.11	1.58
tai100-20-0	**0.96**	0.97	1.96	2.03
tai100-20-1	**0.80**	0.85	1.82	1.80
tai200-20-0	1.24	1.30	**1.07**	1.59
tai200-20-1	**1.12**	1.35	1.32	1.45
tai500-20-0	1.07	1.38	**0.29**	8.90
tai500-20-1	1.21	1.34	**0.68**	8.58

namely LT-GOMEA-Hybrid and P4-Hybrid, that are more effective in solving the permutation-based problems than their original SLL versions.

The effectiveness of the *Hybrid* versions is based on the hybridized linkage discovery technique that seems to be more successful in finding better linkage models for a wider range of problems than SLL or ELL alone.

The most important directions of future research are as follows.

- Proposing other hybrid linkage learning techniques for other than permutation-based search spaces.
- Investigation why LT-GOMEA with linkage learning is less effective than LT-GOMEA-Random for LOP with $d \geq 5\%$.
- Proposing linkage quality measures for permutation-based problems and investigating the linkage quality influence on the effectiveness of evolutionary methods designed to solving permutation-based problems.

Finally, one of the conclusions that arise from the results presented in this paper is that two test cases of different problems may be more similar in their nature, than the two test cases of the same problem. This observation shows the importance of having fitness landscape techniques that can assist in choosing or adjusting the linkage learning technique applied.

ACKNOWLEDGMENTS

We thank Jerzy Sas for his help in performing the experiments.

REFERENCES

[1] M. Ayodele, J. McCall, O. Regnier-Coudert, and L. Bowie. 2017. A Random Key based Estimation of Distribution Algorithm for the Permutation Flowshop

Scheduling Problem. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2364–2371. https://doi.org/10.1109/CEC.2017.7969591

[2] Peter A.N. Bosman, Ngoc Hoang Luong, and Dirk Thierens. 2016. Expanding from Discrete Cartesian to Permutation Gene-pool Optimal Mixing Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, 637–644.

[3] Anton Bouter, Tanja Alderliesten, Cees Witteveen, and Peter A. N. Bosman. 2017. Exploiting Linkage Information in Real-Valued Optimization with the Real-Valued Gene-Pool Optimal Mixing Evolutionary Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Berlin, Germany) *(GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 705–712. https://doi.org/10.1145/3071178.3071272

[4] Anton Bouter, Stefanus C. Maree, Tanja Alderliesten, and Peter A. N. Bosman. 2020. Leveraging Conditional Linkage Models in Gray-Box Optimization with the Real-Valued Gene-Pool Optimal Mixing Evolutionary Algorithm. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 603–611. https://doi.org/10.1145/3377930.3390225

[5] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano. 2014. A Distance-Based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem. *IEEE Transactions on Evolutionary Computation* 18, 2 (2014), 286–300.

[6] Stefan Chanas and Przemysław Kobylanski. 1996. A New Heuristic Algorithm Solving the Linear Ordering Problem. *Comput. Optim. Appl.* 6, 2 (Sept. 1996), 191–205. https://doi.org/10.1007/BF00249646

[7] Hollis B. Chenery and Tsunehiko Watanabe. 1958. International Comparisons of the Structure of Production. *Econometrica* 26, 4 (1958), 487–521.

[8] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian. 1996. Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research* 3, 1 (1996), 1–21. https://doi.org/10.1016/0969-6016(96)00004-4

[9] Paulo Roberto de Oliveira da Costa, Stefano Mauceri, Paula Carroll, and Fabiano Pallonetto. 2018. A genetic algorithm for a green vehicle routing problem. *Electronic notes in discrete mathematics* 64 (2018), 65–74.

[10] Kalyanmoy Deb and David E. Goldberg. 1993. Sufficient Conditions for Deceptive and Easy Binary Functions. *Ann. Math. Artif. Intell.* 10, 4 (1993), 385–408.

[11] Brian W. Goldman and William F. Punch. 2014. Parameter-less Population Pyramid. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (Vancouver, BC, Canada) *(GECCO '14)*. ACM, 785–792.

[12] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. 1984. A Cutting Plane Algorithm for the Linear Ordering Problem. *Operations Research* 32 (12 1984), 1195–1220.

[13] N. Hansen, S. D. Müller, and P. Koumoutsakos. 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation* 11, 1 (2003), 1–18. https://doi.org/10.1162/106365603321828970

[14] Georges R. Harik and Fernando G. Lobo. 1999. A Parameter-less Genetic Algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1* (Orlando, Florida) *(GECCO'99)*. 258–265.

[15] Shih-Huan Hsu and Tian-Li Yu. 2015. Optimization by Pairwise Linkage Detection, Incremental Linkage Set, and Restricted / Back Mixing: DSMGA-II. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*. ACM, 519–526.

[16] Sašo Karakatič and Vili Podgorelec. 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing* 27 (2015), 519–532. https://doi.org/10.1016/j.asoc.2014.11.005

[17] Hillol Kargupta, Kalyanmoy Deb, and David E. Goldberg. 1992. Ordering genetic algorithms and deception. In *Parallel Problem Solving from Nature – PPSN II*. Springer, 47–56.

[18] Richard M. Karp. 1972. *Reducibility among Combinatorial Problems*. Springer US, Boston, MA, 85–103.

[19] Chantal Olieman, Anton Bouter, and Peter A. N. Bosman. 2020. Fitness-based Linkage Learning in the Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm. *IEEE Transactions on Evolutionary Computation* (2020). https://doi.org/10.1109/TEVC.2020.3039698

[20] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. 2014. Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation* 18, 3 (2014), 378–393.

[21] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao. 2017. DG2: A Faster and More Accurate Differential Grouping for Large-Scale Black-Box Optimization. *IEEE Transactions on Evolutionary Computation* 21, 6 (Dec 2017), 929–942.

[22] Kalia Orphanou, Dirk Thierens, and Peter A. N. Bosman. 2018. Learning Bayesian Network Structures with GOMEA. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 1007–1014. https://doi.org/10.1145/3205455.3205502

[23] Martin Pelikan and David E. Goldberg. 2001. Escaping Hierarchical Traps with Competent Genetic Algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation* (San Francisco, California) *(GECCO'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 511–518.

[24] M. W. Przewozniczek. 2017. Problem Encoding Allowing Cheap Fitness Computation of Mutated Individuals. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. 308–316.

[25] Michal Witold Przewozniczek, Piotr Dziurzanski, Shuai Zhao, and Leandro Soares Indrusiak. 2021. Multi-Objective parameter-less population pyramid for solving industrial process planning problems. *Swarm and Evolutionary Computation* 60 (2021), 100773. https://doi.org/10.1016/j.swevo.2020.100773

[26] Michal W. Przewozniczek, Bartosz Frej, and Marcin M. Komarnicki. 2020. On Measuring and Improving the Quality of Linkage Learning in Modern Evolutionary Algorithms Applied to Solve Partially Additively Separable Problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 742–750.

[27] Michal W. Przewozniczek and Marcin M. Komarnicki. 2018. The Influence of Fitness Caching on Modern Evolutionary Methods and Fair Computation Load Measurement. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '18)*. ACM, 241–242.

[28] Michal W. Przewozniczek and Marcin M. Komarnicki. 2020. Empirical Linkage Learning. *IEEE Transactions on Evolutionary Computation* 24, 6 (Dec 2020), 1097–1111.

[29] Celso S. Sakuraba and Mutsunori Yagiura. 2010. Efficient local search algorithms for the linear ordering problem. *International Transactions in Operational Research* 17, 6 (2010), 711–737. https://doi.org/10.1111/j.1475-3995.2010.00778.x

[30] Y. Sun, M. Kirley, and S. K. Halgamuge. 2018. A Recursive Decomposition Method for Large Scale Continuous Optimization. *IEEE Transactions on Evolutionary Computation* 22, 5 (2018), 647–661. https://doi.org/10.1109/TEVC.2017.2778089

[31] E. Taillard. 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research* 64, 2 (1993), 278 – 285. Project Management anf Scheduling.

[32] Dirk Thierens. 2010. The Linkage Tree Genetic Algorithm. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I*. 264–273.

[33] Dirk Thierens and Peter Bosman. 2012. Predetermined versus Learned Linkage Models. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 289–296.

[34] Dirk Thierens and Peter A.N. Bosman. 2011. Optimal Mixing Evolutionary Algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin, Ireland) *(GECCO '11)*. ACM, New York, NY, USA, 617–624.

[35] Dirk Thierens and Peter A.N. Bosman. 2013. Hierarchical Problem Solving with the Linkage Tree Genetic Algorithm. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, 877–884.

[36] L. Darrell Whitley, Francisco Chicano, and Brian W. Goldman. 2016. Gray Box Optimization for Mk Landscapes Nk Landscapes and Max-Ksat. *Evol. Comput.* 24, 3 (Sept. 2016), 491–519. https://doi.org/10.1162/EVCO_a_00184

[37] Szymon Wozniak, Michal W. Przewozniczek, and Marcin M. Komarnicki. 2020. Parameter-Less Population Pyramid for Permutation-Based Problems. In *Parallel Problem Solving from Nature – PPSN XVI*, Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (Eds.). Springer International Publishing, Cham, 418–430.

[38] Adam M. Zielinski, Marcin M. Komarnicki, and Michal W. Przewozniczek. 2019. Parameter-less Population Pyramid with Automatic Feedback. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. ACM, New York, NY, USA, 312–313. https://doi.org/10.1145/3319619.3322052