Direct linkage discovery with empirical linkage learning

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

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

ABSTRACT

Problem decomposition is an important part of many state-of-the-art Evolutionary Algorithms (EAs). The quality of the decomposition may be decisive for the EA effectiveness and efficiency. Therefore, in this paper, we focus on the recent proposition of Linkage Learning based on Local Optimization (3LO). 3LO is an empirical linkage learning (ELL) technique and is proven never to report the *false linkage*. *False linkage* is one of the possible linkage defects and occurs when linkage marks two independent genes as a dependent. Although thanks to the problem decomposition quality, the use of 3LO may lead to excellent results, its main disadvantage is its high computational cost. This disadvantage makes 3LO not applicable to state-of-the-art EAs that originally employed Statistical-based Linkage Learning (SLL) and frequently update the linkage information. Therefore, we propose the Direct Linkage Empirical Discovery technique (DLED) that preserves 3LO advantages, reduces its costs, and we prove that it is precise in recognizing the *direct* linkage. The concept of direct linkage, which we identify in this paper, is related to the quality of the decomposition of overlapping problems. The results show that incorporating DLED in three significantly different state-of-the-art EAs may lead to promising results.

CCS CONCEPTS

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

KEYWORDS

Genetic Algorithms, Linkage Learning, Model Building, Empirical linkage learning

ACM Reference Format:
Michal W. Przewozniczek, Marcin M. Komarnicki, and Bartosz Frej. 2021. Direct linkage discovery with empirical linkage learning. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459333

1 INTRODUCTION

NP-hard nature is a feature of many real-world problems [9, 10, 29]. Thus, optimizers that can find high-quality solutions to such problems are in need. EAs, including Genetic Algorithms (GAs), were shown effective in solving various real-world problems of NP-hard nature and are frequently employed for this purpose [1, 2, 12]. However, with the increase of problem size, the effectiveness of the classical EAs usually decreases significantly [3, 7]. The employment of problem-modeling techniques is one of the promising ways to overcome this issue. Indeed, model-based EAs have been shown to outperform the classical EAs in solving practical [24, 28] and theoretical problems [15, 23]. The recent research concerning the influence of problem decomposition on the EAs' effectiveness shows that the quality of the employed model may be decisive for obtaining high-quality results.

The recently proposed 3LO is a new proposition of ELL technique. It is proven that it will never report the false linkage. However, its computational cost makes it inapplicable to EAs that originally employ SLL techniques and update their linkage model frequently (usually at every iteration). Therefore, the objectives of this paper are as follows. We propose DLED, a new ELL technique that is less expensive than 3LO, i.e. requires a lower number of Fitness Functions Evaluations (FFE). We introduce DLED into three different state-of-the-art EAs. In this gropup, we consider two SLL-based optimizers, namely, the Linkage Tree Gene-pool Optimal Mixing Evolutionary Algorithm (LT-GOMEA) [4] and Parameter-less Population Pyramid (P3) [14]. The third considered EA is the 3LO Algorithm (3LOa) that was designed to utilize the 3LO benefits. The experiments performed on the set of problems of various features (additively separable, overlapping, employing various deceptive trap functions) show that the incorporation of DLED may lead to high quality results. The results also show that using computationally less expensive DLED instead of 3LO (in the case of 3LOa) does not decrease the method effectiveness.

The rest of this paper is organized as follows. In Sections 2 and 3, we present SLL and SLL-using EAs. The idea of DLED is presented in Section 4, while Section 5 describes its incorporation into considered methods. Finally, the last section summarizes this paper and presents the most important future work directions.

2 STATISTICAL-BASED LINKAGE LEARNING

Linkage, i.e., possible gene dependencies, may be obtained in many various ways [8, 27, 33]. Many SLL-using EAs have been proposed recently [6, 14, 17, 19, 22, 30]. In SLL, the model is built using the statistical analysis of gene values in the population. Mutual information (MI) is employed frequently (MI) [20] to measure the dependency between two random variables X and Y: $I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)}$ In terms of EAs, both variables refer to genes, and the given probabilities are calculated based on the existence of specific gene values in the population.

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459333

MI of each gene pair is stored in the Dependency Structure Matrix (DSM) derived from the organization theory [17]. The main goal of linkage learning is to exploit the underlying problem structure. This structure consists of Building Blocks (BBs), i.e., groups of interdependent genes. Therefore, DSM is then processed to obtain a more complex structure that clusters larger groups, e.g., linkage tree (LT) [27, 30] or incremental linkage set (ILS) [17]. LT is a hierarchical structure that leaves are single gene indexes, and a root is a group of all gene indexes. LT is created from bottom to top, merging the most dependent nodes. In ILS, initially, the set contains the start gene index, and then consecutively, the most dependent vertices are incrementally added to it.

3 SLL-USING EAS

Exploiting the underlying problem structure by utilizing linkage learning techniques is used by modern EAs to make the search more efficient and effective [4, 14, 17, 19, 27, 30]. In this section, we describe three state-of-the-art optimizers, namely LT-GOMEA [4, 30], P3 [14], and Population-sizing Dependency Structure Matrix Genetic Algorithm-II with Comparative Mixing (psDSMGA-II-CM) [5, 19]. All are parameter-less and employ SLL.

LT-GOMEA maintains many subpopulations of different sizes [4, 16]. For each subpopulation, separated LT is created and updates. The nodes of LT (clusters of dependent genes) take part in Optimal Mixing (OM) [32] that replaces crossover. In OM, the *source* individual is updated by replacing its genes marked by a cluster with the *donor* individual's genes. If this replacement operation does not decrease the source's fitness, it is preserved or reverted otherwise. When LT-GOMEA detects the lack of improvements during OM, then the forced-improvements (FI) phase is executed. In FI, the best individual found so far is used as a donor.

P3 uses the same SLL technique as LT-GOMEA and employs LTs and OM. However, the population size in P3 is not fixed and resembles a pyramid. The population is divided into subpopulations called levels. A new individual is created and added to the pyramid at every iteration. It is optimized with the First Improvement Hill Climber (FIHC) [14] and climb the pyramid from bottom to top. An individual is added to a higher level only if it was improved by OM and has not been added to any level before. psDSMGA-II-CM is a recent proposition that enhances DSMGA-II [17] by replacing the restricted mixing operator with the comparative mixing (CM) operator. CM filters ILS in order to obtain more diverse linkage. Such filtering improves the method's effectiveness.

4 DIRECT LINKAGE EMPIRICAL DISCOVERY

The recently proposed 3LO was proven never to report *false linkage* [27]. Thus, it may propose linkage of significantly higher quality than SLL. However, the price for 3LO advantages is its high computational cost. To discover linkage, 3LO optimizes a genotype of an individual with FIHC. It compares the results of this optimization with the results of the optimization of the same genotype with one gene value disturbed. Since FIHC stops only when no improvement is found during its whole iteration. Thus, the execution of FIHC may be expensive. Let us consider the example that uses the order-k bimodal deceptive functions of unitation defined as follows.

$$bimodal\_trap(u) = \begin{cases} k/2 - |u - k/2| - 1 & , u \neq k \wedge u \neq 0 \\ k/2 & , u = k \vee u = 0 \end{cases} \quad (1)$$

where u is *unitation* (the sum of binary gene values) and k is the function size.

In the proposed example, as the optimization problem, we employ the concatenation of three order-4 bimodal deceptive functions (the values of the considered functions for the consecutive unitations from 0 to 4 are: 2, 0, 1, 0, 2). The individual we discover the linkage from is $\overrightarrow{x}$ = [0010 0100 0011]. For this example, the FIHC optimization order will be the reverse of the gene order. First, we obtain $opt(\overrightarrow{x})$ = [0011 0101 0011], for which we require two FIHC iterations ($2n + 1$ FFE). Then, we perturb the first gene and obtain $\overrightarrow{x^{(1)}}$ = [1011 0101 0011], after the first FIHC iteration (that costs $n + 1$ FFE) we obtain $opt(\overrightarrow{x^{(1)}})$ = [1010 0101 0011], the second iteration of FIHC will not make any changes but will cost another $n + 1$ FFE. In this case, computing *linkage scraps* for all the genes in the genotype will consume $n \cdot (n + 1) + n + 1 = n^2 + 2n + 2 = 170$ FFE. Such operation is expensive, but each discovered linkage scrap, e.g., $1 * *1$ $* * * *$ $* * * *$ (all genes position marked with '1' are considered dependent) for the first gene, will be a true part of an existing building block (single deceptive function).

Let us now consider an overlapping problem. We employ the same three order-4 bimodal deceptive functions, but the first function is defined on genes number 1-4, the second on genes 4-7, and the third on genes 7-10. Thus, we may say that the functions overlap by 1 bit. We employ the same FIHC optimization order as in the previous example and $\overrightarrow{x}$ = [0110100010]. This linkage discovery example is shown in Figure 1. First, we obtain $opt(\overrightarrow{x^{(1)}})$ (this procedure improves the fitness of the second and the third block). Then, we perturb the first gene and execute FIHC for the first time. The difference to the optimization of $\overrightarrow{x}$ is that the fourth gene will be changed to '1' (this will decrease the value of the second block by one but increase the value of the first block by two). During the second iteration of FIHC, the same situation will take place for the 7^{th} gene. The third FIHC iteration will optimize the 8^{th} gene. Finally, the fourth FIHC iteration will not make any changes. The linkage discovery only for the first gene will cost $4n + 1$ FFE, which seems expensive. The proposed *linkage scrap* will identify the dependency between genes {1,4,7,8}. This information is true. However, it is intuitive to state that gene 1 is mostly dependent on genes {2,3,4}, less dependent on genes {5,6,7}, and least dependent (although still dependent) on genes {8,9,10}. Thus, it is desirable to detect the *direct* dependencies rather, than all genes dependent on the particular gene. Thus, as a *direct linkage* of gene g, we consider those genes that belong to the same building block as g.

Therefore, the main motivation of this paper is to propose a new linkage learning technique that will have the following features:

- Same as 3LO never reports the *false linkage*.
- Its computational cost will be low enough to apply it to state-of-the-art methods - LT-GOMEA and P3.
- It will only discover the directly dependent genes.

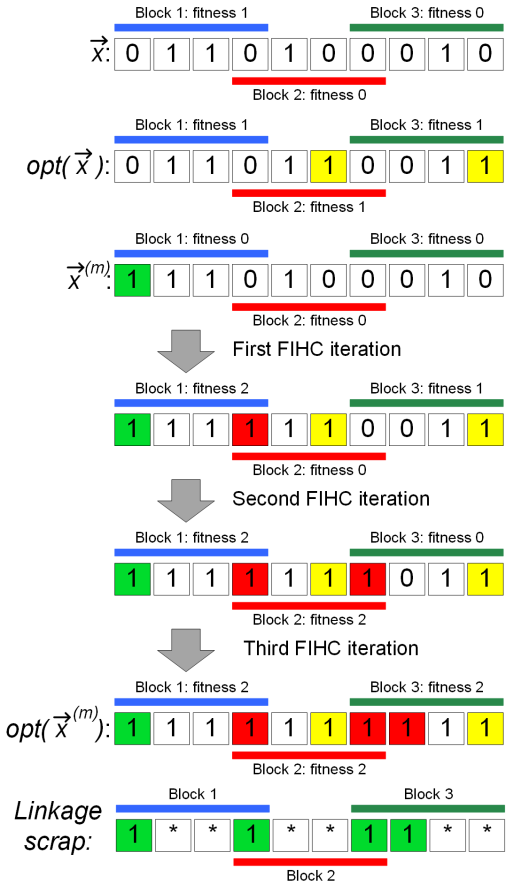

Figure 1: 3LO linkage discovery for problem with overlaps

To meet the above requirements, we propose the Direct Linkage Empirical Discovery technique (DLED). DLED is presented in Pseudocode 1 and is dedicated to binary search spaces. However, it can be easily extinguished to other discrete non-binary domains other than permutation-based. Same as in 3LO, the linkage is discovered from a particular individual and is discovered for a particular gene. However, no greedy optimization is employed. In DLED, we introduce a perturbation to the gene we discover the linkage for (line 4). However, to discover the dependencies between the considered gene (*geneNo*) and the rest of the genotype, for each other gene (*otherNo*), we check the fitness change that will be triggered by flipping the other gene's value (lines 11-14). If flipping the other gene's value improves the fitness of one genotype (original or perturbed) but does not improve the fitness of the other genotype, then we find the other gene (*otherNo*) dependent on the gene (*geneNo*) we discover the fitness for (lines 15-24). Note that after each dependency check, the states of the original and perturbed genotype are restored (lines 9-10).

The DLED procedure requires $2n+2$ FFE for linkage discovery for a single gene. Thus, it requires $2n^2+2n$ FFE for the whole genotype. This value seems similar to the cost of 3LO. However, for DLED, this required number of FFE is an exact value and can not increase.

Pseudocode 1 Direct Linkage Empirical Discovery

```
1:  function DLED(geneNo, ind)
2:      fitOrig ← Fitness(ind);
3:      indPert ← ind;
4:      indPert[geneNo] ← FlipGene(indPert[geneNo]);
5:      fitPert ← Fitness(indPert);
6:      depGenes ←< empty >
7:      for each otherNo ∈ ind do
8:          if otherNo ≠ geneNo then
9:              indMod ← ind;
10:             indPertMod ← indPert;
11:             indMod[otherNo] ← Flip(ind[otherNo]);
12:             indPertMod[otherNo] ← Flip(indPert[otherNo]);
13:             fitOrigMod ← Fitness(indMod);
14:             fitPertMod ← Fitness(indPertMod);
15:             if fitOrig < fitOrigMod then
16:                 newValBetter ← true
17:             else
18:                 newValBetter ← false
19:             if fitPert < fitPertMod then
20:                 pertNewValBetter ← true
21:             else
22:                 pertNewValBetter ← false
23:             if newValBetter ≠ pertNewValBetter then
24:                 depGenes ← depGenes + otherNo
25:     return depGenes;
```

Thus, the cost of linkage discovery presented in Figure 1 for DLED will be 22 instead of 170 required by 3LO.

The other significant difference between DLED and 3LO is the linkage obtained for overlapping problems. In DLED, we obtain only those genes that are directly dependent on the gene we discover the linkage for (*geneNo*). For instance, for the example considered in Figure 1, the list of genes dependent on the first gene would be: {2,3,4}. No dependence between the first gene and genes 5-10 can be obtained, because in each dependency check operation, the value of gene 4 remains the same. A formal argument is given below.

For $\vec{x} = [x_1, ..., x_n]$ and $I = \{i_1, ..., i_k\} \subset \{1, ..., n\}$ let us denote $\vec{x_I} = [x_{i_1}, ..., x_{i_k}]$.

Definiton 1. For an additively separable problem with overlaps described by a fitness function

$$f(\vec{x}) = \sum_j f_j(\vec{x_{I_j}}), \tag{2}$$

where I_j are (not necessarily disjoint) subsets of $\{1, ..., n\}$, we say that genes x_l and x_m are:

- *directly linked* if l and m belong to a common I_j,
- *linked* if there is a sequence $j_0, ..., j_s$ such that $l = j_0$, $m = j_s$ and x_{j_i}, $x_{j_{i+1}}$ are directly linked for all $i = 0, ...s-1$; in particular, if $s > 1$ we will call x_l and x_m *indirectly linked*,
- *not linked (independent)* if none of the above holds.

Theorem 1. *For an additively separable problem with overlaps given by* (2) *DLED indicates only directly linked genes.*

Proof. Denote by $\overrightarrow{x^{(l_1,...,l_s)}}$ the vector which is equal to $\vec{x}$ on all coordinates except for $l_1, ..., l_s$, where the values are flipped from 0 to 1 or vice versa. DLED reports that genes x_l and x_m are linked if

$$\begin{cases} f(\vec{x}) < f(\overrightarrow{x^{(m)}}) \\ f(\overrightarrow{x^{(l)}}) \geq f(\overrightarrow{x^{(l,m)}}) \end{cases} \quad \text{or} \quad \begin{cases} f(\vec{x}) \geq f(\overrightarrow{x^{(m)}}) \\ f(\overrightarrow{x^{(l)}}) < f(\overrightarrow{x^{(l,m)}}) \end{cases} \tag{3}$$

Fix $l, m \in \{1, ..., n\}$ and define $\mathcal{J}$ as the set of those $j \in \{1, ..., n\}$ for which $l \in I_j$. Assume that x_l and x_m are not directly linked. If $j \in \mathcal{J}$ then I_j does not contain m and, consequently,

$$\begin{cases} \sum_{j\in\mathcal{J}} f_j(\overrightarrow{x_{I_j}}) = \sum_{j\in\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(m)}}) \\ \sum_{j\in\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(l)}}) = \sum_{j\in\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(l,m)}}) \end{cases} . \tag{4}$$

So the validity of (3) depends only on values of f_js for $j \notin \mathcal{J}$. But for sets I_j not containing l we have $\overrightarrow{x_{I_j}} = \overrightarrow{x_{I_j}^{(l)}}$ and $\overrightarrow{x_{I_j}^{(m)}} = \overrightarrow{x_{I_j}^{(l,m)}}$, and consequently $\sum_{j\notin\mathcal{J}} f_j(\overrightarrow{x_{I_j}}) = \sum_{j\notin\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(l)}})$ and $\sum_{j\notin\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(m)}}) = \sum_{j\notin\mathcal{J}} f_j(\overrightarrow{x_{I_j}^{(l,m)}})$. In order to obtain fitness comparison in (3), the expressions from the first equality are added to left hand sides of (4), while the values from the second one are added to the right hand sides, hence, it is not possible to obtain mismatch in either of conditions in (3). □

In 3LO, we assumed that all genes in a single *linkage scrap* are dependent on each other. For instance, in the example presented in Figure 1, we obtain the following linkage scrap: {1,4,7,8}. In 3LO, we assume that *all* genes in a single linkage scrap are dependent on each other (e.g., we assume that the fourth gene is dependent on gene 8). In DLED, we do not make such an assumption. We only consider that the genes obtained in a single DLED linkage discovery operation are dependent on the gene *geneNo*. Therefore, the list of genes obtained from a single DLED operation will be denoted as the Directly Dependent Genes List (DDGL). For the sake of clarity, we introduce the following notation for DDGL: $\{x_i \rightarrow x_{d_1}, x_{d_2}, .., x_{d_r}\}$, where $D = \{d_1, .., d_r\}$ is a subset of $\{1, .., n\}$ and $i \neq d_j \forall d_j$. For instance, the following DDGL $\{1 \rightarrow 2, 3, 4\}$ means that gene 1 was found dependent on genes 2, 3, and 4.

To show the importance of the difference between *linkage scraps* and DDGL, let us consider two DDGLs obtained for the individual $\vec{x}$ presented in Figure 1. As stated before, if we execute DLED for the first gene, we will obtain $\{1 \rightarrow 2, 3, 4\}$. In this case, all genes are dependent on each other (e.g., gene 2 is dependent on gene 4). In figure 2, we present the DLED behavior for the fourth gene of the same individual as in Figure 1. The gene values that improve the fitness of *ind* and *indPert* differ on positions {1,2,3,5,6,7}. Therefore, DDGL will be $\{4 \rightarrow 1, 2, 3, 5, 6, 7\}$. Gene 4 is indeed directly dependent on all these genes. However, it is not true that genes number 1 and 7 are directly dependent on each other. Note that such a way of understanding gene dependencies is consistent with the outcome of Differential Grouping 2 presented in [21].

Finally, the last difference between 3LO and DLED is that 3LO depends on the FIHC optimization order (i.e., depending on the optimization order, 3LO may return different linkage discovery results). In contrary to 3LO, DLED is deterministic.

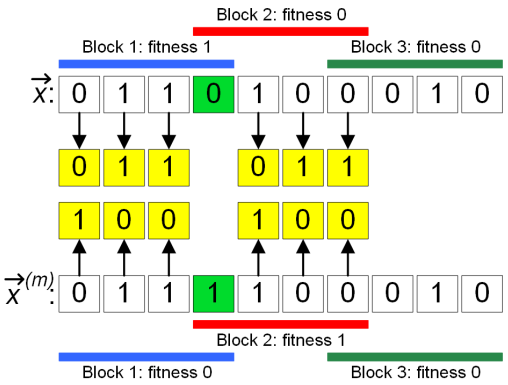

Figure 2: DDGL discovery example

Table 1: The example of DSM obtained from DLED

	1	**2**	**3**	**4**	**5**
1	X	2	1	0	0
2	2	X	2	0	0
3	1	2	X	1	0
4	0	0	1	X	1
5	0	0	0	1	X

To obtain masks for OM, the discovered DDGLs are used to create a DSM-like matrix. Each cell of this matrix is equal to the number of DDGLs that indicate that a particular pair of cells is dependent. For instance, for a 5-bit problem and the following DDGLs $\{1 \rightarrow 2, 3\}$, $\{2 \rightarrow 1, 3\}$, $\{3 \rightarrow 2, 4\}$, $\{4 \rightarrow 5\}$, we will obtain the DSM presented in Table 1. Using DSM created from DDGLs, we create an LT in the same way as in LT-GOMEA and P3. However, each time we generate an LT, we temporarily add a small random value to each DSM entry. This operation prevents the eventual bias that may influence linkage if more than one cell has the same value.

5 DLED INCORPORATION IN MODERN EVOLUTIONARY ALGORITHMS

In the previous section, we have proposed DLED. It has significant advantages over 3LO – it will only discover the direct gene dependencies and is less computationally expensive. To check the potential brought by DLED, we introduce it to LT-GOMEA [4] and P3 [14]. Additionally, we also introduce DLED to the 3LO Algorithm (3LOa) [27] designed to utilize 3LO advantages.

LT-GOMEA employs the population-sizing scheme [16] and maintains many subpopulations. In LT-GOMEA, a separate linkage is maintained for each subpopulation, it is updated at every iteration. DLED introduction, if possible, adjusts to this behavior.

LT-GOMEA-DLED general procedure is presented in Pseudocode 2. It employs a probability of DLED linkage generation that is initially set to one for all genes (lines 2-4). Whenever the population sizing-scheme decides to add a new subpopulation, the DLED linkage generation probabilities are reset (lines 11- 12). As in the

Pseudocode 2 LT-GOMEA-DLED general procedure

```
1:  SubPops ← CretateSubPop(1);
2:  geneProb ← CreateTable(ProblLen)
3:  for each gene ∈ geneProb do
4:      geneProb[gene] ← 1
5:  while ¬StopConditin do
6:      if add new subpop then
7:          largestSubPop ← GetLargestSubPop(SubPops);
8:          newSubPop ← CretateLTGA(largestSubPop.size · 2);
9:          ZeroDSM(newSubPop.DSM);
10:         SubPops ← SubPops + newSubPop;
11:         for each gene ∈ geneProb do
12:             geneProb[gene] ← 1
13:     runMeSubPop ← SelectSubPoptoRun(SubPops);
14:     for each gene ∈ geneProb do
15:         if Random01() < geneProb[gene] then
16:             geneProb[gene] ← geneProb[gene] · 0.5
17:             ind ← RandomlyChooseInd(runMeSubPop);
18:             ddgl ← DLED(gene, ind);
19:             ApplyDDGLToDSM(ddgl, runMeSubPop.DSM);
20:     RunSubPopIteration(runMeSubPop);
```

original LT-GOMEA, at each method iteration, LT-GOMEA-DLED chooses one of the subpopulations (line 13). For this subpopulation (*runMeSubPop*), before executing its iteration, DLED is executed in the following way. For each gene, the probability of linkage generation is checked (line 14). If linkage is generated for a particular gene, then the linkage generation probability that refers to this gene is halved (line 16). The individual for linkage discovery is chosen randomly from *runMeSubPop* subpopulation (line 17). DDGL is executed for the chosen individual and the considered gene (line 18). After the appropriate DSM update, an iteration of *runMeSubPop* subpopulation is executed in a standard way but uses the linkage supported by DLED (line 20).

The information from each *ddgl* found is applied to DSM in a way presented in the previous section (line 19). Additionally, if *ddgl* discovers a dependency between the pair of genes that was not known before, the linkage discovery probabilities for these genes are reset to one. For instance, let us consider DSM from Table 1, $ddgl = \{1 \rightarrow 3, 5\}$, and $geneProb = \{0.5, 0.25, 0.25, 1, 0.25\}$. For such *ddgl*, the values of cells $DSM[1, 3]$ and $DSM[3, 1]$ will be increased from one to two. Concordantly, the values of cells $DSM[1, 5]$ and $DSM[5, 1]$ will be increased from zero to one. Thus, they introduce a new dependency that was not known before. Therefore, the probabilities of linkage generation for genes 1 and 5 will be reset to one. After this operation, the linkage generation probabilities table will be $geneProb = \{1, 0.25, 0.25, 1, 1\}$.

Updating linkage during the method run was found more effective than using the predetermined linkage learning models [31]. Additionally, frequent linkage modifications lead to using a diverse linkage that was recommended in [27]. However, generating DLED linkage for the whole genotype at each method iteration is not a reasonable choice because DLED (although less expensive than 3LO) is still significantly more expensive than the SLL employed by the original LT-GOMEA. The probabilities of linkage generation for each gene were introduced to overcome this issue.

In P3, the population resembles a pyramid, and each of the pyramid levels is a separate subpopulation. Therefore, the general procedure of P3-DDGL is organized as follows. As for LT-GOMEA-DDGL, we employ the *geneProb* table initialized in the same way at the beginning of the method run. Whenever the new level is added to the pyramid, the gene linkage generation probabilities are reset in the same way while adding the new LTGA subpopulation in LT-GOMEA. DLED linkage generation procedure is executed for each level before every P3-DLED iteration. Each pyramid level maintains its separate linkage (each subpopulation in LT-GOMEA-DLED maintains its separate linkage as well). After DLED updates, an iteration of P3 is executed normally, but it uses the DLED-DSM. In P3-DLED (same as in P3), the number of pyramid levels may rise at the beginning of the run, and later on, new levels may be added rarely or never. In such a situation, the probability of linkage changes decreases significantly, which would be disadvantageous for the method's effectiveness. Therefore, in P3-DLED, the linkage generation probabilities are reset whenever the number of individuals stored in the pyramid doubles.

The last method we have introduced DLED into was 3LOa. 3LOa was proposed to use 3LO benefits but limit the linkage generation frequency due to 3LO costs. Thus, in 3LOa, linkage is generated only when the new best individual is found. Therefore, in 3LOa-DLED, we have simply replaced 3LO with DLED. Each time linkage is generated, we execute the DLED procedure for the new best-found individual and all available genes.

6 THE RESULTS

6.1 Test Problems and Experiments Setup

We consider the set of nine different problem types that differ in their nature. These problems were also considered in [14, 19, 27]. Concatenation of deceptive functions [11] is a well-known benchmark. Deceptive functions are found hard because optimizers are attracted by their local optima. Here, we use concatenations of order-3 standard deceptive functions. They have one global and one local optimum. We also consider their extension, step deceptive functions [14], that extends their landscapes by adding plateaus of size s. In consequence, the number of local optima increases. We employ order-3 and order-5 step deceptive functions. We set $s = 2$ for both. Another chosen problem is bimodal functions concatenation [13]. The definition of order-k bimodal function is given in formula (1). There are two global optima and $\binom{k}{k/2}$ local optima in each bimodal function. Here, we choose concatenations of order-10 bimodal functions and their noised version with even more local optima. The values of noised bimodal deceptive function depending on the unitation may be found in [18, 25, 27].

We also include the following problems of the overlapping nature. We consider cyclic concatenations built from step order-5 and bimodal order-10. The overlap is $m = 3$. An overlapping real-world problem is the Ising Spin Glass (ISG). In ISG, a sample solution represents spin values. The best value of ISG of size n minimizes the sum $-\sum_{i,j=1}^{n} x_i x_j J_{ij}$, where $J_i j$ is a coupling constant of the ith and the jth spins represented by x_i and x_j respectively. The structure of ISG is not as much overlapping as in Nearest Neighbor

Table 2: Median population size necessary to find the optimal solution

	LT-GOMEA*			P3**		
	DLED	**SLL**	**rat.**	**DLED**	**SLL**	**rat.**
Dec.3	64	128	0.50	163	43	***3.78***
St.dec.3	1024	4096	0.25	4906	17371	0.28
St.dec.5	16384	32768	0.50	70553	127747	0.55
Bm.10	4096	32768	0.13	27180	142740	0.19
Bm.10n.	1024	65536	0.02	2441	179644	0.01
C.St.d.5	4.E+06	8.E+06	0.50	155594	468595	0.33
C.Bm.10	32768	65536	0.50	110614	447856	0.25
ISG	2048	512	***4.00***	7709	455	***16.94***
NK-land	1024	2048	0.50	3743	6209	0.60
Max3Sat	1024	1024	1.00	3979	4624	0.86

**the size of the largest subpopulation*
***the size of the whole pyramid*

Table 3: Median linkage quality in runs for which the optimal solution was found

	LT-GOMEA			P3		
	n	**DLED**	**SLL**	***n***	**DLED**	**SLL**
Dec.3	*1998*	1.00	0.34	*1998*	0.99	1.00
St.dec.3	*1995*	1.00	0.69	*1995*	1.00	0.92
St.dec.5	*2002*	1.00	1.00	*1199*	1.00	0.83
Bim.10	*2000*	1.00	1.00	*1600*	1.00	0.99
Bim.10 n.	*2000*	1.00	0.99	*800*	1.00	0.71

NK-landscapes. In the chosen instances, each gene is dependent on five consecutive genes. When a gene is located at the end of the genotype, its neighbors are taken from the beginning. The last considered problem is Max3Sat. Max3Sat consists of clauses containing three logical variables. Our goal is to find such gene values that maximize the number of satisfied clauses.

In this paper, we consider LT-GOMEA and P3 in both versions – the original (LT-GOMEA-SLL and P3-SLL) and the DLED-based. Additionally, we employ 3LOa to check if 3LOa-DLED is competitive with the original 3LOa that employs 3LO. Finally, we use psDSMGA-II-CM [19] to check if the proposed DLED-based EAs are competitive with other state-of-the-art EAs. Each experiment is repeated 20 times. To verify the statistical significance of the results differences, we employ the unpaired Wilcoxon test and a significance level of 5%. Some of the considered EAs incorporate fitness caching [26, 27]. Thus, the FFE-based stop condition may be unfair. Therefore, each experiment was assigned 12 hours of computation time on the PowerEdge R430 Dell server (Intel Xeon E5-2670 2.3 GHz 64GB RAM). To ensure the fairness of the comparison, the number of computation processes was always one less than the number of available CPU nodes. All experiments were single-threaded, and no other resource-consuming processes were running.

6.2 DLED Computation Costs and Quality

The objective of this paper is to introduce ELL into the state-of-the-art SLL-using EAs. To this end, we have proposed DLED that is less computationally expensive than 3LO. The introduction of DLED into LT-GOMEA and P3 raises the following questions, which we address in this section.

- How DLED influences LT-GOMEA and P3 run?
- Is DLED-linkage of higher quality than SLL-linkage?
- What is the cost of DLED linkage generation?

In Table 2, we present the maximum subpopulation size and the total pyramid size at the end of the run for LT-GOMEA and P3, respectively. We consider only those runs in which the optimal solution was found. For most of the considered problems, DLED versions required a smaller population size to find the optimal solution. If the optimal solution is found for the smaller population size (both methods automatically adjust their population size during the run), then it indicates that DLED proposes the linkage of a higher quality for most of the considered problems. Note that the $dledMaxPop/sllMaxPop$ ratio presented in Table 2 for some problems is below 0.25 As shown in the next section, for these problems, the DLED versions of LT-GOMEA and P3 may significantly outperform their SLL-based predecessors. For two problems, the situation is the opposite – the ratio is higher than 1 (up to almost 17 for ISG-based P3-DLED and P3-SLL comparison). For these problems, SLL-based EAs perform better. Such a situation is expected for order-3 deceptive functions concatenation – in [25], SLL was formally shown to propose a high-quality linkage for this kind of problems quickly. According to ISG, the reasons why SLL seems to be more suitable for decomposing this problem requires further investigation.

In Table 3, we compare the linkage quality based on the *Fill* measure [25]. The highest Fill value is 1 and corresponds to the so-called *perfect linkage*, while the linkage of the lowest possible quality is assigned 0. Since LT-GOMEA and P3 maintain many subpopulations, and each subpopulation has its own linkage, we report the fill value of the linkage that is of the highest quality from all available. The comparison is limited to partially additively separable problems because the employed measure is inappropriate for overlapping problems. As presented in Table 3, SLL can propose linkage of high quality for bimodal, bimodal noised, and step deceptive problems. However, to obtain such high-quality linkage, it may require an even 70-times larger population and a long optimization process (the higher the quality of individuals is, the better SLL-linkage quality should be). In some cases (e.g., SLL-linkage quality for P3 and bimodal noised problems), even a 12-hour run and the population of over $3 \cdot 10^6$ individuals is not enough to discover linkage of a quality that would be close to to the quality of DLED-linkage.

The results indicate that, for most of the considered problems, DLED supports a high-quality linkage that allows for fast optimization. In Table 4, we present FFE and computation time percentage spent on DLED. Note that for all considered deceptive function concatenations, this percentage is high. Except for one case, it is higher than 50%. Moreover, for some considered problems, it is close to 100%. Such a result seems intuitive and consistent with observations presented in [25] – deceptive function concatenations become easy to solve for a GA-like optimizer if linkage of high quality is supported. Additionally, the effectiveness of such optimizers decreases quickly with the decrease of linkage quality [25].

Table 4: DLED costs - FFE and time percentage spent on linkage discovery

	LT-GOMEA			P3		
	n	**FFE**	**Time**	***n***	**FFE**	**Time**
Dec.3	2000	0.99	0.99	2000	>0.99	0.96
St.dec.3	2000	0.95	0.94	2000	0.97	0.77
St.dec.5	2000	0.62	0.60	800	**0.57**	**0.68**
Bim.10	2000	0.76	0.75	1200	0.83	0.73
Bim.10 n.	2000	0.90	0.90	2000	**0.39**	**0.76**
Cyc.St.dec.5	150	<0.01	<0.01	100	**<0.01**	**0.34**
Cyc.Bim.10	2000	0.23	0.23	600	**0.23**	**0.68**
ISG	784	0.44	0.48	784	0.78	0.81
NK-land	600	0.82	0.80	600	0.93	0.89
Max3Sat	150	0.56	0.41	150	0.75	0.73

For the considered overlapping problems, the DLED discovery cost differs significantly from below 1% (cycling order-5 step deceptive functions concatenation) up to over 90 % (P3-DLED for NK landscapes). For these problems, it is hard to justify why DLED is improving (e.g. cycling bimodal functions concatenation) or decreasing (ISG) the effectiveness of the method. This issue requires further investigation. A promising research direction may be proposing linkage quality measures for the overlapping problems and using these measures to execute EAs using an artificial linkage of a given quality.

For some problems, the DLED FFE-cost percentage differs significantly from the time-cost one (e.g., P3-DLED solving the cyclic problems). The reason for this situation is as follows. P3 employs population-based fitness caching [26] – before it computes the fitness of any individual, it first checks if such individual is a part of its current population. If so, then P3 copies the fitness value from the same, already rated, individual, and no FFE is spent. Despite its simplicity, such optimization may lead to a significant reduction of the computation costs. However, it may also lead to the following situation. When DLED linkage discovery is triggered, the fitness is computed for many genotypes similar to the one we discover the linkage from. If P3 is stuck and the pyramid is large, likely, the genotypes considered during DLED are already represented in the pyramid. If so, then for each fitness check performed during DLED, the computation time will be spent on checking if the genotype exists in the pyramid, but no FFE will be spent (the same, already rated genotype will be found in the pyramid). Thus, the computation time will increase, but FFE will not. In such situations, the FFE-based stop condition does not seem reliable. More information considering the influence of fitness caching on the computation time and FFE-based stop condition reliability may be found in [26]. The above results and their analysis confirm that using the time-based stop condition in the experiments considered in this paper is justified.

6.3 Main Results

In Table 5, we present the comparison between the proposed DLED and SLL based on LT-GOMEA and P3. We compare the percentage of optimal solutions found by DLED and SLL versions (or 3LO version for 3LOa). We consider the largest problem size for which at least one of the competing versions has found the optimal solution in at least 50% of the runs. If the differences in the successful runs percentage are not statistically significant, then we compare FFE necessary to find an optimal solution. If these differences are equal, then we find DELD and SLL versions equally effective for the particular problem. We employ +/-/= signs to mark if DLED was statistically better, worse, or equal than SLL/3LO. Such comparison does not take into consideration the size of the differences. Therefore, in the same table, we propose a ranking that considers the percentage of successful runs first and the median FFE until finding the optimal solution if this percentage is equal and higher than 50%. If the successful runs percentage is below 50% the method is assigned the lowest possible rank. The results show that introducing DLED into LT-GOMEA and P3 is beneficial for many of the considered problems. A similar situation takes place for 3LOa, which shows that replacing 3LO with DLED, in general, preserves 3LOa-3LO advantages. LT-GOMEA-DLED obtains the best results, while LT-GOMEA-SLL takes second place. The median ranking of both 3LOa versions and pcDSMGA-II-CM is similar. Finally, both P3 versions take two last places. Nevertheless, P3-DLED seems to be able to solve a larger set of problems than P3-SLL. Therefore, it outperforms its predecessor. However, this feature has its price – P3-DLED is slower in solving problems, for which P3-SLL works particularly fast.

Table 5: General comparison

	LT-GOMEA DLED/SLL	**P3 DLED/SLL**	**3LOa DLED/3LO**	**ref* SLL**
Dec.3	5 / 2 (-)**	7 / 1 (-)	4 / 6 (+)	3
St.d.3	4 / 3 (-)	5 / 7*** (+)	1 / 2 (+)	6
St.d.5	2 / 3 (+)	7 / 7 (=)	7 / 1 (-)	7
Bm.10	3 / 4 (+)	7 / 7 (=)	2 / 1 (=)	7
Bm.10n.	3 / 7 (+)	4 / 7 (+)	1 / 2 (+)	7
C.St.d.5	2 / 7 (+)	7 / 7 (=)	7 / 7 (=)	1
C.Bm.10	1 / 7 (+)	7 / 7 (=)	7 / 7 (=)	7
ISG	7 / 2 (-)	6 / 1 (-)	5 / 4 (=)	3
NK-land	3 / 2 (-)	5 / 1 (-)	7 / 6 (-)	3
Max3Sat	6 / 5 (=)	3 / 1 (=)	7 / 7 (=)	4
Med.Rank.	**3** / **3.5**	6.5 / 7	6 / 5	5
Avg.Rank.	**3.6** / **4.2**	5.8 / 4.6	4.8 / 4.3	4.8
	5+ / 3- / 1=	2+ / 3- / 5=	3+ / 2- / 5=	N/A

**psDSMGA-II-CM*
***the symbol in the brackets indicates if DLED version is statistically better, worse or equal*
****underline – the percentage of optimal solutions is below 50%*

In Section 4, we prove that DLED will only report the direct linkage. This feature was not directly utilized by any of the EAs we have introduced DLED in. Proposing the appropriate mechanisms that do so is the future work. However, both, LT-GOMEA-DLED and P3-DLED, perform significantly better for both considered cycling trap problems. As shown in the example in Section 4, 3LO may not be suitable for decomposing such problems. Thus, it is reasonable to assume that a precise discovery of a directed linkage only influences LT-GOMEA and P3 performance in solving the cycling trap problems.

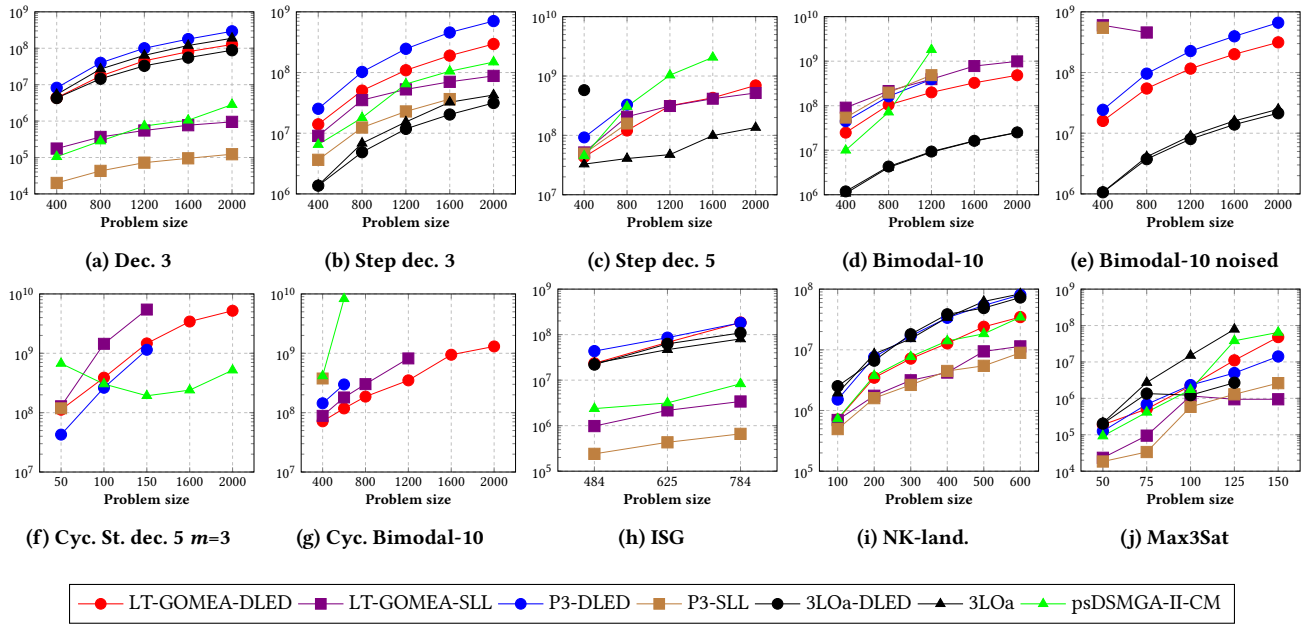

Figure 3: FFE-based scalability on the considered problems (the results with at least 50% optimal solutions found)

In Figure 3, we present the scalability of the considered EAs. Note that LT-GOMEA-DLED is the only method that has found the optimal solution in at least 50% of the runs for all considered problems (see Table 5). LT-GOMEA-DLED performs significantly better than LT-GOMEA-SLL for all problems using step order-5, bimodal, and noised bimodal deceptive functions. For these problems, SLL requires significantly more computation resources than DLED to discover linkage of quality that is high enough. On the other hand, for ISG and NK landscape problems, LT-GOMEA-DLED requires more FFE due to the DLED costs.

The P3-based DLED and SLL comparison seems similar to the one considering LT-GOMEA. P3-DLED performs better for problems using bimodal functions, while P3-SLL is a better choice for order-3 standard deceptive function concatenations (to recall, SLL decomposes this kind of problems quickly and precisely [25]). The SLL version is also faster in solving ISG and NK fitness landscape problems due to DLED linkage generation costs. The surprising observation is that even equipped with the perfect or near-perfect linkage, P3 cannot solve the longer deceptive functions concatenations. This is caused by the fact that the population in P3 has an unlimited size (see Figure 3d – P3-DLED is unable to solve bimodal functions concatenations of the length larger than 1200 bits). The individuals that were added to the pyramid will not be improved any more. Thus, the lack of individual-improvement pressure significantly slows down the convergence. On the other hand, the P3 framework seems to be particularly suitable for solving Max3Sat.

For most of the deceptive function concatenations, the most effective EAs are both versions of 3LOa. Such results are expected because among all 3LOa employs a search through a limited number of dependent gene combinations. Thus, if the problem contains additively separable parts that are precisely recognized, then the 3LOa procedure should find the optimal solution efficiently. Only for the concatenation of order-5 step deceptive functions, the performance of 3LOa-DLED is low. This seems to be caused by the 3LOa procedure that rarely discovers linkage (originally, it was adjusted to 3LO, which is more expensive than DLED). On the other hand, both 3LOa versions fail to solve even the shortest test cases of cyclic step deceptive functions concatenations.

7 CONCLUSION AND FUTURE WORK

The main objective of this paper was to introduce ELL into state-of-the-art methods that were originally proposed with SLL techniques. To obtain this objective, we have proposed DLED – the new ELL technique faster than its predecessor. Since SLL-using EAs frequently update the linkage, we have proposed a mechanism defining the linkage discovery probability for each gene. The experiments show that the proposed DLED-using EAs, especially LT-GOMEA-DLED, are competitive with other state-of-the-art EAs. DLED seems to be more suitable for solving problems built from non-standard deceptive functions (additively separable and overlapping). The results indicate that the hybridization of DLED and SLL may lead to the further improvement of results quality. Another important future research direction is proposing the linkage quality measures for the overlapping problems. These measures shall help in understanding the EA-based overlapping problems optimization process (including the explanation why using DLED is sometimes advantageous ofer using SLL and sometimes it is not). In this paper, we prove that the proposed DLED will always discover a direct linkage. Therefore, a promising research direction will be proposing a new linkage representation that utilizes this feature.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Rubén Aguilar-Rivera, Manuel Valenzuela-Rendón, and J.J. Rodríguez-Ortiz. 2015. Genetic algorithms and Darwinian approaches in financial applications: A survey. *Expert Systems with Applications* 42, 21 (2015), 7684–7697. https://doi.org/10.1016/j.eswa.2015.06.001

[2] Md Asafuddoula, Tapabrata Ray, and Ruhul Sarker. 2011. An adaptive differential evolution algorithm and its performance on real world optimization problems. In *2011 IEEE congress of evolutionary computation (CEC)*. IEEE, 1057–1062.

[3] Maumita Bhattacharya, Rafiqul Islam, and Jemal Abawajy. 2016. Evolutionary optimization: a big data perspective. *Journal of network and computer applications* 59 (2016), 416–426.

[4] 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.

[5] Chia-hua Chang and Tian-Li Yu. 2016. Investigation on Parameterless Schemes for DSMGA-II. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion)*. ACM, 85–86.

[6] Ping-Lin Chen, Chun-Jen Peng, Chang-Yi Lu, and Tian-Li Yu. 2017. Two-edge Graphical Linkage Model for DSMGA-II. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Berlin, Germany) *(GECCO '17)*. ACM, 745–752.

[7] Stephen Chen, James Montgomery, and Antonio Bolufé-Röhler. 2015. Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution. *Applied Intelligence* 42, 3 (2015), 514–526.

[8] Ying-ping Chen, Tian-Li Yu, Kumara Sastry, and David E. Goldberg. 2007. A Survey of Linkage Learning Techniques in Genetic and Evolutionary Algorithms. Illinois Genetic Algorithms Library, Tech. Rep..

[9] Alberto Colorni, Marco Dorigo, Francesco Maffioli, Vittorio Maniezzo, GIOVANNI Righini, and Marco Trubian. 1996. Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research* 3, 1 (1996), 1–21.

[10] Swagatam Das and Ponnuthurai N Suganthan. 2010. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur University, Nanyang Technological University, Kolkata* (2010), 341–359.

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

[12] Kaizhou Gao, Zhiguang Cao, Le Zhang, Zhenghua Chen, Yuyan Han, and Quanke Pan. 2019. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA Journal of Automatica Sinica* 6, 4 (2019), 904–916.

[13] David E. Goldberg, Kalyanmoy Deb, and Jeffrey Horn. 1992. Massive Multimodality, Deception, and Genetic Algorithms. *Urbana* (1992).

[14] 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.

[15] 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

[16] 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.

[17] 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.

[18] Marcin M. Komarnicki and Michal W. Przewozniczek. 2017. Parameter-less Population Pyramid with Feedback. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17)*. ACM, 109–110.

[19] Marcin M. Komarnicki, Michal W. Przewozniczek, and Tomasz M. Durda. 2020. Comparative Mixing for DSMGA-II. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 708–716. https://doi.org/10.1145/3377930.3390223

[20] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statist.* 22, 1 (03 1951), 79–86. https://doi.org/10.1214/aoms/1177729694

[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] 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

[25] 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.

[26] 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.

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

[28] Michał Witold Przewoźniczek. 2020. Subpopulation initialization driven by linkage learning for dealing with the Long-Way-To-Stuck effect. *Information Sciences* 521 (2020), 62–80. https://doi.org/10.1016/j.ins.2020.02.027

[29] Michał Witold Przewoźniczek, Krzysztof Walkowiak, Arunabha Sen, Marcin Komarnicki, and Piotr Lechowicz. 2019. The transformation of the k-Shortest Steiner trees search problem into binary dynamic problem for effective evolutionary methods application. *Information Sciences* 479 (2019), 1–19. https://doi.org/10.1016/j.ins.2018.11.015

[30] 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.

[31] 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.

[32] 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.

[33] Tian-Li Yu, David E. Goldberg, Kumara Sastry, Claudio F. Lima, and Martin Pelikan. 2009. Dependency Structure Matrix, Genetic Algorithms, and Effective Recombination. *Evol. Comput.* 17, 4 (2009), 595–626.