On measuring and improving the quality of linkage learning in modern evolutionary algorithms applied to solve partially additively separable problems

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

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

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

ABSTRACT

Linkage learning is frequently employed in modern evolutionary algorithms. High linkage quality may be the key to an evolutionary method's effectiveness. Similarly, the faulty linkage may be the reason for its poor performance. Many state-of-the-art evolutionary methods use a Dependency Structure Matrix (DSM) to obtain linkage. In this paper, we propose a quality measure for DSM. Based on this measure, we analyze the behavior of modern evolutionary methods. We show the dependency between the linkage and the effectiveness of the considered methods. Finally, we propose a framework that improves the quality of the linkage.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**;

KEYWORDS

Linkage Quality Measurement, Genetic Algorithm, Estimation-of-Distribution Algorithm, Linkage Learning, Model Building

ACM Reference Format:
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 *Genetic and Evolutionary Computation Conference (GECCO '20), July8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390242

1 INTRODUCTION

Linkage learning techniques are used by evolutionary methods to detect dependencies between genes encoding a solution to an optimized problem. Such knowledge, if it is precise, allows for the detection of the problem structure, and, if used properly, may significantly increase the effectiveness of an evolutionary method. For instance, if an optimizer is given an accurate knowledge about the problem nature, the number of expected fitness function evaluations (FFE) necessary to find an optimal solution may scale linearly with the problem size [16]. Therefore, in black-box optimization, it is important to obtain a high-quality linkage that may be decisive for the effectiveness of a method [15]. Thus, linkage quality measures are necessary to gather knowledge about the quality of linkage obtained by an evolutionary method. Such measures are also necessary to better understand an evolutionary method behavior and analyze the reasons lying behind the successful runs as well as behind the failures. In [10], authors propose linkage quality measures that allow telling if each independent subcomponent of a problem is completely detected or not. To the best of our knowledge, these are the only linkage quality measures proposed that far. Such measures inform only if a *perfect linkage* was found or not [13]. Therefore, they are not enough – there is a significant difference between the correct detection of 90% of the subcomponent and detecting only 10% of it.

This paper concentrates on solving *partially additively separable problems* defined as follows. The problem is partially additively separable if it has a following general form [9]:

$$f(\overrightarrow{x}) = \sum_{i=1}^{r} f_i(\overrightarrow{x}_i) \qquad (1)$$

where $\overrightarrow{x} = [x_1, ..., x_n]$ is a global decision vector of n dimensions, $\overrightarrow{x_i}$ are mutually exclusive decision vectors of f_i, and r is the number of independent subcomponents.

Many of the typical benchmark problems are partially additively separable. However, such problems also conveniently represent the modular nature of many real-world problems [9]. Therefore, optimization methods that successfully solve such problems are valuable in both: theory and practice.

The objectives of this paper are as follows. We propose a linkage quality measure that allows checking how precise is a particular set of linkage information even if it is not perfect. We define what is the *perfect linkage* stored in the Dependency Structure Matrix (DSM) that is frequently employed to represent linkage [4, 6, 14]. For one of the types of considered optimization problems, we formally estimate the population size necessary to obtain a perfect linkage. Using a set of state-of-the-art methods and the proposed technique of artificial linkage creation, we empirically check what is the minimum linkage quality to successfully solve different types of partially additively separable problems. We measure and present the linkage quality obtained by state-of-the-art evolutionary methods on the set of considered problems. On this basis, we analyze the behavior of the considered methods and identify the premises of their success or failure. Finally, we propose the Unbiased Linkage Improver (ULI)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '20, July8–12, 2020, Cancún, Mexico
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390242

framework to improve linkage that is obtained on the evolutionary method run. ULI may be applied to any evolutionary method. We show that despite its simplicity, it may significantly improve the effectiveness of an evolutionary method on the problems for which a high-quality linkage is hard to find.

The rest of the paper is organized as follows. In the next section, we briefly present related work. In Section 3, we present the formal estimation of the minimum population size necessary to obtain a perfect linkage. In the fourth section, we give a formal definition of the proposed linkage quality measure. In Section 5, we propose the ULI framework. The results and analysis of all performed experiments are presented in Section 6. Finally, in the last section, we conclude this paper and highlight the main future work directions.

2 RELATED WORK

One of the important advances in the domain of single-objective black-box optimization was employing the Dependency Structure Matrix (DSM) to obtain linkage in the evolutionary methods [6, 15]. The concept of DSM is derived from the organization theory. DSM is a square matrix, and its entries indicate the dependencies between some components. The single DSM entry $d_{i,j} \in R$, denotes the relationship between the i^{th} and j^{th} components. The higher value of the entry indicates the higher dependency between the components related to the entry. When DSM is employed to obtain linkage in evolutionary methods, a single component is represented as a single gene. Many measures were used that far to measure pairwise dependencies between genes [4, 6, 15]. Here, we use the mutual information measure [6] to illustrate the process of DSM creation.

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{2}$$

where both X and Y are random variables. Note that $\forall_{X,Y}\ I(X;Y) \geq 0$ and $I(X;Y)$ equals 0 when X and Y are independent, because then $p(x,y) = p(x)p(y)$.

To compute DSM values for a given population, the gene value frequencies are counted. The main goal of creating DSM is to extract some useful information about groups of genes that are dependent on one another. Since DSM contains only pairwise dependency values, a clustering algorithm is necessary to merge pairs of genes into larger groups. There are different algorithms [2, 4, 6, 14] that can meet this objective. Here, we describe a hierarchical clustering algorithm that frequently used. In [4, 14] hierarchical clustering algorithm uses the $D(G_i, G_j)$ distance measure. For the i^{th} and j^{th} genes, it is calculated using the mutual information and joint entropy:

$$D(G_i, G_j) = \frac{H(G_i, G_j) - I(G_i, G_j)}{H(G_i, G_j)} \tag{3}$$

where

$$H(G_i, G_j) = -\sum_{g_i \in G_i} \sum_{g_j \in G_j} p_{i,j}(g_i, g_j) \ln p_{i,j}(g_i, g_j) \tag{4}$$

The hierarchical clustering algorithm creates the linkage tree. The linkage tree consists of nodes (i.e., clusters), which are the groups of genes that are considered to be related. The linkage tree creation procedure is greedy because, during its every step, two nearest clusters (according to the distance measure) are merged. Initially, clusters containing only a single gene are created. Then, the merging operation is repeated until only one cluster consisting of all genes remains.

One of the methods that employ linkage trees is the Linkage Tree Gene-pool Optimal Mixing Evolutionary Algorithm (LT-GOMEA), which was proposed in [1]. To exchange information between individuals, LT-GOMEA uses the optimal mixing (OM) operator. In OM, the genes from the *donor* individual, marked by the cluster, replace genes in the *source* individual. If the fitness of the source decreases, then the operation is reversed. Otherwise, the source remains modified. LT-GOMEA uses the population-sizing scheme [5], which makes it parameter-less (otherwise, the only parameter would be the population size). LT-GOMEA maintains its multiple instances, each with different population size. It starts with a population containing only one individual. Then, the instance with a doubled population is executed at each 4^{th} iteration. The instances are found useless if all of its individuals are the same, or its average population fitness is worse than the average fitness of at least one instance with a larger population size. If an instance is found useless, then all instances with a smaller population are found useless as well.

Parameter-less Population Pyramid (P3) [4] also employs OM and linkage tree. The population of P3 has a pyramid-like structure, where each *level* is a separate subpopulation with its own DSM. At each iteration, P3 creates a new random individual and optimizes it with the First Improvement Hill Climber (FIHC). For an individual $\overrightarrow{x}$ that is a vector of n decision variables $[x_1, ..., x_n]$ FIHC works as follows. Considering all genes in a random order, it checks all available values for each gene. The first value that leads to fitness improvement over the original x_i value replaces the original one. This procedure is executed until the iteration of FIHC will not change any gene. After FIHC, the new individual is mixed with OM with all individuals in the pyramid, starting from the lowest level. If this operation improves an individual, then it is added to an upper pyramid level (the new pyramid level is created if necessary).

Dependency Structure Matrix Genetic Algorithm II (DSMGA-II) [6] employs DSM differently to LT-GOMEA and P3. It creates the *incremental linkage set* that is a sequence of dependent genes. The first gene is selected randomly. The next gene is the one that is not in the sequence yet and is the most linked to the gene that is the last in the sequence, and so on. To improve an individual, the genes marked by an incremental linkage set are flipped (this operation is called *restricted mixing*). If restricted mixing decreases the fitness, then the changes are reverted. Otherwise, the change made by restricted mixing is injected to the other individuals in a population (this operation is called the *back mixing*). Recently, the population-sizing DSMGA-II (psDSMGA-II) [8] was proposed. psDSMGA-II uses a similar population-sizing technique as LT-GOMEA [1].

3 OBTAINING THE PERFECT LINKAGE FOR DECEPTIVE TRAP FUNCTIONS CONCATENATIONS

In this section, we consider the problems built from the concatenation of m deceptive trap functions. The single deceptive order-k trap function is defined as follows.

$$dec(u) = \begin{cases} k-1-u & \text{if } u < k \\ k & \text{if } u = k \end{cases} \tag{5}$$

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

We estimate the probability of obtaining the DSM containing the *perfect linkage* for a population of a given size, where each individual was optimized by FIHC. The *perfect linkage* is a linkage that points true gene dependencies without any genes missing. Let us consider the 6-bit problem built from two order-3 trap functions. The first three genes refer to the first trap function, while the latter three genes refer to the second trap function. Note that these two blocks of genes are separable (i.e., to find the optimal value of the considered problem, we can optimize both blocks separately). Since the blocks are separable, then there are two groups of dependent genes (genes 1, 2, and 3 form the first group, while genes 4, 5, and 6 from the second one). The DSM that contains the perfect linkage for such a problem points that the most dependent genes of the first gene are genes 2 and 3 (the order is not important). Similarly, for the second gene, the most dependent genes should be 1 and 3, while for the fifth gene, the most dependent genes should be 4 and 6, and so on. Note that if the *perfect linkage* is supported to the evolutionary method, then (if the linkage is used properly), the evolutionary method is expected to solve the problem effectively [16]. Thus, estimating a chance for obtaining the *perfect linkage* is important for the field of evolutionary computation.

Consider a concatenation of r deceptive trap functions of order k. By a *block* we will understand the domain of a single deceptive function. Let us randomly (with equal probabilities) choose a population of s individuals, where individual $\overrightarrow{x} = [x_1, ..., x_n]$ is a vector of $n = rk$ binary variables. To each individual we apply FIHC. Denote by X_i a random variable with values in $\{0, 1\}$, whose probability distribution $(p_i(0), p_i(1))$ is obtained by taking frequencies of zeros and ones at position i, respectively. Fix any positions i, j and m, so that i and j lie in the same block, meaning that these genes are dependent, while m lies in a different block, i.e., m^{th} gene is independent of i^{th} and j^{th} genes. We want to estimate $P\big(I(X_i;X_j) > I(X_i;X_m)\big)$. Using a well-known equality $I(X;Y) = H(X) - H(X|Y)$ we get

$$I(X_i;X_j) - I(X_i;X_m) = H(X_i|X_m) - H(X_i|X_j).$$

After applying FIHC, all genes in a block have the same value, hence $H(X_i|X_j) = 0$. Thus,

$$\begin{aligned} &P\big(I(X_i;X_j) > I(X_i;X_m)\big) \\ &\quad = P\big(H(X_i|X_m) - H(X_i|X_j) > 0\big) \\ &\quad = P\big(H(X_i|X_m) > 0\big). \end{aligned} \tag{6}$$

Clearly, $H(X_i|X_m)$ is non-negative and $H(X_i|X_m) = 0$ only if all non-zero values of $p_{i,m}(a,b) = P(X_i = a, X_m = b)$ are equal to $p_m(b)$, i.e., if in each individual from the population the value of the m^{th} gene determines the value of the i^{th} gene. This gives the following possibilities for genes x_i and x_m: either all individuals in the population share only one combination of values, namely,

$$(\forall\overrightarrow{x}\ x_ix_m = 00) \quad \text{or} \quad (\forall\overrightarrow{x}\ x_ix_m = 01) \quad \text{or}$$
$$(\forall\overrightarrow{x}\ x_ix_m = 10) \quad \text{or} \quad (\forall\overrightarrow{x}\ x_ix_m = 11),$$

or exactly two combinations occur throughout the population, except for cases

$$\forall\overrightarrow{x}\ (x_ix_m = 00 \text{ or } x_ix_m = 10) \text{ and}$$
$$\forall\overrightarrow{x}\ (x_ix_m = 01 \text{ or } x_ix_m = 11)$$

Using the classical definition of probability we derive the probability that an individual has 0 at ith gene and either 0 or 1 at mth gene as equal to $\frac{(2^k-2)2^{(r-1)k}}{2^{rk}} = 1 - \frac{1}{2^{k-1}}$. Since the starting population is chosen by s independent drawings, the probability of choosing the population containing only such individuals (including populations with all individuals sharing only one combination of values, 00 or 01) is equal to $\left(1 - \frac{1}{2^{k-1}}\right)^s$. Similarly, the probability that an individual has 1 at ith gene is $\frac{1}{2^{k-1}}$ and the probability that all population has 1 at i^{th} gene is equal to $\frac{1}{2^{s(k-1)}}$. The probability that zeros or ones occur simultaneously at both genes is equal to $\frac{((2^k-2)^2+2^2)2^{(r-2)k}}{2^{rk}} = 1 - \frac{1}{2^{k-2}} + \frac{1}{2^{2k-3}}$. Finally, the probability that an individual has 0 at i, while 1 at m or vice versa is equal to $\frac{2\cdot 2\cdot(2^k-2)2^{(r-2)k}}{2^{rk}} = \frac{1}{2^{k-2}} - \frac{1}{2^{2k-3}}$. Note that each of the four situations when all individuals share only one combination of values falls into exactly two of considered cases. Therefore,

$$\begin{aligned} P\big(H(X_i|X_m) = 0\big) &= \left(1 - \frac{1}{2^{k-1}}\right)^s + \left(\frac{1}{2^{k-1}}\right)^s + \\ &+ \left(1 - \frac{1}{2^{k-2}} + \frac{1}{2^{2k-3}}\right)^s + \left(\frac{1}{2^{k-2}} - \frac{1}{2^{2k-3}}\right)^s - \frac{4}{2^{krs}}. \end{aligned}$$

Clearly,

$$P\big(H(X_i|X_m) = 0\big) \leq 4 \cdot \left(1 - \frac{1}{2^{k-2}} + \frac{1}{2^{2k-3}}\right)^s, \tag{7}$$

because the third summand majorizes all other terms.

The mutual information function $(i, j) \mapsto I(X_i;X_j)$, treated as a function on a Cartesian product $\{1, ..., n\} \times \{1, ..., n\}$, is constant on blocks. The probability that there is a triple i, j, m such that i and j belong to one block, while m belongs to another one and $I(X_i;X_j) \leq I(X_i;X_m)$ is smaller than $P\big(H(X_i|X_m) = 0\big)$ multiplied by the number of all blocks (which may contain i and j) and the number of all blocks decreased by one (which then may contain m). By (6) and (7), mutual information between i^{th} and j^{th} genes belonging to the same block majorizes mutual information between i^{th} and m^{th} genes lying in independent blocks for all triples of genes with probability greater than

$$1 - 4r(r-1)\left(1 - \frac{1}{2^{k-2}} + \frac{1}{2^{2k-3}}\right)^s.$$

To ensure that mutual information supports a correct linkage for all genes with probability q, it suffices to take a population of size

$$s > \log_a \frac{1-q}{4r(r-1)} \tag{8}$$

where $a = 1 - \frac{1}{2^{k-2}} + \frac{1}{2^{2k-3}}$. For instance, consider a concatenation of 100 deceptive trap functions of order 3, which gives a variety of 2^{300}

individuals. The above formula yields that to obtain correct linkage with probability 0.99 one needs $s > 32$. The above estimation was verified empirically, the results are presented in Section 6.2.

The obtained formula shows that the population sizes necessary to obtain a *perfect linkage* for trap function concatenations seem low. This statement is not necessarily true for other commonly used test problems, though by choosing a population in a sequence of independence drawings, we will always make the strong law of large numbers play on our side. However, even if the perfect linkage is hard to find by DSM-based linkage learning techniques, the imperfect linkage information may be sufficient to find the optimum as well. Therefore, in the next section, we define a linkage quality measure. With this measure, we can check what was the linkage quality at the end of the run when the optimal solution was found. Moreover, we can also generate an artificial linkage of the desired quality and check if such a linkage is sufficient to find the optimal result.

4 PROPOSED LINKAGE QUALITY MEASURE

In this section, we propose the quality measure for a linkage stored in DSM. We assume that if we possess a high-quality linkage, then for each gene x_i, the genes that are truly dependent on it shall correspond to higher entries in DSM. Based on it, we may create a ranking of dependencies between genes. For instance, let us assume that the first gene is truly dependent on genes 2, 3, 4, and 5, and is not dependent on any other genes. If linkage points at genes 5, 3, 4, and 2 as the most dependent on the first gene, then such a linkage is *perfect.* If linkage points out genes 3, 6, 4, 7, and 2, then such a linkage is imprecise but may still be useful.

Below, we define the *Fill* measure. Its value is defined for a particular position in a genotype, but one may as well understand it as a measure of the quality of the content of a single DSM row that refers to a particular gene.

Let $BlockSize(i)$ be the number of genes that depend on the i^{th} gene (i.e., the size of the block the i^{th} gene belongs to). Let $TrueDep(i, M, n)$ be the number of genes that are correctly pointed by the matrix M to be among n genes, most dependent on the i^{th} gene. We then set

$$Fill(i, M) = \frac{TrueDep(i, M, BlockSize(i))}{BlockSize(i)}, \qquad (9)$$

where i is the position in the genotype and M is a DSM for some population. Clearly, the best quality of linkage corresponds to $Fill(i, M)$ being equal to 1, while the worst quality corresponds to 0.

The proposed *Fill* measure is only applicable to problems that are additively separable. Proposing the linkage quality measures for problems with overlapping blocks is future work.

5 LINKAGE-IMPROVING FRAMEWORK

In this section, we define a framework that may be applied to any evolutionary method. The motivation behind our proposition is supporting an evolutionary method, a linkage that is not biased by the evolutionary process. Let us consider the following example. $\overrightarrow{x}$ is a solution to the problem that is a concatenation of 100 order-3 deceptive trap functions. The linkage information points that all genes from the first and second block and two genes (first and third) from the third block are dependent. The linkage does not allow a separation of the first block neither from the second block nor from the two genes from the third block. The first blocks of $\overrightarrow{x}$ are as follows $\overrightarrow{x}$ = [000 000 111...]. Consider a mixing operation, using linkage described above, in which $\overrightarrow{x}$ is a source and $\overrightarrow{x}$ = [111 111 000...] is a donor. After such an operation, the genotype of the donor will be $\overrightarrow{x_{mixed}}$ = [111 111 101...], because $fit(\overrightarrow{x_{mixed}}) > fit(\overrightarrow{x})$. Note that although the fitness of $\overrightarrow{x}$ has risen, the imprecise linkage information may be strengthened (for instance, if in most of the population, the first three blocks contain only '0's, which would be typical for deceptive trap functions). If so, then such a corrupted linkage may prevent an evolutionary method from finding an optimal solution.

The above example shows that for some of the problems, it may be favorable to obtain linkage without bias caused by an evolutionary process. Therefore, we propose a framework denoted as Unbiased Linkage Improver (ULI). ULI overview is given in Pseudocode 1.

Pseudocode 1 Unbiased Linkage Improver Overview

```
$ImprPop \leftarrow empty$; $ImprDSM \leftarrow random$;
$ImprFFE \leftarrow 0$; $OptFFE \leftarrow 0$;
while $\neg stopCondition$ do
    $StartFFE \leftarrow$ GetFFE();
    if $ImprFFE \leq OptFFE$ then
        $NewInd \leftarrow$ GetRandomGenotype();
        $NewInd \leftarrow$ FIHC($NewInd$);
        $GenesToFlip \leftarrow$ GenSize($NewInd$);
        while $GenesToFlip > 1$ do
            $IndBackup \leftarrow NewInd$;
            FlipGenes($NewInd, GenesToFlip$);
            $NewInd \leftarrow$ FIHC($NewInd$);
            if fitness($NewInd$) $\leq$ fitness($IndBackup$) then
                $NewInd \leftarrow IndBackup$;
                $GenesToFlip \leftarrow GenesToFlip/2$;
        $ImprPop \leftarrow ImprPop + NewInd$;
        $ImprDSM \leftarrow$ GetDSM($ImprPop$);
        $ImprFFE \leftarrow ImprFFE +$ GetFFE() $- StartFFE$;
    else
        RunOptIter();
        RunOptIterExternalLinkage($ImprDSM$);
        $OptFFE \leftarrow OptFFE +$ GetFFE() $- StartFFE$;
```

ULI is built from two parts: the evolutionary optimizer (eg., P3, LT-GOMEA, or DSMGA-2) and the separate part supposed to support additional linkage information (denoted as *linkage improver*). At each iteration, ULI executes a single iteration of linkage improver or two iterations of an optimizer. ULI chooses the one that has consumed a smaller number of FFE that far. A single iteration of linkage improver is as follows. First, a new individual is randomly generated and optimized by FIHC. Then, half of the genotype of a new individual is flipped, and the resulting genotype is optimized by FIHC. If this operation improves an individual, the operation is repeated. Otherwise, the genotype changes are reverted, and the number of genes to flip is decreased by half. The operation is performed until the number of genes to flip is higher than one. After such an optimization procedure, the new individual is added to the linkage improver population, and DSM is generated for this population.

If ULI decides to execute an optimizer, then it executes two iterations of it. The first iteration of an optimizer is executed in a regular way. However, in the second iteration, an optimizer is given a linkage gathered by the linkage improver (in the form of the DSM matrix generated on the base of linkage improver's population).

Despite its simplicity, ULI has some important advantages. It is parameter-less, and the individuals added to the linkage improver's population do not influence each other optimization processes. ULI was designed to improve linkage quality on the first dependency level. Thus, it may help to solve problems with overlapping blocks or a high level of the hierarchy. On the other hand, it may improve the effectiveness of evolutionary methods applied to solve problems for which the linkage is poorly detected by the DSM-using linkage learning techniques. Moreover, if an optimizer is capable of solving a particular overlapping problem, then it may still solve when used inside ULI.

6 THE RESULTS

In the research presented in this paper, we consider LT-GOMEA, P3, and psDSMGA-II because they are all the state-of-the-art methods in the field of single-objective optimization. Additionally, they are all parameter-less. All methods were coded in C++. LT-GOMEA source codes were published by its authors.[1]. For P3 and psDSMGA-II, we use the source code pointed in [4] and [6], respectively. All the source codes we have used and the detailed results of the runs are available at https://github.com/przewooz/LinkageQuality.

For the considered test problem set, and the considered methods, the stop condition based on FFE does not seem reliable [11, 12]. Therefore, we use the time-based stop condition. The time limit was set on 12 hours that should give all methods enough computation resources to reach the convergence. All experiments were executed on PowerEdge R7425 Dell server AMD Epyc 7601 2.2GHz 256GB RAM, with Windows 2019 Server installed. Each experiment was single-threaded, and the number of computation process was always the same and lower than the number of available processor cores. Such an execution procedure shall assure the fairness of the comparison. Unless stated otherwise, the number of experiments per test case was 30.

All considered methods employ a multi-population scheme and dynamically add new populations during the run (LT-GOMEA and psDSMGA-II also delete the populations during the run). In all methods, each population has a separate linkage. Therefore, when we report linkage quality for the whole method run, we report the highest linkage quality among all populations created by a method.

6.1 Test Problems

In this paper, we concentrate on measuring the linkage quality for additively separable problems. To show differences in linkage learning difficulties, we consider different types of deceptive functions. The first considered function is deceptive trap function defined in Section 3. We use order-3 and order-5 deceptive trap functions. Note that if FIHC is used, each block that refers to a single deceptive trap function may contain either only '0's or only '1's. This feature makes if relatively easy to find a perfect linkage with the use of DSM. Therefore, we also use deceptive step trap functions defined by formula (10). Deceptive step trap function is a modification of a deceptive trap function, which introduces plateaus of size s.

$$step\_trap(u) = \frac{(k-s) \mod s + dec(u)}{s} \tag{10}$$

We use order-3 and order-5 deceptive step trap functions $s = 2$. Note that such order-3 deceptive step trap function is 7-bit long (the same function was use in [4]). The order-5 deceptive step trap is 11-bit long.

Another type of deceptive functions considered in this paper are the bimodal deceptive function [3], defined as follows

$$bimodal\_trap(u) = \begin{cases} k/2 - |u - k/2| - 1 & , u \neq k \wedge u \neq 0 \\ k/2 & , u = k \vee u = 0 \end{cases} \tag{11}$$

Bimodal deceptive functions are significantly different from deceptive step traps because they have two optima and $\binom{k}{k/2}$ local optima. As we show in the results section, it is harder to detect a high-quality linkage with the use of DSM when a problem has many suboptima. We also use bimodal noised deceptive functions [7, 17]. The noised bimodal functions increase the number of suboptima and make these suboptima different concerning the unitation. The values for order-10 function are in Table 1. Both bimodal deceptive function types are considered in the order-10 version.

Table 1: The values of bimodal noised deceptive function of order-10

Unitation	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Function value	4	0	2	1	3	2	3	1	2	0	4

The problems considered in this paper are built from deceptive functions concatenations. Except for mixed bimodal functions concatenation, all problems use only one function type. In mixed bimodal functions concatenation, half of the problem is built from the bimodal functions, and the other half is built from the bimodal noised functions.

6.2 The Resistance to Low Linkage Quality

In this paper, we propose a measure to determine the quality of the linkage. All considered methods use linkage learning. Therefore, it is important to check how high linkage quality is a must to find an optimal solution. To this end, we introduce an artificial linkage into the competing methods. The procedure of the artificial linkage creation creates an artificial DSM and is as follows. To generate a perfect linkage, for all DSM entries that represent a true gene dependencies for a given problem, the value of each entry is randomly chosen from range [0.6, 1.0], while the value of the entries that refer to gene pairs without true dependency is randomly chosen from range [0, 0.4]. Note that for DSM generated in such a way, for the additively separable problems, the value of the fill measure will be 1 (i.e., optimal). If we wish to generate the linkage of the lower quality, for instance, we wish to obtain a DSM for which the expected average fill value will be 0.6, then for each entry that represents a true dependency between two genes, we perform a

[1]https://homepages.cwi.nl/~bosman/source_code.php, downloaded at 2018.08.30

Table 2: Artificial linkage quality sufficient to solve the problem*

	Bimodal		Bim.mix		Bim.nois.		Dec.3		St.Dec.3		Dec.5		St.Dec.5	
	400	1200	400	1200	400	1200	400	1200	400	1200	400	1200	400	1200
LT-GOMEA	0.5	0.6	0.5	0.6	0.4	0.5	0.9	1	0.7	0.8	0.8	0.9	0.6	0.7
P3	0.5	0.6	0.5	0.6	0.5	0.5	0.9	1	0.7	0.8	0.8	0.9	0.6	N/A

Results obtained for 30 independent runs

random check with a probability 0.6. If the check is passed, then the value of the entry is generated in a regular way. Otherwise, the value of the entry is randomly chosen from the range [0, 0.4].

Surprisingly, when psDSMGA-II was supported with a perfect linkage, it was unable to find an optimal solution for 1200-bit problems. Such an observation was unexpected. The detailed results analysis has shown that when psDSMGA-II is supported a perfect linkage right from the beginning of the run, then it tends to delete all populations except the currently largest one. Such behavior leads to increasing the size of the largest population almost at every method iteration. The reason for this phenomenon is as follows. When psDSMGA-II is supported with a perfect linkage right at the beginning of the run, any population it creates quickly converges. Thus, larger populations can dominate smaller ones as they execute the backpropagation frequently. Each population is close to finding an optimal solution, but it requires more iterations to improve its best individuals. In effect, when psDSMGA-II is supported with a perfect linkage at the beginning of the run, its behavior is close to continuously doubling the size of the single population that is initialized randomly. The size of this population quickly rises to hundreds of thousands, preventing the method from converging to the optimal solution. In a regular psDSMGA-II run, the above phenomenon does not take place because every population must first evolve and improve the quality of its DSM before it can dominate the smaller populations. Therefore, an important future work direction is to find a population-sizing schema that would be more suitable for DSMGA-II. Due to the above phenomenon, the results for psDSMGA-II will avoid in this subsection.

In Table 2, we present the lowest linkage quality for which LT-GOMEA and P3 were able to find an optimal solution. The test procedure was as follows. Each method, for each problem, was first supported with an optimal linkage (of average fill quality 1). Then, the artificial linkage quality was lowered by 0.1 until a method failed to find an optimal solution in all of the 30 runs. The obtained results show that for 400-bit, the linkage quality sufficient to find an optimal solution is slightly lower than for 1200-bit problems. The results are the same or close for both methods except for 1200-bit order-5 deceptive step functions concatenation. For this problem, when P3 was supported with a perfect linkage, it was able to find an optimal solution only in 40% of the runs. This is much more than in the case of regular P3 run in which it was only 3%, but still, this result is unexpected. The reason for the fail was similar as in psDSMGA-II case - P3 was creating a very tall pyramid containing many individuals. Such a population structure was limiting the convergence and preventing P3 from finding an optimal solution. It seems that an important future work direction is to propose a P3 modification that would limit its pyramid size and introduce a selection pressure.

The results presented in Table 2 show that the highest quality linkage is required to solve problems built from deceptive trap functions. Such an observation may be surprising because this kind of problem seems to be the easiest to solve for all considered methods. Based on the calculations presented in Section 3, for the DSM-using linkage learning techniques, it seems easy to find a linkage of high quality. Therefore, for the method employing such techniques, it seems easy to solve the concatenations of deceptive trap functions. However, it seems easy because it is easy to find a high-quality linkage, not because the problem itself is easy to solve.

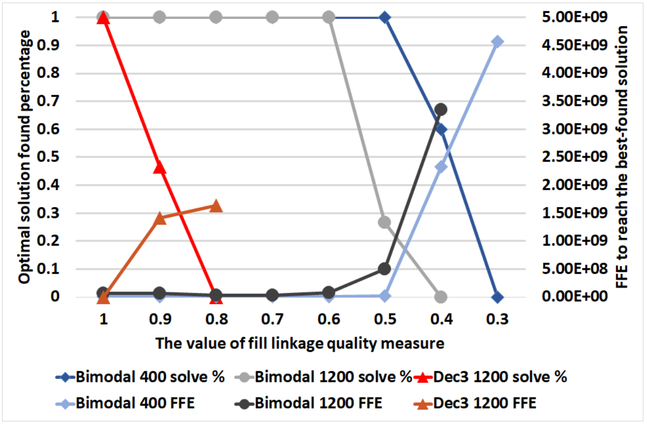

Figure 1: The influence of linkage quality on the percentage of optimal solutions found and the median amount of FFE necessary to reach the best-found solution by LT-GOMEA on the chosen optimization problems

In Figure 1, we present the dependency between the quality of linkage and the percentage of optimal solutions found and the median FFE necessary to obtain the best-found solution in a run. As long as the linkage quality is high enough, an optimal solution is found in all runs at a very low cost. However, when the linkage quality drops down to some critical value (different for all three problems presented in Figure 1), then the percentage of successful runs drops down immediately as well. At the same time, the FFE necessary to find the best results in a run (no matter optimal or not) rises significantly as well.

6.3 Main Results

In this section, we empirically verify the accuracy of the estimation presented in Section 3, and present the comparison between the competing methods in their standard versions and when ULI framework is used. Finally, we support some additional data to show the influence of linkage quality on considered evolutionary methods.

Table 3: Population* necessary to get perfect linkage**

	ULI			FIHC			Est.
	med	**min**	**max**	**med**	**min**	**max**	
Dec.3 399	19	15	25	21	15	28	34
Dec.3 1200	22	14	32	23	14	37	38
Dec.5 400	34	14	84	38	14	113	118
Dec.5 1200	42	14	99	49	14	121	136

**Results obtained for 100 independent runs*
***Estimation made for probability 0.99*

Table 3 presents the population sizes that were necessary to obtain a perfect linkage for given deceptive trap concatenations. The population sizes are given for two cases. In the first individuals were optimized by a procedure proposed in ULI, while in the second one, each individual was only processed by FIHC. Finally, in the last column, we report the value of the estimation proposed in formula 8. The calculation is made for probability 0.99. Note that the estimation is made for the populations where individuals were only updated by FIHC.

As presented in Table 3the maximum population size obtained in 100 runs is slightly lower than the estimation. Thus, the estimation proposed in formula 8 seems reliable. Note that the population size values obtained when ULI procedure was employed are lower than in the case of using only FIHC. We have verified the medians equality with Sign Test, the p-value for the null hypothesis that the medians are equal was lower than 0.003 for four cases. Thus, we may state that the differences are statistically significant. Such a result is expected because the ULI procedure is more sophisticated than FIHC, and the optimization results of its work shall be of higher quality than in the case of FIHC.

In Table 4, we present the summarized results for the highest considered problem length (1200 bits). Except of the percentage of the optimal solution found, we present the resource cost of finding the best solution (for FFE and computation time) and the linkage quality at the end of the run. For both deceptive trap functions concatenation ULI slightly delays finding the optimal result (for both considered measures – FFE and computation time). Such a result is expected because, as shown in Section 3, for deceptive trap functions, DSM can quickly find the perfect linkage. Thus, the linkage improver only slows down the computation. For deceptive step functions, the influence of ULI does not seem significant – the percentage of optimal solutions found and the resources necessary to find them remain similar. Note that for deceptive step traps, the linkage quality gathered by all considered optimizers is relatively high or even close to perfect (LT-GOMEA). Thus, if a method was unable to find an optimal solution, the reason for the failure is in the method procedure rather than in missing the linkage of reasonably high quality.

The significant influence of ULI is the most visible for all problems built from bimodal functions. For LT-GOMEA applied to bimodal functions concatenation, the FFE and computation time costs of finding the optimal solution are significantly reduced (up to about 8 times). Moreover, ULI-LT-GOMEA successfully finds an optimal solution for bimodal mixed and bimodal noised concatenations, while LT-GOMEA does not. Note that for all these problems, the quality of linkage found by ULI-LT-GOMEA is close to perfect. Surprisingly, when ULI is used, for all methods, the linkage supported by the linkage improver is of lower quality than the linkage found by an optimizer, but the linkage found by an optimizer using ULI is higher (sometimes significantly) than for the same methods that do not use ULI. The reasonable explanation of this fact is as follows. Most probably, Linkage the Improver can successfully mark some of the blocks. When an optimizer uses such an external linkage can successfully process these correctly marked building blocks. The result of this situation is twofold. The fitness of the population rises, and the appropriate dependencies are strengthened in the DSM matrix leading to a fast increase of the overall DSM quality of the optimizer.

The influence of ULI on P3 is even more significant than in the case of LT-GOMEA. ULI-P3 can find a significantly higher percentage of optimal solutions for bimodal functions than P3. Surprisingly, for the bimodal deceptive functions concatenation the quality of linkage found by P3 is close to optimal. The situation is similar to the problem built from step deceptive functions of order 5. In both cases P3 has found a high-quality linkage but was unable to find the optimal result in almost all of the runs despite the fact it was given a high amount of computation resources. The reason for this fact is the P3 algorithm itself. During the run, at each P3 iteration, the pyramid grows. In consequence, each P3 iteration is more expensive, and additionally, the selection pressure is low – the individuals of the higher quality do not replace those of the lower quality but coexist with them in the same pyramid. Therefore, if P3 will not find the high-quality linkage in the early stages of the run, and if it will not converge quickly, then finding an optimal solution may be hardly likely for this method. Such a reasoning is confirmed by both – results reported in other papers and results reported by ULI-P3. In [7, 17] the improved P3 versions are proposed. These modified P3s, under some circumstances, add the current best-found individual to the lowest level of the pyramid. Thanks to this, the convergence is improved. A similar effect (although achieved differently) takes place for ULI-P3. In ULI-P3, the convergence of P3 is improved by a quick finding linkage of reasonably high quality. Note that ULI-P3 successfully solves bimodal deceptive functions concatenation with a lower linkage quality than the linkage quality gathered by P3 (note that the P3 success rate on this problem was only 3%).

Unfortunately, the influence of ULI on psDSMGA-II was either neutral or negative. In Section 3, we have described a phenomenon that if psDSMGA-II was supported with a perfect linkage at the beginning of the run, then it was unable to find an optimal solution. Similarly to the psDSMGA-II behavior described in Section 3, ULI-psDSMGA-II tends to quickly create large populations (up to hundreds of thousands of individuals). Therefore, it is unable to find the optimal solution. Since ULI is not beneficial for psDSMGA-II, this method will be avoided in the rest of this section.

It is important to check if the size of the population with the highest quality is somehow influenced by ULI. These results are reported in Table 5. Except for the deceptive trap functions, for all problems, the best linkage is found for the significantly lower population sizes when ULI is used. For some of the problems, this linkage is sufficient to solve the problem. Moreover, based on the results presented in Tables 4 and 5, for ULI-LT-GOMEA and for

Table 4: Main results for the considered problems with n=1200

	LT-GOMEA					psDSMGA-II					P3				
Problem	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill
Dec.3	1.00	5.5E+5	28	0.38	N/A	1.00	1.1E+5	338	0.52	N/A	1.00	6.0E+4	19	0.99	N/A
Dec.5	1.00	2.5E+6	63	0.98	N/A	1.00	2.0E+5	341	0.69	N/A	1.00	2.5E+5	66	0.98	N/A
St.Dec.3	1.00	5.3E+7	835	1.00	N/A	1.00	1.2E+7	7052	0.41	N/A	1.00	2.3E+7	7168	0.90	N/A
St.Dec.5	1.00	3.1E+8	10389	0.97	N/A	0.97	1.8E+8	10972	0.49	N/A	0.03	1.3E+8	41992	0.79	N/A
Bimodal	1.00	5.0E+8	8075	0.99	N/A	0.77	1.1E+7	968	0.50	N/A	0.03	2.3E+8	42298	0.98	N/A
Bim.mix	0.03	1.5E+9	42773	0.77	N/A	0.93	4.3E+7	3516	0.43	N/A	0.03	4.6E+8	42581	0.50	N/A
Bim.nois.	0.10	1.5E+9	43674	0.89	N/A	0.97	1.4E+8	9264	0.30	N/A	0.03	3.9E+8	37715	0.13	N/A

	ULI-LT-GOMEA					ULI-psDSMGA-II					ULI-P3				
Problem	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill	Sol ved [%]	FFE	Time [s]	Opt Fill	LI Fill
Dec.3	1.00	1.1E+6	35	0.73	0.77	1.00	3.6E+5	396	0.11	0.08	1.00	1.4E+5	30	0.97	0.02
Dec.5	1.00	3.6E+6	99	0.68	0.94	1.00	5.6E+5	352	0.16	0.13	1.00	5.2E+5	135	0.97	0.11
St.Dec.3	1.00	7.8E+7	1361	1.00	0.01	1.00	5.9E+7	7598	0.34	0.01	1.00	1.4E+8	10008	0.92	0.02
St.Dec.5	1.00	4.9E+8	7569	1.00	0.01	0.80	3.7E+8	11620	0.24	0.02	0.03	7.0E+8	42338	0.77	0.02
Bimodal	1.00	6.7E+7	1880	0.99	0.74	0.43	8.0E+7	3924	0.98	0.93	1.00	5.2E+7	4458	0.85	0.68
Bim.mix	1.00	6.8E+8	16280	0.99	0.70	0.87	1.9E+8	8095	0.56	0.54	0.52	5.3E+8	41496	0.74	0.64
Bim.nois	1.00	1.3E+9	34036	0.99	0.48	0.63	8.6E+8	23158	0.33	0.35	0.00	7.8E+8	42744	0.42	0.30

Table 5: The median population size with the highest DSM quality for the considered problems with n=1200

	LT-GOMEA		P3	
	Standard	ULI	Standard	ULI
Dec.3	64	96	21	24
Dec.5	256	256	89	88
St.Dec.3	4096	2048	679	642
St.Dec.5	16384	8192	3602	2285
Bimodal	16384	1024	8158	1515
Bim.mix	32768	4096	7981	209
Bim.nois.	32768	8192	2209	61

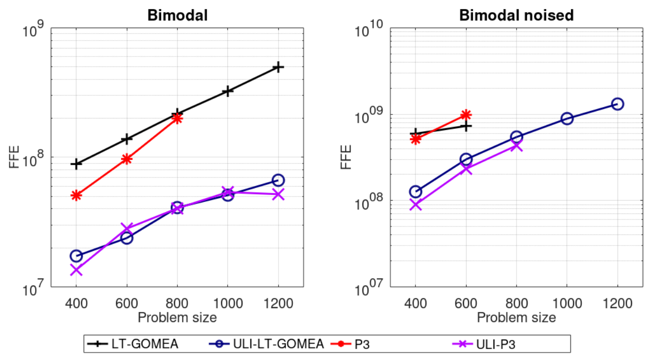

Figure 2: Scalability analysis for median FFE necessary to find an optimal solution for bimodal-based problems

mixed bimodal and bimodal noised concatenations, the linkage quality found by these smaller populations is higher than the linkage quality found by significantly larger populations in LT-GOMEA.

In Figure 2, we present the scalability analysis for the two chosen bimodal problems. For these problems, it is clear that if LT-GOMEA or P3 uses the ULI framework, then it scales significantly better.

7 CONCLUSION

In this paper, we have proposed a linkage quality measure for partially additively separable problems. Using the proposed measure, we have investigated the dependency between the linkage quality and the effectiveness of the evolutionary methods. Finally, we have proposed the ULI framework that allows improving the linkage quality. The results analysis indicates many interesting phenomena. The main future work directions are as follows. The linkage quality measures shall be proposed for problems with overlapping blocks. A population-sizing schema that is more suitable for DSMGA-II should be proposed. Another interesting future research direction is the modification of P3 that would improve its convergence. Finally, the population size that is sufficient to obtain a perfect linkage shall be analytically found for other types of deceptive functions.

ACKNOWLEDGMENTS

This work of Michal Przewozniczek and Marcin Komarnicki was supported by the Polish National Science Centre (NCN) under Grant 2015/19/D/ST6/03115 and the statutory funds of the Department of Computational Intelligence, Wroclaw University of Science and Technology. The work of Bartosz Frej was supported by the statutory funds of the Faculty of Pure and Applied Mathematics, Wroclaw University of Science and Technology.

REFERENCES

[1] 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.

[2] 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 (GECCO '17)*. ACM, 745–752.

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

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

[5] 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 (GECCO'99)*. 258–265.

[6] 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.

[7] 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.

[8] Marcin M. Komarnicki and Michal W. Przewozniczek. 2019. Parameter-Less, Population-Sizing DSMGA-II. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 289–290.

[9] 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.

[10] 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. https://doi.org/10.1109/TEVC.2017.2694221

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

[12] 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.

[13] Michal Witold Przewozniczek and Marcin Michal Komarnicki. 2020. Empirical Linkage Learning. *IEEE Transactions on Evolutionary Computation* (2020), (in press).

[14] 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.

[15] 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.

[16] 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

[17] 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 (GECCO '19)*. ACM, New York, NY, USA, 312–313. https://doi.org/10.1145/3319619.3322052