Comparative Mixing for DSMGA-II

Marcin M. Komarnicki
Department of Computational Intelligence
Wroclaw University of Science and Technology
Wroclaw, Poland
marcin.komarnicki@pwr.edu.pl

Michal W. Przewozniczek
Department of Computational Intelligence
Wroclaw University of Science and Technology
Wroclaw, Poland
michal.przewozniczek@pwr.edu.pl

Tomasz M. Durda
Department of Computational Intelligence
Wroclaw University of Science and Technology
Wroclaw, Poland
tomekdur@wp.pl

ABSTRACT

Dependency Structure Matrix Genetic Algorithm-II (DSMGA-II) is a recently proposed optimization method that builds the linkage model on the base of the Dependency Structure Matrix (DSM). This model is used during the Optimal Mixing (OM) operators, such as the Restricted Mixing (RM) and the Back Mixing (BM). DSMGA-II was shown to solve theoretical and real-world optimization problems effectively. In this paper, we show that the effectiveness of DSMGA-II and its improved version, namely Two-edge Dependency Structure Matrix Genetic Algorithm-II (DSMGA-IIe), is relatively low for NK-landscape problems. Thus, we propose the Comparative Mixing (CM) operator that extends the RM operator. The CM operator modifies the linkage information obtained from the DSM-based linkage model by comparing the receiver individual with a randomly selected member of the population. Such modification enables DSMGA-II to solve NK-landscape problems effectively and does not limit DSMGA-II performance on most problems for which it was already shown effective.

CCS CONCEPTS

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

KEYWORDS

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

ACM Reference Format:
Marcin M. Komarnicki, Michal W. Przewozniczek, and Tomasz M. Durda. 2020. Comparative Mixing for DSMGA-II. In *Genetic and Evolutionary Computation Conference (GECCO '20), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3377930.3390223

1 INTRODUCTION

Many real-world optimization problems have overlapping structure, i.e., they cannot be divided into separable blocks. One of the recent research directions is developing the optimization methods that use DSM to discover dependency between genes or block of genes accurately. One of the recent propositions is Dependency Structure Matrix-II (DSMGA-II), which was proposed by Hsu and Yu in 2015 [9]. DSMGA-II was show ineffective in solving NK-landscape problems [11, 12, 20]. One of the features of NK-landscape problems is a heavily overlapping structure. Therefore, the objective of this paper is to propose a new DSMGA-II version that shall improve DSMGA-II effectiveness on NK-landscapes problems and at least preserve DSMGA-II effectiveness on other problems.

The rest of this paper is organized as follows. In the next subsection, we present the related optimization methods. The third section gives a detailed description of the proposed Comparative Mixing operator. Section 4 presents the obtained results, including scalability and statistical analysis of the significance of the results. Finally, the last section concludes the paper.

2 RELATED OPTIMIZATION METHODS

In this section, we introduce two optimization methods, namely Linkage Tree Genetic Algorithm (LTGA) and Parameter-less Population Pyramid (P3). Both are state-of-the-art propositions and use DSM to improve their effectiveness.

LTGA [16, 18] is one of the earliest method propositions that use Dependency Structure Matrix (DSM) to discover dependencies between genes. LTGA, which is an example of a population-based method, builds the Linkage Tree on the base of DSM using the hierarchical clustering algorithm. Nodes of the Linkage Tree are sets of gene indexes that are found to be dependent on one another. These nodes are usually called masks. During the single LTGA iteration, all masks excluding root are used while executing the Optimal Mixing (OM) [17] operation. Since LTGA maintains a population of the fixed number of individuals, it is not a parameter-less method. Due to the difficulty of setting the proper population size, LTGA was extended by the population sizing scheme [1]. The extended version of LTGA (psLTGA) implements mainly the general population sizing schema proposed in [8].

Another example of the evolutionary method that uses the Linkage Tree is P3 [5–7, 10, 11, 20]. Undoubtedly, the strength of P3 is the fact that it is a parameter-less optimizer. It means that P3 may be applied to any optimization problem without any *a priori* tuning procedure. P3 is not a standard population-based method. In P3, individuals are stored in the form of a pyramid, where each pyramid level is a subpopulation containing unique individuals, i.e., if one individual is a member of a specific level, it cannot be included in other subpopulations. This structure resembles a pyramid because higher levels should contain better-fitted individuals, which should not be as many as less fitted ones. The P3 iteration

GECCO '20, July 8–12, 2020, Cancún, Mexico

ACM ISBN 978-1-4503-7128-5/20/07…$15.00
https://doi.org/10.1145/3377930.3390223

may be divided into two steps. At the beginning of each iteration, a new individual is created on a random basis and then optimized using the local search algorithm, namely First Improvement Hill Climber [5]. During the other step of the P3 iteration, the newly created individual climbs a pyramid. While performing this operation, the individual is mixed using the OM operator with all individuals on each pyramid level, taking only the Linkage Tree related to a specific subpopulation into consideration.

3 PROPOSED METHOD

This section firstly introduces DSMGA-II and DSMGA-IIe. Then we describe the Comparative Mixing operator and the modification of the Back Mixing operation. At the end of the section, the population sizing scheme is presented.

3.1 Dependency Structure Matrix Genetic Algorithm-II

Dependency Structure Matrix Genetic Algorithm-II (DSMGA-II) [9] is a recent proposition of an evolutionary method that uses DSM to discover dependencies between genes. DSMGA-II is a population-based method whose individuals are randomly initialized and optimized using the bit-flipping greedy hill-climbing (GHC) afterward. DSMGA-II introduces two new OM operators that are Restricted Mixing (RM) and Back Mixing (BM). The general DSMGA-II procedure is adopted from [9] and shown in Algorithm 1.

Algorithm 1 The general DSMGA-II procedure

```
P : population, S : selected population,
s : selection pressure, R : constant,
DSM : dependency structure matrix, M : mask
input: l : problem size, p : population size
output: best individual in P
P ← PopulationInitialization(l, p)
P ← GHC(P)
while ¬ShouldTerminate do
    S ← TournamentSelection(P, s)
    DSM ← UpdateMatrix(S)
    for k ← 1 to R do
        I ← random permutation from 1 to p
        for i ∈ I do
            (P_i, M) ← RestrictedMixing(P_i)
            if M ≠ ∅ then
                P ← BackMixing(P_i, M)
return best inidividual in P
```

Due to building the linkage model and utilizing it during the OM operators subsequently, DSMGA-II is capable of effectively solving many optimization problems [2, 9, 11]. The linkage model is built on the base of DSM. DSM is a square matrix that is used by DSMGA-II to store dependencies between genes. The value of the single DSM entry $d_{i,j} \in R$ indicates how strong the i^{th} and j^{th} gene are dependent. The greater the value is, they are found to be less independent. The relationship between each pair of genes is measured by mutual information [13]. Mutual information is an example of a pairwise dependency measure and is defined as follows:

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

where X and Y denote random variables. If X and Y are independent, then the $I(X;Y)$ value is minimum. Regarding DSMGA-II, formula (1) should consider dependencies between genes. Thus, it may be reformulated as follows:

$$I(G_i;G_j) = \sum_{g_i \in G_i} \sum_{g_j \in G_j} p_{i,j}(g_i,g_j) \ln \frac{p_{i,j}(g_i,g_j)}{p_i(g_i)p_j(g_j)} \tag{2}$$

where i^{th} and j^{th} gene are denoted by G_i and G_j, respectively. The probability that in a whole population the g_i value is assigned to the i^{th} gene is indicated by $p_i(g_i)$. The value of the joint probability $p_{i,j}(g_i,g_j)$ takes into account that the i^{th} gene has the g_i value and the j^{th} gene has the g_j value simultaneously. Notice that it is possible to assign 0 to $p_i(g_i)$, $p_j(g_j)$, or $p_{i,j}(g_i,g_j)$. Then, the value of $\ln \frac{p_{i,j}(g_i,g_j)}{p_i(g_i)p_j(g_j)}$ is assumed to be equal 0 as well. DSMGA-II is mainly designed to deal with binary optimization problems. Therefore, the $I(G_i,G_j)$ value can be calculated using the following formula:

$$\begin{aligned} I(G_i;G_j) = {} & p_{i,j}(0,0) \ln \frac{p_{i,j}(0,0)}{p_i(0)p_j(0)} + p_{i,j}(0,1) \ln \frac{p_{i,j}(0,1)}{p_i(0)p_j(1)} \\ & + p_{i,j}(1,0) \ln \frac{p_{i,j}(1,0)}{p_i(1)p_j(0)} + p_{i,j}(1,1) \ln \frac{p_{i,j}(1,1)}{p_i(1)p_j(1)} \end{aligned} \tag{3}$$

The main goal of constructing DSM is to apply a clustering algorithm over it. The output of the clustering algorithm should be groups of genes that are highly dependent on one another. These groups are commonly called Building Blocks (BBs). DSMGA-II builds the Incremental Linkage Set (ILS) to meet this objective. Because DSM may be considered as a graph, ILS is iteratively constructed by finding the Approximation Maximum-Weight Connected Subgraph (AMWCS). At the end of the ILS construction procedure (see Algorithm 2), ILS consists of $\lfloor l/2 \rfloor$ masks of length from 1 to $\lfloor l/2 \rfloor$, where l is the considered optimization problem size, whereas masks are sequences of gene indexes and are believed to be BBs. Let us consider DSM as a graph whose vertices are just all gene indexes and each pair of vertices is linked. Moreover, each edge has an assigned value called a linkage that is equal to mutual information. The first mask in ILS is created by randomly selecting one of the vertices. The second mask is constructed by adding to the end of the first mask, the gene index with the max linkage that has not been already chosen in the first step. Generally, the next mask in ILS is created by adding to the end of the previously constructed mask the vertex with the max linkage that has not been already added during any of the previous steps. The vertex with the max linkage is selected regarding the following formula:

$$v^* = \arg\max_{v \in V} \sum_{m \in M} I(G_v;G_m) \tag{4}$$

where V consists of vertices that are not members of any of already constructed masks and M is the previous mask. Let us consider DSM from Table 1. For instance, when the 3^{rd} vertex is a randomly selected gene index, the following ILS will be constructed: $<\{3\}, \{3,4\}, \{3,4,1\}>$.

Table 1: DSM to demonstrate the ILS creation procedure

Gene index	1	2	3	4	5	6
1	X	0.001	0.063	0.112	0.275	0.112
2	0.001	X	0.002	0.033	0.001	0.164
3	0.063	0.002	X	0.459	0.063	0.033
4	0.112	0.033	0.459	X	0.112	0.089
5	0.275	0.001	0.063	0.112	X	0.112
6	0.112	0.164	0.033	0.089	0.112	X

Algorithm 2 The ILS construction procedure

```
ILS : incremental linkage set, V : candidate vertices set,
M : mask, l : problem size
output: ILS : incremental linkage set
ILS ← ∅
M ← []
V ← {1, 2, ..., l}
v ← a random vertice in V
for i ← 1 to ⌊l/2⌋ do
    add v to M
    ILS_i ← M
    V ← V \ {v}
    v* ← the vertex in V with the max linkage
    v ← v*
```

Obtained ILS is then employed during RM and BM operators. Both are examples of the OM operator. The main idea of this operation is to change only one individual regarding a given mask and check the difference between the fitness value before and after the change. If the fitness value decreases, then the change has to be reverted. Otherwise, when the fitness value remains the same or is greater after the genotype modification, then the change is retained. In the RM operator, a selected individual called the receiver is being modified by one of the possible ILSs that is chosen randomly. Masks, which are members of the selected ILS, are processed sequentially. Values of genes marked by a considered mask are flipped. After that, new values are checked if there is at least one individual in a population that has the same values of genes on positions marked by the mask. If the result of this operation, namely supply check, is positive and the fitness value of the receiver individual is equal or greater than before the modification, then the receiver individual remains modified and the RM procedure terminates. Otherwise, the bit-flipping operations are reversed and the next mask is being processed. If the receiver individual after the modification is accepted, then the BM operator is performed and the receiver individual becomes the donor individual for other individuals in a population like in the typical OM operator [17]. The mask used during the mixing is the last considered mask while performing the RM operation. If during the BM operation there is no strict improvement in the fitness value of any individual in a population, all genotype modifications that result in the same fitness values are accepted. Otherwise, only strict improvement changes remain.

3.2 Two-edge Dependency Structure Matrix Genetic Algorithm-II

In the original version of DSMGA-II, the process of finding the AMWCS uses DSM, considered as a graph, that has exactly one link between each pair of vertices, which are simply gene indexes. This approach allows for general dependencies between genes, i.e., mutual information is calculated for all possible values of genes. The two-edge linkage model proposed in Two-edge Dependency Structure Matrix Genetic Algorithm-II (DSMGA-IIe) [2] takes dependencies between complementary patterns into account while finding AMWCS.

In this version of building ILS, the values of the genes of the receiver individual are taken into consideration. Thus, DSM is divided into $\text{DSM}_{(00\cup11)}$ and $\text{DSM}_{(01\cup10)}$. The values of their entries are calculated using formula (5) and formula (6), respectively.

$$I_{(00\cup11)}(G_i, G_j) = p_{i,j}(0,0)\ln\frac{p_{i,j}(0,0)}{p_i(0)p_j(0)} + p_{i,j}(1,1)\ln\frac{p_{i,j}(1,1)}{p_i(1)p_j(1)} \tag{5}$$

$$I_{(01\cup10)}(G_i, G_j) = p_{i,j}(0,1)\ln\frac{p_{i,j}(0,1)}{p_i(0)p_j(1)} + p_{i,j}(1,0)\ln\frac{p_{i,j}(1,0)}{p_i(1)p_j(0)} \tag{6}$$

Note that $I(G_i, G_j) = I_{(00\cup11)}(G_i, G_j) + I_{(01\cup10)}(G_i, G_j)$, so DSM = $\text{DSM}_{(00\cup11)}$+$\text{DSM}_{(01\cup10)}$ as well. Thus, a graph used to find AMWCS has two links between each pair of gene indexes. The value of the first edge is taken from $\text{DSM}_{(00\cup11)}$, whereas the other edge value is obtained from $\text{DSM}_{(01\cup10)}$. Nevertheless, while finding the vertex with the max linkage using formula (7), only one link is considered depending on the receiver individual that takes part in the RM operation.

$$v^*_{two\_edge} = \arg\max_{v\in V}\sum_{m\in M} L(G_v, G_m) \tag{7}$$

$$L(G_i, G_j) = \begin{cases} I_{(00\cup11)}(G_i, G_j), R_i = 0 \wedge R_j = 0 \vee R_i = 1 \wedge R_j = 1 \\ I_{(01\cup10)}(G_i, G_j), R_i = 0 \wedge R_j = 1 \vee R_i = 1 \wedge R_j = 0 \end{cases} \tag{8}$$

where R_i is the i^{th} gene value of the receiver individual.

3.3 Comparative Mixing

In the original version of DSMGA-II, all possible ILSs in a single method iteration are the same for all individuals in a population. It leads to obtaining poor effectiveness while solving NK-landscape problems [11, 12, 20], which are heavily overlapping problems. Due to proposing the new way of creating ILS depending on the individual, the recent DSMGA-II modification, i.e., DSMGA-IIe improves this effectiveness. Unfortunately, in many cases, its process of constructing the linkage model has to start from scratch because of the population diversity. This operation is time-consuming. Therefore, we propose the Comparative Mixing (CM) operator that extends the RM operator by filtering given ILS on the base of a randomly selected member from a population. Thus, during the single method iteration, distinct individuals may have different possible ILSs to use. We named the new version of DSMGA-II that uses the CM

operator as Dependency Structure Matrix Genetic Algorithm-II with the Comparative Mixing operator (DSMGA-IIc). The general DSMGA-IIc procedure is presented in Algorithm 3. The only difference in comparison to Algorithm 1 is applying the CM operator instead of the RM operator.

Algorithm 3 The general DSMGA-IIc procedure

```
1:  P : population, S : selected population,
2:  s : selection pressure, R : constant,
3:  DSM : dependency structure matrix, M : mask
4:  input: l : problem size, p : population size
5:  output: best individual in P
6:  P ← PopulationInitialization(l, p)
7:  P ← GHC(P)
8:  while ¬ShouldTerminate do
9:      S ← TournamentSelection(P, s)
10:     DSM ← UpdateMatrix(S)
11:     for k ← 1 to R do
12:         I ← random permutation from 1 to p
13:         for i ∈ I do
14:             (P_i, M) ← ComparativeMixing(P_i)
15:             if M ≠ ∅ then
16:                 P ← BackMixing(P_i, M)
17: return best inidividual in P
```

At the beginning of the CM procedure regarding Algorithm 5, ILS must be constructed. This operation (Algorithm 4) resembles filtering ILS obtained in the DSMGA-II way (Algorithm 2) and results in at most $\lfloor l/2 \rfloor$ ILSs, where l denotes a given problem size. The rest of the CM procedure is almost the same as in the RM operator. The only difference is that the length of ILS may be smaller than $\lfloor l/2 \rfloor$.

Algorithm 4 The ILS construction procedure for the CM operator

```
1:  ILS : incremental linkage set, V : candidate vertices set,
2:  M : mask, l : problem size, R : receiver,
3:  P : population, C : comparing solution
4:  input: R : receiver
5:  output: ILS : incremental linkage set
6:  ILS ← ∅
7:  M ← []
8:  V ← {1, 2, ..., l}
9:  v_c ← a random vertice in V
10: add v_c to M
11: ILS_1 ← M
12: V ← V \ {v_c}
13: C ← a random individual in P different than R
14: i ← 2
15: while i ≤ ⌊l/2⌋ and V ≠ ∅ do
16:     v* ← the vertex in V with the max linkage
17:     if C_{v*} = C_{v_c} ⇔ R_{v*} = R_{v_c} then
18:         add v* to M
19:         ILS_i ← M
20:         i ← i + 1
21:     V ← V \ {v*}
```

Algorithm 5 The CM procedure

```
1:  ILS : incremental linkage set, M : mask, R : receiver,
2:  f : evaluation function, P : population,
3:  l : problem size, T : trial solution
4:  R_M : pattern of R extracted by M
5:  R_{M'} : complement pattern of R_M
6:  input: R : receiver
7:  output: R : receiver, M : mask
8:  ILS ← the output of Algorithm 4 for R
9:  for i ← 1 to |ILS| do
10:     M ← ILS_i
11:     if R_{M'} ⊂ P then
12:         T ← R
13:         update T with R_{M'}
14:         if T ∈ P then
15:             return (R, ∅)
16:         if f(T) ≥ f(R) then
17:             R ← T
18:             return (R, M)
19:     else
20:         return (R, ∅)
21: return (R, ∅)
```

The newly proposed ILS construction procedure requires two individuals: the receiver that is currently mixing using the RM operator and the comparing individual, which is randomly selected from a population and has to be different from the receiver. During this operation, both the receiver and the comparing individual genes are compared to their reference gene marked by a randomly selected gene index (line 9 of Algorithm 4). Assuming that the comparing individual contains the optimal values of a whole Building Block (BB), we would like to flip only these values of the receiver genes that do not match the reference relation. This relation tells us if both values of the reference gene and the considered gene are the same or not. Let us consider the concatenation of three order-3 deceptive functions [3]. When 111 111 010 is the receiver individual, 111 000 111 is the comparing individual, and the 7^{th} gene has been selected as the reference gene, we would like to obtain the following mask $\{7, 9\}$ to get the optimal solution. Thus, we consider all vertices ordered descending by the max linkage value until we obtain the mask of length $\lfloor l/2 \rfloor$, or there are no more vertices to consider. Each candidate vertex is checked if the reference relation for the comparing individual is the same as for the receiver individual. If yes, then the vertex will be a member of the obtained ILS. Otherwise, it will not. Thanks to this filtering, even if a gene index should appear at the beginning of the mask, it would be skipped, because it already has a proper value.

To clarify the ILS construction procedure for the CM operator, let us provide an example. We consider DSM given in Table 1, 111000 as the receiver, 110111 as the comparing individual, and the 3^{rd} gene as the reference gene. The obtained ILS will be as follows: $<\{3\}, \{3, 4\}, \{3, 4, 5\}>$. Moreover, Figure 1 illustrates the crucial part of the ILS creation process, i.e., gene indexes filtering. The plus sign (+) indicates these gene indexes that can be included in resulted ILS.

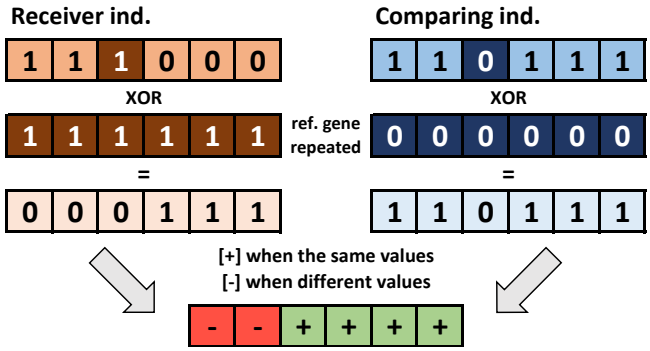

Figure 1: Sample gene indexes filtering by the 3^{rd} gene (marked by the dark background)

3.4 Supply bound

One of the reasons that the RM operation is terminated is when there are no individuals in a population which have the same values of genes on positions marked by the mask. This supply bound may prevent unnecessary fitness function evaluations. On the other hand, it can prevent from finding an optimal solution irremediably. An example of this phenomenon is optimizing concatenated trap function [3]. When after the BM operation, all 1*s* are replaced by 0*s* for at least one trap function, then the supply bound prevents return bit-flipping. Thus, in DSMGA-IIc, the BM operation is avoided when it leads to a lack of diversity of at least one gene (Algorithm 6). The modified BM procedure is shown in Algorithm 7.

Algorithm 6 Should execute the BM operator

GN_i^j : the number of i^{th} genes of value j
input: T : trial solution, M : mask
output: $true$ or $false$
for $i \leftarrow 1$ **to** $|M|$ **do**
  if $GN_{M_i}^{T_{M_i}} = 1$ **then**
    return $false$
return $true$

3.5 Population sizing scheme

DSMGA-II, DSMGA-IIe, and DSMGA-IIc are single-parameter methods. The only parameter is the population size. Their transformation to parameter-less methods may be done using the population sizing scheme proposed in [12]. Population-sizing DSMGA-II (psDSMGA-II), Population-sizing DSMGA-IIe (psDSMGA-IIe), and Population-sizing DSMGA-IIc (psDSMGA-IIc) maintain multiple populations at the same time. Each population is independent and has a different size. Furthermore, they do not share their linkage model. At the beginning of the method run, a single population containing two individuals is created. At each 4^{th} population iteration, the iteration of the population with the doubled size is executed. If it does not exist, then it is created. All populations that are considered as useless are removed.

Supply bound requires a nondiversified population to work effectively. Only then is it likely to find such gene values that are not present in any of the individuals. The nondiversified population is reached, e.g., by executing GHC at the beginning of each DSMGA-II-based method run. Thus, the creation of a new population in the population sizing scheme starts with applying GHC as well.

Algorithm 7 The modified BM procedure

M : mask, D : donor, P : population, f : evaluation function,
T : trial solution, E : candidates,
D_M : pattern of D extracted by M
input: D : donor, M : mask
output: P : population
$E \leftarrow \emptyset$
$improved \leftarrow false$
for $i \leftarrow 1$ **to** $|P|$ **do**
  $T \leftarrow P_i$
  if shouldBackMixing(T, M) **then**
    update T with D_M
    if $f(T) > f(P_i)$ **then**
      $P_i \leftarrow T$
      $improved \leftarrow true$
    else
      if $f(T) = f(P_i)$ **then**
        $E_i \leftarrow T$
      else
        $E_i \leftarrow P_i$
  else
    $E_i \leftarrow P_i$
if $improved = false$ **then**
  $P \leftarrow E$
return P

4 RESULTS

In this section, the results of the performed research are presented. We had two main objectives. First, we wish to check if the proposed psDSMGA-IIc performs better than psDSMGA-II and psDSMGA-IIe for NK-landscapes problems, and at least equally well for other considered problems. The second research objective was to compare psDSMGA-IIc with other up-to-date evolutionary methods that are psLTGA and P3.

The test problem set was taken from DSMGA-II papers [2, 9]. We only extended this set by problems of greater sizes. Our experiments were limited to 8 hours. The time-based stop condition was chosen because of the influence of fitness-caching [15]. Thus, to ensure the fairness of the comparison, all technical assumptions from [14, 15] were met. We used HP Elite Desk800 3.4 GHz 8GB RAM server with Intel Core i7-4770 CPU and Windows 7 64-bit installed to carry out all experiments. For each problem type, 30 independent runs were executed. The data pack that contains the full source code, the settings files, and detailed experiment results are available at https://github.com/kommar/psDSMGA-IIc. The source code of each competing method was downloaded from the Author's code repository and included in the data pack.

Table 2: Folded trap function ($k = 6$) values

Unitation	0	1	2	3	4	5	6
Function value	1.0	0.0	0.4	0.8	0.4	0.0	1.0

4.1 Experiments setup

All experiments were conducted using five optimization methods - psDSMGA-II, psDSMGA-IIe, psDSMGA-IIc, psLTGA, and P3. Each chosen competing optimizer is a recently proposed DSM-based method. It is important to allow for psDSMGA-II and psDSMGA-IIe because they are predecessors of psDSMGA-IIc. All methods are parameter-less, so any tuning procedure was needed.

The efficiency of all methods was tested using six different problem types. First, concatenated trap function was chosen. In general, trap function [3] is hard to optimize, because an optimizer is being deceived toward local optimum instead of looking for a global optimum that is hard to find. Concatenated trap function consists of m fully separable trap functions that may be considered as BBs. As in [2, 9], we defined k-order trap function as:

$$\text{trap}(u) = \begin{cases} 1 & , u = k \\ \frac{k-1-u}{k} & , u \neq k \end{cases} \tag{9}$$

where u denotes unitation, i.e., the number of 1s in a genotype. During our experiments, we used $k = 5$ and $m \in \{100, 150, 200, 300, 400\}$. Thus, the sizes of considered problems were 500, 750, 1000, 1500, and 2000, respectively. Another chosen problem was folded trap that may be known as bimodal trap [4]. Folded trap has two global optima - first in 0s, the other in 1s - and $\binom{k}{k/2}$ local optima, where k denotes the trap size. In this paper, we used $k = 6$ and considered problems of sizes $\{498, 750, 1002, 1500, 1998\}$. Table 2 presents the values of considered trap function.

The problems that have been described above are fully separable. We also conducted our experiments on overlapping problems. In this type of problem, a single gene may be a member of several BBs. We checked the effectiveness of considered methods using cyclic trap problems [19]. Cyclic trap of order k is very similar to concatenated trap of the same order. Instead of separable trap functions, cyclic trap consists of overlapping trap functions with wraparound. Here, we used $k = 5$ and problems of sizes $\{500, 752, 1000, 1500, 2000\}$. We also considered Nearest Neighbor NK-landscapes. In this problem, each gene is dependent on k neighbors. Neighbors are defined as succeeding genes. To check how the chosen methods managed different levels of overlapping, we used $k \in \{5, 6, 7\}$. For $k \in \{5, 6\}$ we considered problems of sizes $\{150, 300, 600\}$, whereas $\{147, 294, 595\}$ were chosen for $k = 7$. Many optimization problems may be reformulated to the maximum satisfiability problem (MAX-SAT). The main goal of solving this problem is to satisfy the maximum number of clauses containing 3 logical variables. We decided to check the effectiveness of optimization methods on MAX-SAT problems of the following sizes $\{50, 75, 100, 150, 200\}$. The last problem used in our experiments was Ising Spin Glass, which is the real world problem driven from statistical mechanics. In this problem, each gene in genotype represents spin of value -1 or 1. Adjacent spins are linked by a coupling constant J_{ij}. Let us consider the following formula:

$$-\sum_{i,j=0}^{l} x_i x_j J_{ij} \tag{10}$$

where x_i and x_j denote the i^{th} and j^{th} spin, respectively. The problem size is defined by l. Our goal is to minimize this formula. While conducting experiments, we used $l \in \{529, 784, 1024, 1521, 2025\}$ and considered only the $2D \pm J$ version of Ising Spin Glass. In this version, the coupling constant is equal to -1 or 1 and each spin has four neighbors due to defining a neighborhood by the $2D$ grid.

4.2 Experiments results

In this subsection, the results of our experiments are presented. First, we compare the effectiveness of psDSMGA-IIc with other chosen optimization methods. Secondly, we show how the CM operator influences the search process in terms of the length of the mask used during the BM operation.

The main results are presented in Table 3 and Table 4. The first table shows the median computation time in seconds that is necessary to find the optimum. The other presents the number of fitness function evaluations (FFE). Both measures are reported only for the largest instances for each optimization problem. Success rate (SR) is also reported to show the percentage of optimally solved problems. According to the Wilcoxon test (Table 5), all differences between methods in terms of computation time are significant.

Based on Table 3 and Figure 2, we can state that psDSMGA-IIc performs faster and scales better for cyclic traps, NK-landscapes, and 2D spin-glasses in comparison to psDSMGA-II and psDSMGA-IIe. Note that psDSMGA-IIc outperforms them, especially for NK-landscapes. It is the only DSMGA-II-based method that solves almost all of NKs6 instances (the others solve at most 3%) and any of NKs7 instances. On the other hand, psDSMGA-IIc is slower and scales worse than psDMSGA-II and psDSMGA-IIe when MAX-SAT problem is taken into account. psDSMGA-IIc also performs worse than psDSMGA-II in solving folded traps, but outperforms psDSMGA-IIe, because it can not reach the optimum during any run. According to Figure 2, it can be said that psDSMGA-IIc and psDSMGA-II perform almost equally in finding the optimum in concatenated traps. In contrast to psDSMGA-IIe whose SR is only 30% for the largest instance.

The comparison between psDSMGA-IIc and two state-of-the-art DSM-based methods, namely psLTGA and P3, shows that both perform faster (Table 3) for almost all problems than the method proposed in this paper. There are two exceptions. First, P3 is not able to find the optimum for folded traps in any run. Secondly, SR obtained by psLTGA is 17% lower when compares to psDSMGA-IIc for MAX-SAT problems. Nevertheless, if we take into account only these runs that result in finding the optimum, then psLTGA is much faster. It is interesting to compare psDSMGA-IIc with psLTGA on cyclic traps taking both computation measures into account. On the one hand, psLTGA is faster by a factor of 10 concerning computation time. On the other hand, when we compare FFE (Table 4), psLTGA is slower by a factor of 10. It may be explained by fitness caching [15] and by the fact that building Incremental Linkage Set is much more time consuming than building the Linkage Tree.

Table 3: Main results - median computation time necessary for reaching the optimum

Problem	l	psDSMGA-II SR [%]	psDSMGA-II Time median [s]	psDSMGA-IIe SR [%]	psDSMGA-IIe Time median [s]	psDSMGA-IIc SR [%]	psDSMGA-IIc Time median [s]	psLTGA SR [%]	psLTGA Time median [s]	P3 SR [%]	P3 Time median [s]
Concat. trap	**2000**	**100**	2839.5	30	24932.0	**100**	2512.0	**100**	**121.0**	**100**	139.5
Cyclic trap	**2000**	**100**	5386.5	**100**	5620.0	**100**	4203.0	**100**	**386.0**	**100**	625.0
Folded trap	**1998**	97	4000.0	0	N/A	**100**	5833.0	**100**	**2690.0**	0	N/A
NKs5	**600**	53	4775.5	93	9456.5	**100**	2148.5	**100**	**480.0**	**100**	798.5
NKs6	**600**	0	N/A	3	10176.0	**93**	8548.0	90	**1479.0**	90	4423.0
NKs7	**595**	0	N/A	0	N/A	13	15795.0	57	**8608.0**	**63**	11034.0
MAX-SAT	**200**	**100**	391.0	**100**	323.0	97	2003.0	80	332.0	**100**	**57.0**
2D spin-glass	**2025**	97	21873.0	87	19225.0	**100**	9637.0	**100**	1117.0	**100**	**701.5**

Table 4: Main results - median FFE necessary for reaching the optimum

Problem	l	psDSMGA-II SR [%]	psDSMGA-II FFE median	psDSMGA-IIe SR [%]	psDSMGA-IIe FFE median	psDSMGA-IIc SR [%]	psDSMGA-IIc FFE median	psLTGA SR [%]	psLTGA FFE median	P3 SR [%]	P3 FFE median
Concat. trap	**2000**	**100**	1.22E+06	30	9.74E+06	**100**	8.12E+05	**100**	4.13E+06	**100**	**4.63E+05**
Cyclic trap	**2000**	**100**	3.14E+06	**100**	3.14E+06	**100**	2.41E+06	**100**	1.29E+07	**100**	**1.04E+06**
Folded trap	**1998**	97	**1.50E+07**	0	N/A	**100**	2.19E+08	**100**	1.34E+08	0	N/A
NKs5	**600**	53	9.09E+07	93	4.88E+07	**100**	4,73E+07	**100**	1.13E+07	**100**	**9.55E+06**
NKs6	**600**	0	N/A	3	5.16E+07	**93**	1.70E+08	90	**3.15E+07**	90	4.98E+07
NKs7	**595**	0	N/A	0	N/A	13	2.83E+08	57	1.59E+08	**63**	**1.30E+08**
MAX-SAT	**200**	**100**	1.63E+07	**100**	1.12E+07	97	9.20E+07	80	1.82E+07	**100**	**3.15E+06**
2D spin-glass	**2025**	97	2.34E+07	87	1.34E+07	**100**	1.52E+08	**100**	4.18E+07	**100**	**2.82E+06**

Table 5: The Wilcoxon test p-values referring to computation time

Null hypothesis: psDMSGA-IIc is equal to	ps DSMGA-II	ps DSMGA-IIe	ps LTGA	P3
Concat. trap (2000)	0.003	N/A	0.000	0.000
Cyclic trap (2000)	0.000	0.000	0.000	0.000
Folded trap (1998)	0.000	N/A	0.000	N/A
NKs5 (600)	N/A	0.000	0.000	0.000
NKs6 (600)	N/A	N/A	0.000	0.000
MAX-SAT (200)	0.000	0.000	0.000	0.000
2D spin-glass (2025)	0.000	0.000	0.000	0.000

One of the reasons that DSMGA-II-based can find the optimal solution for many theoretical and practical problems is employing the BM operator. Its effectiveness depends on the mask that is the input to this operation. The main reason between psDSMGA-IIc and psDSMGA-II is replacing the RM operation by the CM operation. Thus, this change influences the effectiveness of the BM operation because the provided mask is obtained differently. In Table 6, we present the mean length of the mask given to the BM operator for each problem type. We choose the medium instances in terms of difficulty because SR of 100% is needed for reliable comparison. For almost all instances, the mask length is much longer for psDSMGA-IIc. Only for concatenated traps, both masks have similar lengths,

Table 6: Mean mask length during the BM operation

Problem	psDSMGA-II	psDSMGA-IIc
Concat. trap (1000)	8.01	13.28
Cyclic trap (1000)	8.94	39.10
Folded trap (1002)	3.72	31.63
NKs5 (300)	4.10	46.00
MAX-SAT (100)	1.21	11.97
2D spin-glass (1024)	5.13	74.36

and it is the only problem that psDSMGA-II and psDSMGA-IIc perform almost equally. When a mask is short, it probably can exchange only one building block. The longer one may consist of a few building blocks. It can be useful, especially when the considered problem is overlapping. NK-landscape is an example of a heavily overlapping problem, so this phenomenon can be the reason that psDSMGA-IIe performs the best from all DSMGA-II-based methods on NK-landscapes.

5 CONCLUSION

The objective of this paper was to propose a new psDSMGA-II version, that would improve the psDSMGA-II effectiveness when applied to solve NK-landscape problems, and that would preserve the psDSMGA-II effectiveness on other problems. Based on the presented results and the scalability analysis, this objective was

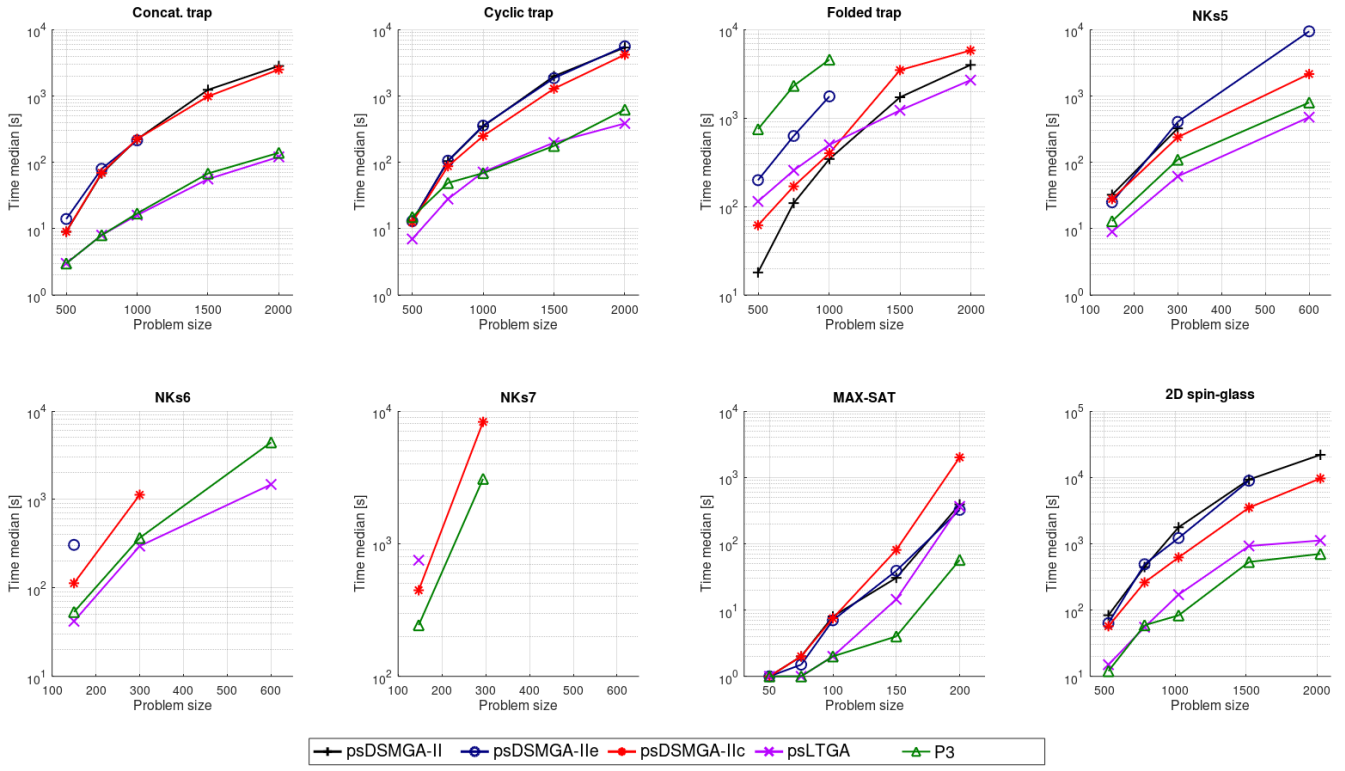

Figure 2: Time scalability of considered methods (only results with $SR \geq 80\%$ are taken into account)

almost reached. psDSMGA-IIc outperforms psDSMGA-II in solving NK-landscape problems and performs worse on less than half of other considered problems.

Based on the presented research, future work shall address the following issues. The RM and CM operators act differently. Perhaps for different problem types, usage of one of the above operators will be beneficial during the different optimization stage. Thus, an adaptation strategy may be proposed to decide whether RM or CM should be applied. Finally, psDSMGA-IIc shall be faced with other hard theoretical and practical problems.

ACKNOWLEDGMENTS

This work was supported in by the Polish National Science Centre (NCN) under Grant 2015/19/D/ST6/03115 and the statutory funds of the Department of Computational Intelligence.

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)*. 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)*. 745–752.

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

[4] Kalyanmoy Deb, Jeffrey Horn, and David E. Goldberg. 1993. Multimodal Deceptive Functions. *Complex Systems* 7, 2 (1993), 131–154.

[5] Brian W. Goldman and William F. Punch. 2014. Parameter-less Population Pyramid. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. 785–792.

[6] B. W. Goldman and W. F. Punch. 2015. Fast and Efficient Black Box Optimization Using the Parameter-less Population Pyramid. *Evol. Comput.* 23, 3 (2015), 451–479.

[7] Brian W. Goldman and Dirk Sudholt. 2016. Runtime Analysis for the Parameter-less Population Pyramid. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. 669–676.

[8] 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.

[9] 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*. 519–526.

[10] Kazuyuki Inoue, Taku Hasegawa, Yuta Araki, Naoki Mori, and Keinosuke Matsumoto. 2015. Adaptive Control of Parameter-less Population Pyramid on the Local Distribution of Inferior Individuals. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO '15)*. 863–870.

[11] 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)*. 109–110.

[12] 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)*. 289–290.

[13] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statist.* 22, 1 (1951), 79–86.

[14] Halina Kwasnicka and Michal Przewozniczek. 2011. Multi Population Pattern Searching Algorithm: A New Evolutionary Method Based on the Idea of Messy Genetic Algorithm. *IEEE Trans. Evolutionary Computation* 15 (2011), 715–734.

[15] 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)*. 241–242.

[16] 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.

[17] Dirk Thierens and Peter A.N. Bosman. 2011. Optimal Mixing Evolutionary Algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*. 617–624.
[18] Dirk Thierens and Peter A.N. Bosman. 2013. Hierarchical problem solving with the linkage tree genetic algorithm. In *Proceeding of the 2013 Annual Conference on Genetic and Evolutionary Computation Conference*. 877–884.
[19] Tian-Li Yu, Kumara Sastry, and David E. Goldberg. 2005. Linkage Learning, Overlapping Building Blocks, and Systematic Strategy for Scalable Recombination. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation (GECCO '05)*. 1217–1224.
[20] 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)*. 312–313.