Handling Overlap by Pairwise Linkage Detection, Incremental Linkage Set, and Restricted / Back Mixing:

DSMGA-II

Shih-Huan Hsu
Tian-Li Yu

TEIL Technical Report No. 2015002
February, 2015

Taiwan Evolutionary Intelligence Laboratory (TEIL)
Department of Electrical Engineering
National Taiwan University
No.1, Sec. 4, Roosevelt Rd., Taipei, Taiwan

http://teil.ee.ntu.edu.tw

Handling Overlap by Pairwise Linkage Detection, Incremental Linkage Set, and Restricted / Back Mixing: DSMGA-II

Shih-Huan Hsu, Tian-Li Yu
National Taiwan University
r02921076@ntu.edu.tw, tianliyu@ntu.edu.tw

Abstract

This paper proposes a new evolutionary algorithm to efficiently solve optimization problems with overlapping linkage structures. The proposed algorithm adopts pairwise linkage detection and stores the information in the form of dependency structure matrix (DSM). A new linkage model, called the incremental linkage sets, is then constructed by using the DSM. Inspired by the idea of optimal mixing, restricted mixing and back mixing are proposed. The former aims at efficient exploration with certain constrains. The latter aims at exploitation by refining the DSM so as to reduce unnecessary evaluations. Experimental results show that the proposed method outperforms LT-GOMEA in terms of number of function evaluations on the concatenated/folded/cyclic trap problems and the NK-landscape problems with various degrees of overlapping.

1 INTRODUCTION

Since the importance of linkage and problem decomposition has been addressed in the field of evolutionary computation [HollandHolland1992], many algorithms that adopt linkage learning have been developed. In 2003, Yu *et al.* [Yu & GoldbergYu & Goldberg2006] borrowed the concept of DSM from the organization theory and proposed DSMGA. One of its key mechanisms is pairwise linkage detection. The linkage information is then stored in a DSM and can be latter used to construct different linkage models, such as the marginal product model [Harik & HarikHarik & Harik1999] and the building-block (BB) graph [YuYu2006]. In 2010, Thierens *et al.* [Thierens & BosmanThierens & Bosman2011] proposed the optimal mixing operator (OM) and adopted it in the LT-GOMEA [Bosman & ThierensBosman & Thierens2012,ThierensThierens2010]. Unlike most traditional selection-crossover combination in GAs, the decision-making in OM is noise-free and hence results in a much smaller population-sizing requirement than most estimation-of-distribution algorithms. Combined with the linkage-tree model, the LTGA family has shown strong optimization ability on a wide range of problems [Martins, Fonseca, & DelbemMartins et al.2014,Thierens & BosmanThierens & Bosman2013,Luong, La Poutré, & BosmanLuong et al.2014]. However, LT may not be the most proper linkage model for problem decomposition—recent researches involves LT pruning [Bosman & ThierensBosman & Thierens2013,Wang, Tung, & YuWang et al.2014] and more expressive linkage models such as LN [Bosman & ThierensBosman & Thierens2012]. In this paper, we propose a new evolutionary algorithm. The proposed algorithm adopts the pairwise linkage detection from DSMGA and uses it to construct a new linkage model called the incremental linkage set (ILS). Inspired by the idea of OM, we propose two recombination operators, the restricted and the back mixing. Combining the DSM linkage information, ILS, and the two new mixing operators, the proposed algorithm empirically demonstrates stronger optimization ability than the LT-GOMEA on the concatenated trap, folded trap, cyclic trap with overlapping, and the NK-landscape problems with various degrees of overlapping.

The remainder of this paper is organized into three main parts. The first part revisits researches that are directly related to this paper. The second part details DSMGA-II and its operators. The third part describes the experiments and demonstrates the results, and finally conclusion follows.

2 RELATED WORKS

In standard genetic algorithms, problems can be solved effectively and efficiently if good solutions are mixed adequately. To match different kinds of problem structures, many techniques for building tunable models have been developed during the past decade. This section describes two important researches that we incorporated in our works.

2.1 Dependency Structure Matrix (DSM)

Borrowed from the concept of organization theories, the dependency structure matrix genetic algorithm (DSMGA) utilizes DSM and clustering algorithms to detect interactions among variables. A DSM is essentially an adjacent matrix where each entry represents the pairwise information between two variables. The clustering method utilizes the concept of minimal description length principle (MDL) [RissanenRissanen1978] to obtain a model that minimizes the sum of the model description length and the mismatched data description length. With the extracted building-block information, DSMGA adopts building-block wise crossover instead of traditional gene-wise crossover. Such mechanisms have shown to be beneficial to the search efficiency due to less disruption of subsolutions.

2.2 Optimal Mixing (OM)

The optimal mixing evolutionary algorithm (OMEA) [Thierens & BosmanThierens & Bosman2011] was first proposed as the recombinative optimal mixing evolutionary algorithm (ROMEA) and the gene-pool optimal mixing evolutionary algorithm (GOMEA). Typical genetic algorithms proceed recombination on two parents to generate offspring, and the fitness value is then evaluated. However, OM operators apply function evaluation during the recombination to check if each of the operations improves current solution and take the change only if the fitness value increases. The OM manner somewhat acts like building-block wise local search with interchangeable models, which are also defined as family of subsets (FOS) in [Thierens & BosmanThierens & Bosman2011]. A FOS contains subsets of a certain set S. Each subset of FOS may appear more than once. The set S contains all problem variable indexes, *i.e.,* $\{1, 2, \ldots, \ell\}$. A FOS $\mathcal{F}$ can be written as $\mathcal{F} = \langle \boldsymbol{F}^1, \boldsymbol{F}^2, \ldots, \boldsymbol{F}^{|\mathcal{F}|} \rangle$ where $\boldsymbol{F}^i \subseteq S$, i $\in \{1, 2, \ldots, |\mathcal{F}|\}$. Moreover, every problem variable index is contained in a least one subset in $\mathcal{F}$ to ensure that every linkage is used in mixing operators, *i.e.,* $\forall i \in S\ :\ \left(\exists j \in \{1, 2, \ldots, |\mathcal{F}|\}\ :\ i \in \boldsymbol{F}^j \right)$. According to the results in [Thierens & Bosman-Thierens & Bosman2011, Bosman & ThierensBosman & Thierens2012], OMEAs are superior to most GAs in many aspects while the models are properly constructed. One of the noticeable results is that a way smaller population size is required because of the noise-free mixing, which might alleviate the effect of decision-making in terms of the population sizing [Goldberg, Deb, & ClarkGoldberg et al.1991]. Furthermore, OM operators are capable of dealing with problems with overlapping structures efficiently with certain FOS [Bosman & ThierensBosman & Thierens2012]. The pseudo-code for GOMEA is given in Algorithm 1 and 2.

Algorithm 1: GOMEA

```
Input  : $f$: fitness function, $n$: population size,
         $\mathcal{F}$: FOS
Output: optimization solution

randomly generate $n$ instances for $\mathcal{P}$
while $\neg$ShouldTerminate($\mathcal{P}$) do
    for $i = 1$ to $n$ do
        $O_i \leftarrow$ GenepoolOptimalMixing($f, P_i, \mathcal{P}, \mathcal{F}$)
    for $i = 1$ to $n$ do
        $P_i \leftarrow O_i$
return the best instance in $\mathcal{P}$
```

Algorithm 2: Genepool Optimal Mixing

```
Input  : $f$: fitness function, $R$: receiver,
         $\mathcal{P}$: population, $\mathcal{F}$: FOS
Output: $O$: offspring
$O \leftarrow R$
$B \leftarrow R$
for $i = 1$ to $|\mathcal{F}|$ do
    $r \leftarrow$ random number from 1 to $|\mathcal{P}|$
    $D \leftarrow P_r$
    $B_{\mathbf{F}^i} \leftarrow D_{\mathbf{F}^i}$
    if $f(B) \geq f(O)$ then
        $O_{\mathbf{F}^i} \leftarrow B_{\mathbf{F}^i}$
    else
        $B_{\mathbf{F}^i} \leftarrow O_{\mathbf{F}^i}$
return $O$
```

3 DSMGA-II

The following section gives details of the dependency structure matrix genetic algorithm II (DSMGA-II), which is virtually the extension of DSMGA [Yu & GoldbergYu & Goldberg2006] combined with the idea of OM. We first introduce the framework of DSMGA-II. The concept of DSM construction and incremental linkage sets is described then, and finally two mixing operators are shown — the restricted mixing and the back mixing. Note that all the algorithms in this paper are assumed to solve maximization problems for ease of expression.

3.1 Framework of DSMGA-II

The major components of DSMGA-II are linkage information retrieval via pairwise detection, expressive linkage model construction, and efficient mixing that balance between exploration and exploitation. The algorithm first adopts pairwise linkage detection and stores the information in DSM. After the building-block information is obtained, the mixing operators then proceed with incremental linkage sets, which can be seen as a specific type of FOS. Moreover, in estimation of distribution algorithms (EDAs), hill climbing strengthens the signals of dependencies among variables and improves the quality of model building [Chen, Hsu, Yu, & ChienChen et al.2013]. A bit-flipping local search operator is therefore adopted right after the initialization of population. The DSM is updated at the beginning of each generation with the after-selection population. The selection is tournament selection with the selection pressure s according to the suggested results in [Yu, Sastry, Goldberg, & PelikanYu et al.2007]. To prevent the overfitting on the model building, the algorithm repeats $O(\ell)$ times per generation. Note that the population after selection is only used for updating the DSM; the rest parts of algorithm proceed with the original population.

The pseudo-code for the framework is given in Algorithm 3. Each operator is detailed in the following sections. The population is denoted by $\mathcal{P}$, with the chromosomes $P_1, \dots, P_{|\mathcal{P}|}$, the problem size ℓ, and the after-selection population $\mathcal{S}$. The R is a constant which is proportional to ℓ. The DSM is denoted by $\mathcal{D}$. $\mathcal{I} = \langle I_1, I_2, \dots I_{|\mathcal{P}|}\rangle$, is a random permutation of $\{1, 2, \dots |\mathcal{P}|\}$, and the incremental linkage sets is denoted by $\mathcal{L}$, which is a set of masks and is elaborated in the following section. Each iteration of "while" loop is a generation, and the algorithm terminates while the optimal solution appears.

3.2 Structures and Learning

A dependency structure matrix is a graph representation of the dependency between two variables, where each entry e_{ij} is the measure of dependency between node i and node j. Entries could be real numbers or integers. The larger the e_{ij} is, the more the significant measure between node i

Algorithm 3: DSMGA-II

```
$\mathcal{P}$: population, $\ell$: problem size $\mathcal{D}$: DSM, $\mathcal{L}$: incremental linkage sets, $R$: constant
randomly initialize population $\mathcal{P}$
$\mathcal{P} \leftarrow$ RunLocalSearch($\mathcal{P}$)
while $\neg$ShouldTerminate do
    $\mathcal{S} \leftarrow$ TournamentSelection($\mathcal{P}$, $s$)
    $\mathcal{D} \leftarrow$ UpdateMatrix($\mathcal{S}$)
    for $k = 1$ to $R$ do
        $\mathcal{I} \leftarrow$ random permutation from 1 to $|\mathcal{P}|$
        for $i = 1$ to $|\mathcal{P}|$ do
            $(P_{I_i}, L_s) \leftarrow$ RestrictedMixing($P_{I_i}$)
            $\mathcal{P} \leftarrow$ BackMixing($P_{I_i}, L_s$)
return the best instance in $\mathcal{P}$
```

and node j is. Once the matrix is constructed, the problem can be interpreted as a graph, and the variables are the nodes.

Definition 1. (*Maximum-weight connected subgraph*, **MWCS**) *Given an undirected graph* $G = (V, E)$ *with edge weights* $w : E \to R$*. MWCS is a complete subgraph* $G' = (V', E')$ *of* G*, (i.e.,* $V' \subseteq V$*,* $E' \subseteq E$*,* $\forall u, v \in V' \Rightarrow (u, v) \in E'$*) such that* $\sum_{e \in E'} w(e)$ *is maximum.*

In this paper, pairwise dependency is measured by mutual information [Kullback & LeiblerKullback & Leibler1951], which is shown as follows:

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

where X and Y are two random variables, and x and y are outcomes of the random variables. $p(x,y) = p(x)p(y)$ if X and Y are independent, and $I(X;Y)$ equals to 0 thus.

Many clustering algorithms for DSM have been proposed. The linkage tree genetic algorithm (LTGA) [ThierensThierens2010] applies a hierarchical clustering technique with normalized variation of information to build the linkage tree model of population, and it was further improved in [Thierens & BosmanThierens & Bosman2011], which uses pairwise linkage information with average linkage clustering to achieve efficient computation without apparent drawbacks. We simply use mutual information as the dependency measure, which is the same as [Yu & GoldbergYu & Goldberg2006]. However, instead of clustering variables, we find a specific subgraph called approximation maximum-weight connected subgraph (AMWCS). The concept of AMWCS is similar to MWCS [Lee & DoolyLee & Dooly1998, Alvarez-Miranda, Ljubic, & MutzelAlvarez-Miranda et al.2013], but they are essentially different. The algorithm initializes an AMWCS from a certain node, and iteratively adds a single node into AMWCS. For example, consider a graph $G = (V, E)$, $V = \{v_1, v_2, \ldots, v_5\}$, and an AMWCS $G' = (V', E')$, $V' = \{v_1, v_2, v_3\}$ and a set of candidate nodes $\mathcal{N} = \{v_4, v_5\}$. The objective is to find a node of candidate set that maximize the weight of AMWCS after insertion. The equation can be expressed as:

$$index = \operatorname*{argmax}_{i \in \{4,5\}} \sum_{j=1}^{3} I(v_i, v_j) \ , \tag{2}$$

where $I(v_i, v_j)$ is the mutual information between variable v_i and v_j.

The incremental linkage sets is a specific type of FOS. As shown in Algorithm 4, an index is first randomly chosen as the initial AMWCS, and the rest of indexes are gradually inserted into the AMWCS with the above method until the termination criterion satisfied. According to the results in [YuYu2006], there are three keys to achieve efficient and effective recombination on problems with overlapping structures: (1) Minimization of building-block disruptions, (2) Maximization of effective exchange length, and (3) Non-determinism of information exchange. The third argument suggests that every pair of building-blocks should have a non-zero probability to be mixed, and that is why the algorithm start an AMWCS from a random index. The incremental linkage sets consists of the AMWCSs after every iteration of AMWCS finding. For instance, consider a problem with length 5, and the index 3 was randomly chosen from $\{1, 2, 3, 4, 5\}$ at first (Figure 1). After 4 iterations, the inserted sequence of nodes into AMWCS is $\mathcal{Q} = \langle 3, 1, 5, 4, 2\rangle$, and the incremental linkage sets is $\mathcal{L} = \langle\{3\}, \{3, 1\}, \{3, 1, 5\}, \{3, 1, 5, 4\}, \{3, 1, 2, 4, 2\}\rangle$. Note that both sets are ordered sets.

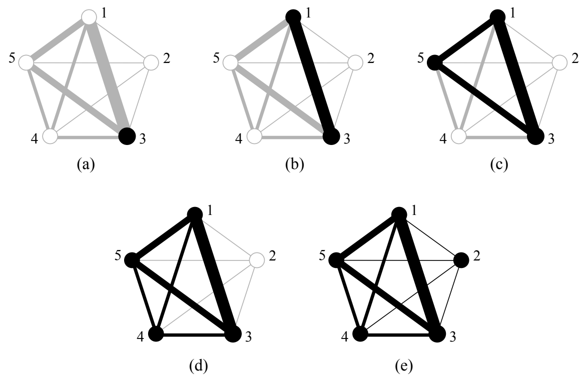

Figure 1: (a) to (e) shows the AMWCS in each iteration. The nodes represent variables and the gray edges represent the dependency measure between pairs. The larger the measure is, the wider the edge is. Black nodes and edges represent the determined AMWCS.

3.3 Mixing Operators

There are two mixing operators in DSMGA-II: one is the restricted mixing, and the other is the back mixing. Unlike canonical genetic algorithms (GAs), DSMGA-II does not actually generate offspring by recombination of parent solutions. Instead, it applies the restricted mixing operator, which flips bits with the the masks. Each mask is essentially a set of indexes that indicates which variables should be considered together during mixing operations. One reason for that is one of the characteristics of mutual information: the higher the mutual information is, the higher the possibility is for the corresponding bits being different from the most solutions in current population. Another reason is about the search efficiency. When two solutions exchange subsolutions with certain mask, each allele might not actually varies because of the same allele that comes from the matching chromosome. Although this drawback may not actually increase NFE, it is still time-consuming. For these two reasons, the restricted mixing starts from choosing a receiver. Subsolutions in the receiver is flipped with masks in the incremental linkage sets from small size to large size, and the change is preserved only if the fitness does not decrease. Once the solution is improved, restricted mixing terminates. Moreover, in terms of building-block supply [Goldberg, Deb, & ClarkGoldberg et al.1991], we believe that good subsolutions should exist in current population. Accordingly, restricted mixing also terminates while the complementary pattern of receiver does not exist in the current population. In other word, it can be considered as a mixing with the chromosome which contains the certain pattern, and this is the reason for calling this operation restricted mixing. The pseudo-code for the restricted mixing is given in Algorithm 4. The population is denoted by $\mathcal{P}$ with problem size ℓ. The sequence of nodes is denoted by $\mathcal{Q}$, and the incremental linkage sets $\mathcal{L} = \langle L_1, L_2, \ldots, L_{|\mathcal{L}|}\rangle$ while each element is a mask. P_L is the pattern of chromosome P selected with mask L. P'_L is the complementary pattern of P_L. T is the trial solution, and the evaluation function is f.

The following figure is an example of restricted mixing. Consider an incremental linkage sets $\mathcal{L}$, a population $\mathcal{P} = \{P_1, P_2, \ldots, P_5\}$, and a receiver $P = P_4$:

After the restricted mixing finishes, the successfully flipped receiver during the restricted mixing becomes the donor of the back mixing. Every chromosomes in the population are then mixed with the flipped pattern of donor, and the change is adopted only if the fitness is improved. This

Algorithm 4: Restricted Mixing

```
$\mathcal{P}$: population, $\ell$: problem size, $\mathcal{Q}$: sequence of nodes
$\mathcal{L}$: incremental linkage sets, $f$: evaluation function, $T$: trial solution
Input: $P$ : receiver

$\mathcal{Q} \leftarrow$ random number from 1 to $\ell$
while $P'_{\mathcal{Q}} \in \mathcal{P}$ do
    add $\mathcal{Q}$ into $\mathcal{L}$
    join the nearest node into $\mathcal{Q}$
for $i = 1$ to $|\mathcal{L}|$ do
    $T \leftarrow P$
    $T_{L_i} \leftarrow T'_{L_i}$
    if $f(T) \geq f(P)$ then
        $P \leftarrow T$
        return $(P, L_i)$
```

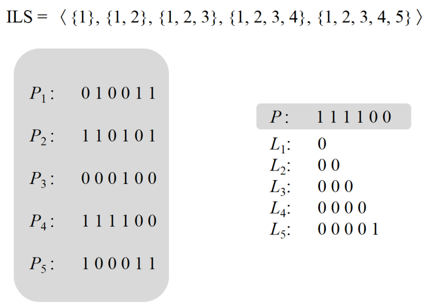

Figure 2: An example of restricted mixing. Population $\mathcal{P}$ is on the left, and receiver $P = P_4$ is on the right. The complementary pattern with the first mask $\{1\}$ of receiver is 0, which exists in chromosome 1 and 3. With the second mask $\{1, 2\}$, the complementary pattern is 00, which exists in chromosome 3 only, and so on. However, the complementary pattern with the fourth mask $\{1, 2, 3, 4\}$ is 0000, which does not exist in any chromosome of the population. Therefore, the last mask that should be utilized is the third one.

operation is called the back mixing. Note that the acceptance criteria is different from restricted mixing. Real-world problems may contain various landscapes such as plateaus and basins, which are the attractions for solutions as well as the local optima. Many operators have been developed to deal with such difficulties, such as the forced improvements (FI) [Bosman & ThierensBosman & Thierens2012]. However, diversity issue should be handled carefully, and that is why the change does not adopted when the fitness of trial is the same as the original chromosome's in back mixing. In short, the idea behind this operator is the graph refining. Better subsolutions will not be replaced because of the greedy recombination, while worse ones should be replaced to eliminate wrong global information. The pseudo-code for the back mixing is given in Algorithm 5.

4 TEST PROBLEMS AND EXPERIMENTS

This section first describes the benchmark problems used in this paper. The setup of the experiments is then given, followed by experimental results and discussions.

Algorithm 5: Back Mixing

```
$\mathcal{P}$: population, $f$: evaluation function, $T$: trial solution
Input: $D$ : donor, $L$: mask
for $j = 1$ to $|\mathcal{P}|$ do
    $T \leftarrow \mathcal{P}_j$
    $T_L \leftarrow D_L$
    if $f(T) > f(\mathcal{P}_j)$ then
        $\mathcal{P}_j \leftarrow T$
return $\mathcal{P}$
```

4.1 Optimization Problems

In our research, five types of linkage benchmark problems are considered. First problem is commonly known as onemax:

$$f_{onemax}(\mathbf{x}) = \sum_{i=1}^{\ell} x_i. \tag{3}$$

The second one is the concatenated trap function [Deb & GoldbergDeb & Goldberg1994]. The function is composed of additively separable trap functions, and each of them contains k variables. The number of variables of the concatenated trap function is $m \cdot k$. In this paper, the number of variables in the test function is referred as the problem size, denoted by ℓ. It is well-known that the deceptive-trap problems can only be efficiently solved when the underlying structure is detected and preserved while mixing [Thierens & GoldbergThierens & Goldberg1993]. The fitness function can be expressed in the following equation:

$$f_{m,k}^{trap}(\mathbf{x}) = \sum_{i=1}^{m} f_k^{trap}\left(\sum_{j=i\cdot k-k+1}^{i\cdot k} \mathbf{x}_j\right), \tag{4}$$

where

$$f_k^{trap}(u) = \begin{cases} 1 & if \text{u=k}, \\ \frac{k-1-u}{k} & otherwise. \end{cases} \tag{5}$$

The third fitness function is the cyclic trap function [Yu, Sastry, & GoldbergYu et al.2005]. The function is composed of overlapping trap functions with wraparound. It is similar to the (m, k) deceptive-trap function. However, all subfunctions, the trap functions, are overlapped by one bit. There are total $m \cdot (k-1)$ variables in the cyclic trap function, which means $\ell = m \cdot (k-1)$. Solving the problem is more difficult than the (m, k) trap problem, because one cannot simply split the problem into trap functions. The fitness function can be expressed in the equation:

$$f_{m,k}^{cyclic}(\mathbf{x}) = \sum_{i=1}^{m} f_k^{trap}\left(\sum_{j=i\cdot(k-1)-k+2}^{i\cdot(k-1)+1} \mathbf{x}_j\right), \tag{6}$$

where

$$f_k^{trap}(u) = \begin{cases} 1 & if \text{u=k}, \\ \frac{k-1-u}{k} & otherwise. \end{cases} \tag{7}$$

and

$$\mathbf{x}_j = \mathbf{x}_{j-\ell}, \text{ if } \ell < j \leq 2\ell. \tag{8}$$

The fourth fitness function that we consider is the folded trap function [Goldberg, Goldberg, Deb, & HornGoldberg et al.1992]. The function is composed of several subfunctions with irregular fitness landscape. There are many variants of folded trap. We simply use bipolar deceptive function with $k = 6$ in our experiment. A bipolar deceptive function contains two global optima and a number of deceptive optima. The key difficulty of this problem is that the chromosomes

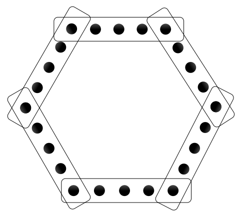

Figure 3: An example of Cyrlic trap, which $k = 5$ and $\ell = 24$.

with unitation $u = \ell/2$ appear mostly in the initial population, and are also the deceptive optima. The fitness function with $k = 6$ can be expressed in the equation:

$$f^{folded}_{m,k=6}(\mathbf{x}) = \sum_{i=1}^{m} f^{folded}_{k=6}\left(\sum_{j=i\cdot k-k+1}^{i\cdot k} \mathbf{x}_j\right), \tag{9}$$

where

$$f^{folded}_{k=6}(u) = \begin{cases} 1 & if|u-3| = 3, \\ 0.8 & if|u-3| = 0, \\ 0.4 & if|u-3| = 1, \\ 0 & if|u-3| = 2. \end{cases} \tag{10}$$

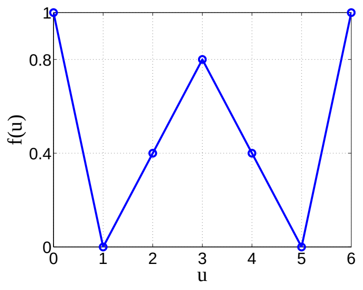

Figure 4: A folded trap function with $k = 6$.

The last fitness function we used is the NK landscape functions with nearest-neighbor interactions and tunable overlap [Pelikan, Sastry, Goldberg, Butz, & HauschildPelikan et al.2009]. NK landscape functions are composed of overlapped, randomly generated subfunctions. In our experiments, various types of NK landscape functions without wraparound are adopted. There are three parameters of the function, which are n, k, and s. n is the problem size, which is also denoted by ℓ. n can be any integer satisfying that $n - k - 1$ is a non-negative multiple of s. k is the number of neighbors of a bit. All input size of subfunctions are $k + 1$. s, the step size, is the offset of two adjacent subfunctions. The function is named according to s. For example, an NK landscape function with $s = 1$ would be called NK-S1. The optimal solution can be derived in polynomial time by dynamic programming given the problem structure. The equation of the NK landscape function is

$$f^{NK}_{n,k,s}(\mathbf{x}) = \sum_{i=0}^{(n-k-1)/s} f^{subNK}_{k,i}(\mathbf{x}_{i\cdot s+1}, \mathbf{x}_{i\cdot s+2}, \ldots \mathbf{x}_{i\cdot s+k+1}), \tag{11}$$

where $f^{subNK}_{k,i}$ are randomly generated functions, subject to the constraint that $f^{subNK}_{k,i}(\mathbf{x}) \in [0, 1]$ for any valid input $\mathbf{x}$. The fitness landscape of the problems depends on the subfunctions, hence the problems are named NK landscape problems.

4.2 Experiment setup

The most commonly used performance measure in the EA field is the minimum number of function evaluations (NFEs) required for the successful convergence. Such measurement is not easily

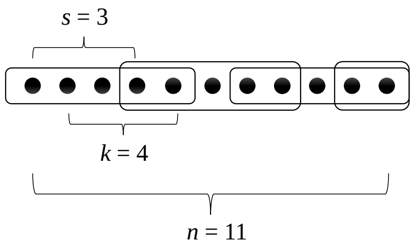

Figure 5: An example of NK-S3, which $k = 4$ and $n = 11$.

retrieved since the required NFEs vary with the population size. Traditionally, most researches utilize a bisection procedure [PelikanPelikan2005] to find the minimum population size that is sufficiently large for a certain number of consecutive successful convergences; the required NFEs at that population size is then used as the performance measure. However, such minimum population sizing usually does not yield the minimum NFEs as desired, and the differences vary for different algorithms.

For fair comparisons, we adopt an adaptive sweeping procedure to find the minimum NFEs. The procedure starts by sweeping the population size through a reasonable range with a predefined step. The NFEs are averaged over a certain number of consecutive successful hits. If the algorithm fails to converge to the global optimum with the population size, the NFEs is recorded as infinite large. The sweeping range is then narrowed around the population size that yields the minimum NFEs with a smaller step. The procedure iterates until the sweeping range becomes small enough. In this paper, the requirement is 10 consecutive successful hits, the initial sweeping range is [0,1000], the initial step size is 200 and then divided by 2 for each iteration, and the termination condition is the sweeping range is within 5% of the population size.

DSMGA-II is tested on the optimization problems with size $\ell = 100, 200$, and 400, except for the folded trap function, which is performed with the problem size $\ell = 120, 240$, and 480. The selection pressure s is set to 2. For every optimization problem that we consider, we perform 100 independent runs. We furthermore included LT-GOMEA that was published with [Bosman & ThierensBosman & Thierens2013] in our experiments. Besides LT-GOMEA, several GOMEAs with different models have been proposed [Bosman & ThierensBosman & Thierens2013, Bosman & ThierensBosman & Thierens2012]. However, the source codes have not published yet, and we are not able to reproduce them yet as well. Therefore, we simply use LT-GOMEA with forced improvements, and apply tournament selection with selection pressure 2 for model building.

For onemax problem, the greedy hill climbing (GHC) operator simply solve the problem after the initialization of DSMGA-II. As a result, onemax problem is excluded in our experiments.

For NK landscape problem, we test our algorithm on NK-S1 to NK-S5 (*i.e.*, parameter s = 1, 2, 3, 4, and 5) with 100 randomly generated instances each, and parameter n is set to 4. NK-S5 is actually non-overlapping NK problem, while NK-S1 is the problem that maximally overlapped. Accordingly, we are able to see how degrees of overlap affect the performance of the algorithm.

For concatenated trap and cyclic-trap, the subfunction size k is set to 5, and the k of folded trap is set to 6.

4.3 Results and Discussions

To investigate the use of the back mixing operator, we run DSMGA-II without back mixing. The results is shown in Figure 6. NFEs of all problems increase, especially for the problems with overlapping structures.

The results for the NFEs are shown in Figures 8. DSMGA-II requires fewer NFEs than LT-GOMEA on all optimization problems that we include in our experiments. For non-overlapping optimization problems, including concatenated trap, folded trap, and NK-S5, the difference between two algorithms is apparent. Intuitively, DSMGA-II is well-performed on problems with deceptive structures probably owing to its flipping-like behavior during restricted mixing. However, DSMGA-II also shows good performance on NK-s5 problem, which is commonly regarded as

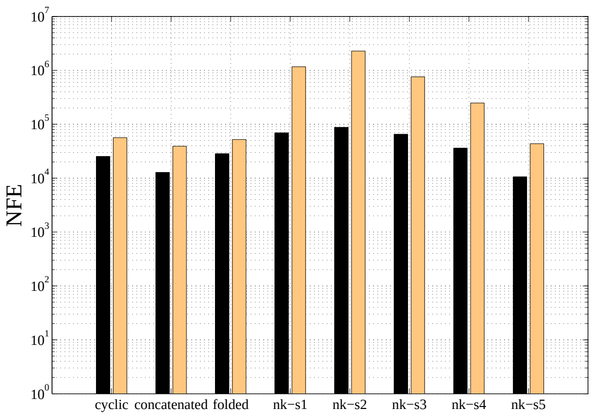

Figure 6: DSMGA-II with/without back mixing on problem size = 100 (folded trap = 120).

a general case in problem structures.

Interestingly, the most significant difference between two algorithms in our experiments occur on folded trap problem. As already known, the structure of folded trap contain lots of plateau, especially the bipolar deceptive function. Such landscape may stuck the solutions in it. The GOMEA deal with the difficulty with stronger drift (*i.e.*, change of solution is accepted when the fitness of neighboring solution is the same), such as forced improvements [Bosman & ThierensBosman & Thierens2012] and the change of the acceptance criteria [Bosman & ThierensBosman & Thierens2013]. Nevertheless, DSMGA-II only allows solutions to move along plateau during restricted mixing. We believe such mechanism is an adequate balance between exploration and exploitation.

The results also demonstrate that DSMGA-II is capable of handling linkage-underlying problems with overlapping structures. This is indicative that the ILS model is competent to express overlapping linkage relations. A possible scenario of model building is shown in the Figure 7. It

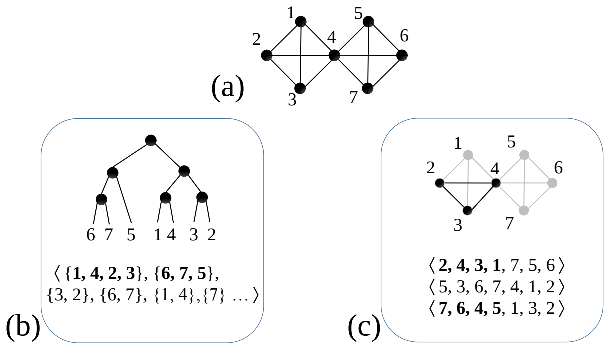

Figure 7: A snapshot of model building. (a) is the real structure of problem with node 4 overlapped. (b) is a LT model constructed from (a), and (c) is a ILS model with some possible AMWCS sequence. Both LT and ILS model capture the problem structure. However, mask {4, 5, 6, 7} only occurs in ILS within a single generation.

is noteworthy that the results of NK-S1 to NK-S5 indicate that the difference between DSMGA-II and LT-GOMEA becomes larger when the degree of overlap of problem structures decrease. The reason is not clear yet and need to be further investigated. The scalability of both algorithms seems similar on problems with severely overlapping structures. DSMGA-II shows good scalability on both non-overlapping and deceptive structures. It is furthermore interesting that the scalability curve of DSMGA-II on folded trap problem levels off as the problem size increases. Although the design of DSMGA-II aims at problems with overlapping structures, it does not compromise the optimization ability on non-overlapping problems.

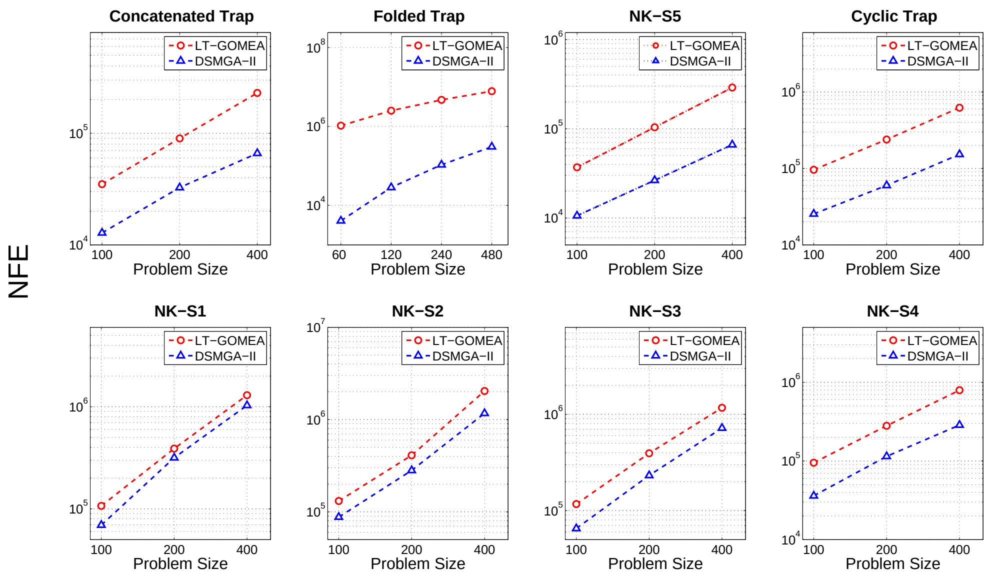

Figure 8: Scalability of DSMGA-II and LT-GOMEA.

5 CONCLUSION

This paper proposes a new evolutionary algorithm with linkage learning, called DSMGA-II. Similar to DSMGA, it adopts pairwise linkage detection and stores the information in a DSM. The linkage information is then used to construct a newly proposed linkage model, the incremental linkage sets (ILS), which is expressive for problems with both overlapping and non-overlapping structures. Inspired by OM, the restricted mixing and the back mixing are designed to balance between exploration and exploitation. For each receiver, the restricted mixing chooses specific donors according to the ILS constructed from the global information instead of random. The back mixing further refines the DSM using promising sub-solutions and hence reduces unnecessary function evaluations. Empirical results shows that DSMGA-II outperforms LT-GOMEA in terms of the number of function evaluations without compromising the scalability on several benchmark problems, including concatenated trap, folded trap, cyclic trap with overlapping, and the NK-landscape problems with various degrees of overlapping.

As for future work, we would like to test DSMGA-II on more problems such as MAXSAT and spin glasses [Pelikan, Pelikan, Goldberg, & GoldbergPelikan et al.2003]. Also, one may think of several possible improvements, such as modifying the starting position of ILS, and we would like to investigate them. Finally, it is important to analyze DSMGA-II from the theoretical perspective to fully understand its strength and weakness.

References

Alvarez-Miranda, E., Ljubic, I., & Mutzel, P. (2013). The maximum weight connected subgraph problem. In *Facets of Combinatorial Optimization* (pp. 245–270). Springer Berlin Heidelberg.

Bosman, P. A., & Thierens, D. (2012). Linkage neighbors, optimal mixing and forced improvements in genetic algorithms. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12 (pp. 585–592). New York, NY, USA: ACM.

Bosman, P. A. N., & Thierens, D. (2013). More concise and robust linkage learning by filtering and combining linkage hierarchies. In Blum, C., & Alba, E. (Eds.), *GECCO* (pp. 359–366). ACM.

Chen, W.-M., Hsu, C.-Y., Yu, T.-L., & Chien, W.-C. (2013). Effects of discrete hill climbing on model building forestimation of distribution algorithms. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13 (pp. 367–374). New York, NY, USA: ACM.

Deb, K., & Goldberg, D. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, *10*(4), 385–408.

Goldberg, D. E., Deb, K., & Clark, J. H. (1991). Genetic algorithms, noise, and the sizing of populations. *COMPLEX SYSTEMS*, *6*, 333–362.

Goldberg, D. E., Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms.

Harik, G., & Harik, G. (1999). *Linkage learning via probabilistic modeling in the ecga* (Technical Report).

Holland, J. H. (1992). *Adaptation in natural and artificial systems.* Cambridge, MA, USA: MIT Press.

Kullback, S., & Leibler, R. A. (1951, 03). On information and sufficiency. *Ann. Math. Statist.*, *22*(1), 79–86.

Lee, H. F., & Dooly, D. R. (1998). Decomposition algorithms for the maximum-weight connected graph problem. *Naval Research Logistics (NRL)*, *45*(8), 817–837.

Luong, N. H., La Poutré, H., & Bosman, P. A. (2014). Multi-objective gene-pool optimal mixing evolutionary algorithms. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14 (pp. 357–364). New York, NY, USA: ACM.

Martins, J. P., Fonseca, C. M., & Delbem, A. C. (2014). On the performance of linkage-tree genetic algorithms for the multidimensional knapsack problem. *Neurocomputing*, *146*(0), 17 – 29. Bridging Machine learning and Evolutionary Computation (BMLEC) Computational Collective Intelligence.

Pelikan, M. (2005). Hierarchical bayesian optimization algorithm. In *Hierarchical Bayesian Optimization Algorithm* (pp. 105–129). Springer Berlin Heidelberg.

Pelikan, M., Pelikan, M., Goldberg, D. E., & Goldberg, D. E. (2003). Hierarchical boa solves ising spin glasses and maxsat. In *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2003), number 2724 in LNCS* (pp. 1271–1282). Springer.

Pelikan, M., Sastry, K., Goldberg, D. E., Butz, M. V., & Hauschild, M. (2009). Performance of evolutionary algorithms on nk landscapes with nearest neighbor interactions and tunable overlap. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09 (pp. 851–858). New York, NY, USA: ACM.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*(5), 465 – 471.

Thierens, D. (2010). The linkage tree genetic algorithm. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part I*, PPSN'10 (pp. 264–273). Berlin, Heidelberg: Springer-Verlag.

Thierens, D., & Bosman, P. A. (2011). Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11 (pp. 617–624). New York, NY, USA: ACM.

Thierens, D., & Bosman, P. A. (2013). Hierarchical problem solving with the linkage tree genetic algorithm. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13 (pp. 877–884). New York, NY, USA: ACM.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. *Urbana*, *51*, 61801.

Wang, S.-M., Tung, Y.-F., & Yu, T.-L. (2014, July). Investigation on efficiency of optimal mixing on various linkage sets. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (pp. 2475–2482).

Yu, T.-L. (2006). *A matrix approach for finding extrema: Problems with modularity, hierarchy, and overlap.* Doctoral dissertation, Champaign, IL, USA. AAI3250351.

Yu, T.-L., & Goldberg, D. E. (2006). Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06 (pp. 1385–1392). New York, NY, USA: ACM.

Yu, T.-L., Sastry, K., & Goldberg, D. E. (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 (pp. 1217–1224). New York, NY, USA: ACM.

Yu, T.-L., Sastry, K., Goldberg, D. E., & Pelikan, M. (2007). Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07 (pp. 601–608). New York, NY, USA: ACM.