2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Investigation on Efficiency of Optimal Mixing on Various Linkage Sets

Shih-Ming Wang
Department of Electrical Engineering
National Taiwan University
Email: r01921055@ntu.edu.tw

Yu-Fan Tung
Department of Electrical Engineering
National Taiwan University
Email: r02921044@ntu.edu.tw

Tian-Li Yu
Department of Electrical Engineering
National Taiwan University
Email: tianliyu@ntu.edu.tw

***Abstract*—The optimal mixing operator (OM) utilizes linkage sets (LSs) to exchange the information of variables between a pair of solutions, and the result of such exchange is adopted only if the exchange leads to improvement of the solution quality. The performance of OM highly depends on the LS it uses. This paper demonstrates that previously proposed LS, the linkage tree model (LT), does not yield the optimal performance. To measure the efficiency of OM on different LSs, the cost-performance (CP) index is defined. Both our CP index and experiments indicate (1) that for fully separable problems, the most suitable LS is the marginal product model (MP), and (2) that for separable problems with overlap, LT is more suitable than MP, and (3) that properly pruned LT leads to higher efficiency and yields a better performance, and (4) that the LS that properly reflects the problem structure yields the best performance on both fully separable problems and problems with overlap.**

I. Introduction

One of the key properties of estimation of distribution algorithms (EDAs) is their abilities to juxtapose partial solutions that jointly represent an important contribution to the solution quality. In EDAs, a probabilistic model is built and a corresponding probability distribution is sampled to generate new solutions [9], [12]. For model building, selection in EDAs reduce noise by filtering the population. However, selection is typically not a very strong process unless the selection intensity is high which drives up the required population size to ensure sufficient diversity for the multi-generational search.

Local search techniques combined with evolution algorithms [1], [13], on the other hand, strongly reduce noise while keeping the diversity as long as different local optima are found. Optimal mixing (OM) [1], [16] introduced after the linkage tree genetic algorithm (LTGA) [11], [15] performs stochastic local search on the learned model. For OM, the learned model is a set of masks, each of which consists of several indices of variables. In this paper, we call such model linkage set (LS), which is formally defined in section III. To generate a new solution, OM starts from a parent and utilizes each mask to donate variables to the parent from random selected parent(s). Each donation is adopted only if the donation leads to improvement of the solution quality. The order of utilizing the masks affects the result and should be predetermined. Since donation is accepted only when improvement happens, the mixing operator of OM is like local search. Such property of OM reduces required population. However, it comes with the price of a higher possibility of convergence of the population to local optima.

The performance of OM depends on the LS it learned. OM can be combined with various LSs such as the marginal product model or the linkage tree model [15]. The linkage tree model consists of nested masks and is robust when adopted with OM to solve problems difficult for genetic algorithms (GAs). However, adopting nested masks is less likely to gain continuous improvement of the solution quality due to the local search property of OM. Donations without improvement lead to waste of function evaluations. Hence, robust LSs might suffer from inefficiency. In this paper, we analyze the efficiencies of applying OM with different LSs to different problems. From the experiment results, we observe that previously proposed LSs do not achieve optimal performances on all problems and we derive general rules for more suitable LSs for different problems. The results indicate that some efforts could be done to strike a balance between the efficiency and the robustness of LSs.

This paper continues as follows. In Section II, we describe OM in detail. In Section III , we state our assumptions on the problems and describe two proposed LSs and their limitations. In Section IV , the definition of LS efficiency is given. In Sections V and VI, we adopt different LSs on two types of problems, fully separable problems and problems with overlap. Some general rules for choosing LS are derived from the experiment results. Finally, we conclude with discussions and possible future work.

II. Optimal Mixing

In this section, we briefly introduce OM. We rewrite some algorithms and borrow some notations from [16]. In all the algorithms, we denote the population and the offspring by P and O respectively. We also assume $|O| = |P| = n$, where $|S|$ denotes the number of elements in the set S. The i^{th} chromosome is denoted by P^i, and P^i_M represents a part of P^i, where M is a mask describing the indices of the part. For example, let $M = \{1, 3, 9\}$, P^5_M represents the 1^{st}, 3^{rd} and 9^{th} variables of the 5^{th} chromosome.

The common EDA framework is outlined in Algorithm 1. LearnModel produces a model which is utilized by MixSolution. OM adopts OptimalMixing as its MixSolution which take a LS as its model. For OptimalMixing shown in Algorithm 2, every chromosome in the parent pool is picked as an receiver once. The receiver is cloned. For every mask in the LS, a donor is chosen and some variables of the donor are donated into the clone according to the mask. If the donation makes no change to the clone, it is skipped. Otherwise

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

Algorithm 1: EDA

```
for i ← 1 to n do
    P_i ← CreateRandomSolution()
    EvaluateFitness(P_i)
while ¬TerminationConditionSatisfied() do
    P' ← Selection(P)
    M ← LearnModel(P')
    O ← MixSolution(P', M)
    P ← Mutation(O)
```

the clone is reevaluated. If the fitness of the clone after the donation is higher than that of the receiver, the receiver is replaced by the clone. The final clone after all the masks have been adopted is the offspring of the next generation.

For most EDAs, the reevaluations of the chromosomes is required only in SELECTION while MIXSOLUTION requires no extra function evaluations. As for OM, the reevaluations of the chromosomes are required in MIXSOLUTION and the first run of SELECTION. No more reevaluation is needed in the following runs of SELECTION since the chromosomes are already evaluated in MIXSOLUTION. Therefore, when analyzing OM, we focus on the efficiency of the mixing operator.

Recombinative optimal mixing evolutionary algorithm (ROMEA) and gene-pool optimal mixing evolutionary algorithm (GOMEA) are two different OMs. ROMEA randomly chooses a donor and uses it for all the masks while GOMEA randomly chooses a donor for each mask. The two different methods are shown in Algorithms 3 and 4. GOMEA is similar to sampling a probability distribution, which is reminiscent of EDA. Experiments [16] show that GOMEA outperforms ROMEA. However, to analyze the efficiency of sequentially adopted masks that might overlap, this paper focus on analyzing ROMEA.

Algorithm 2: OptimalMixing(P, M)

```
for i ← 1 to n do
    D ← GenerateDonorIndex(|M|, n)
    for j ← 1 to |M| do
        if (P^i_{M_j} ≠ P^{D_j}_{M_j}) then
            P^i ←Mix(P^i, P^{D_j}, M_j)
    O_i ← P_i
return O
FUNCTION Mix(P^0, P^1, m) begin
    C ← P^0
    C_m ← P^1_m
    EvaluateFitness(C)
    if fitness[C] ≥ fitness[P^0] then
        P^0 ← C
        fitness[P^0] = fitness[C]
    return P^0
```

III. DEFINITION AND NOTATION

In this section, we state the assumptions on the problems and describe two proposed LSs and their limitations.

Algorithm 3: GOMEA::GenerateDonorIndex(m, n)

```
Initialize integer Array D[m]
for i ← 1 to m do
    D[i] = RandomNumber(1, n)
return D
```

Algorithm 4: ROMEA::GenerateDonorIndex(m, n)

```
Initialize integer Array D[m]
d ← RandomNumber(1, n)
for i ← 1 to m do
    D[i] = d
return D
```

A. Problem Definition

According to the building block hypothesis [3], we consider two types of nearly-fully separable problems in this paper, fully separable problems and separable problems with overlap.

Definition 3.1: A *fully separable problem* is a problem whose fitness function can be written as a summation of a series of sub-functions, and the parameter list for each sub-function is independent.

Definition 3.2: A *problem with overlap* is a problem whose fitness function can be written as a summation of a series of sub-functions, but the parameter list for each sub-function might share some common variables.

In this paper, we consider carefully designed sub-functions that make the parameters of each sub-function form strongly connected group such that EDAs should treat them together in the mixing phase to solve the problem. A group of such variables are called a building block (BB).

Even for problems with overlap, the BBs can be grouped into several independent chunks such that each chunk share no common variables. The synthetic function for each chunk is the summation of the sub-functions for all BBs of the chunk. For simplicity of analyzing, we consider only homogeneous problems with identical chunks.

Definition 3.3: A *homogeneous problem* is a separable problem that can be divided into groups of independent chunks, and the synthetic functions of each chunk are identical.

In the experiments, we consider two homogeneous, binary encoded fully separable problems, the one-max problem and the m-k trap problem. The one-max problem is composed of independent bits, and the fitness function is defined as:

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

where ℓ is the length of the solution. The m-k trap problem is defined as:

$$f_{trap}(\vec{x}) = \sum_{i=0}^{m-1} trap_k(u(y_{ik+1}, y_{ik+2} \cdots, y_{ik+k})), \tag{2}$$

where $\vec{y} = (y_1, y_2 \cdots, y_\ell)$ is a random permutation of $\vec{x}$ and $u(.)$ is the unitary function that counts number of ones in a

bit string, and $trap_k(.)$ is a deceptive function [2] which leads GAs toward converging to its sub-optimal solution. The $trap_k$ function is defined as:

$$trap_k(u) = \begin{cases} 1 & u = k \\ \frac{0.9(k-u)}{k} & otherwise \end{cases}, \quad (3)$$

As for problems with overlap, we design a problem called aggregate trap (atrap) problem. The atrap problem is composed of some basic BBs of size k. The function of Each BB is the trap function. Two BBs share 1 bit and form a chunk. In the experiments, we have $k = 5$. The atrap problem is defined as:

$$f_{atrap}(\vec{x}) = \sum_{i=0}^{\frac{m}{2}} (trap_k(u(y_{2ik+1}, y_{2ik+2} \cdots, y_{2ik+k})) + trap_k(u(y_{2ik+k}, y_{2ik+k+1} \cdots, y_{2ik+2k-1}))), \quad (4)$$

where m is an even number representing the number of BBs.

B. Linkage Set Definition

EDAs model problem structures with different linkage models. For example, cGA [8], one of the simplest EDAs, uses the probability vector as its linkage model, and BOA [10], one of the most powerful EDAs, uses Bayesian network to model problem structures. In this paper, we focus on OM which use the linkage set as its linkage model. Suppose the problem is of ℓ variables. Let the indices of the problem variables be $V = \{1, 2, \cdots, \ell\}$. We define mask and linkage set as follows.

Definition 3.4: A *mask* is a subset of V. A *linkage set* $\mathcal{M}$ is a set of masks. $\mathcal{M} = \{M_1, M_2, \ldots, M_{|\mathcal{M}|}\}$ such that $M_1 \cup M_2 \cup \ldots \cup M_{|\mathcal{M}|} = V$.

In practice, what OM needs is a sequence of masks instead of a set of masks. Hence, we need to specify the order of the masks. In this paper, the elements of a set of masks are written in the order with which OM utilizes them.

Since we consider homogeneous problems, we use some specific homogeneous LSs in this paper. A homogeneous LS can be divided into some isomorphic parts, each containing some masks.

Definition 3.5: A *homogeneous LS* is a LS $\mathcal{M}$ such that there exists a partition $\{\mathbb{M}_1, \mathbb{M}_2, \ldots\}$ of $\mathcal{M}$ and $\mathbb{M}_1, \mathbb{M}_2, \ldots$ are all isomorphic.

For example, let $\mathbb{M}_1 = \{\{1\}, \{2, 3\}\}$ and $\mathbb{M}_2 = \{\{4\}, \{5, 6\}\}$ be two set of masks. $\mathbb{M}_1$ and $\mathbb{M}_2$ are then disjointed and isomorphic. Formally, let S and T be be two subsets of V and let $\mathbb{M}_S$ and $\mathbb{M}_T$ be two set of masks covering variables in S and T correspondingly. $\mathbb{M}_S$ and $\mathbb{M}_T$ are isomorphic if there is a mapping from S to T such that one can transform $\mathbb{M}_S$ to $\mathbb{M}_T$ by applying the mapping to the elements of S in $\mathbb{M}_S$.

C. Marginal Product Model

Definition 3.6: The *marginal product model* (MP) is a partition of V.

For example, $\{\{1, 2, 3\}, \{4\}, \{5, 6\}\}$ is a MP for the problem with 6 variables. In MP, all problem variables indices are contained in exactly one mask. Any two different masks are disjoint. In this paper, we consider two specific homogeneous MP. The (m, k)-MP, and the $(\ell, 1)$-MP.

Definition 3.7: A (m, k)-*MP* is a homogeneous MP composed of m masks, each of size k, and $k = \frac{\ell}{m}$, where ℓ is the number of problem variables. A $(\ell, 1)$-*MP* is a homogeneous MP composed of ℓ masks, each of size 1.

For example, $\{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$ is the $(3, 2)$-MP and $\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}$ is the $(6, 1)$-MP.

D. Linkage Tree Model

Definition 3.8: A *binary tree set* $\mathbb{M}_S$ is a set of masks covering the variables in S, a subset of V. $\mathbb{M}_S = \{M_1, M_2, \ldots, M_{|\mathbb{M}_S|}\}$ such that $\forall M_i \in \mathbb{M}_s, (|M_i| \neq 1) \Rightarrow (\exists! j \exists! k, j \neq k, M_j \neq \emptyset, M_k \neq \emptyset, M_j \cap M_k = \emptyset, M_j \cup M_k = M_i)$ [1].

For example, $\{\{1, 2, 3\}, \{1\}, \{2, 3\}, \{2\}, \{3\}\}$ is a binary tree set.

Definition 3.9: The *linkage tree model* (LT) $\mathcal{M}_{LT}$ is a binary tree set of V such that $\exists! M_i \exists! M_j \in \mathcal{M}_{LT}, M_i \neq M_j, M_i \cup M_j = V$.

The examples of LT are shown in Table I. LT is a binary tree LS. It can be generated by a hierarchical clustering procedure. The clustering starts from the $(\ell, 1)$-MP, which means that each problem variable is independent at first. Then, two masks are combined according to the marginal product metric. Both the combined mask as well as its constituent masks are in LT. The combining of masks continues until only two masks remain that together contain all the problem variables. In this paper, we analyze a pruned version of LT, the (m, k)-LT, which is defined as follows.

Definition 3.10: The (m, k)-*LT* is a homogeneous LS which is the union of m independent and isomorphic binary tree sets, each of which contains some masks and the largest mask is of size k.

The examples of the (m, k)-LT are shown in Table I. The (m, k)-LT is similar to the (m, k)-MP but each mask of the (m, k)-MP is split into hierarchical sub-masks. In our settings, the (m, k)-LT is generated as follows. Starting with all masks of the (m, k)-MP in a queue, add the first mask of the queue to the (m, k)-LT and then separate it into two sub- masks (skip if it is of size 1) and add the two sub-masks back to the queue. The process stops when the queue is empty. If during the process of constructing the (m, k)-LT, the sizes of two separated sub-masks differ at most 1, we call it the balanced (m, k)-LT.

For LT and the (m, k)-LT, the order with which OM utilizes the masks affects the performance. LTGA utilizes the masks in LT in the reverse order of masks creation which is generally a top-down order. In this paper, we consider both top-down order and bottom-up order for LSs with hierarchical structures.

E. Linkage Set

All masks in MP are disjoint while LT and the (m, k)- LT might contain overlapping masks. However, their abilities to

[1] $\exists!$ indicates that the following variable exists and is unique

v={1,2,3,4,5,6}	
(m,k)-MP	$(\ell,1)$-MP
$\{\{1,2,3\},\{4,5,6\}\}$	$\{\{1\},\{2\},\{3\},\{4\},\{5\},6\}\}$
top-down LT	bottom-up LT
$\{\{1,2,3,4\},\{1,2\},\{1\},\{2\},$ $\{3,4\},\{3\},\{4\},\{5,6\},\{5\},\{6\}\}$	$\{\{1\},\{2\},\{1,2\},\{3\},\{4\},\{3,4\},$ $\{1,2,3,4\},\{5\},\{6\},\{5,6\}\}$
top-down (m,k)-LT	bottom-up (m,k)-LT
$\{\{1,2,3\},\{1\},\{2,3\},\{2\},\{3\},$ $\{4,5,6\},\{4\},\{5,6\},\{5\},\{6\}\}$	$\{\{1\},\{2\},\{3\},\{2,3\},\{1,2,3\},$ $\{4\},\{5\},\{6\},\{5,6\},\{4,5,6\}\}$

TABLE I. EXAMPLES FOR THE SIX LSS

describe overlapping structures are limited. For example, $\{\{$ 1 , 2 , 3 $\}$, $\{$ 3 , 4 , 5 $\}\}$ is a LS but it can not be represented by any LT. Moreover, both LT and the (m,k)-LT require masks covering single variable, which results in many small masks and might lead to inefficiency.

In this paper, we try to figure out the most suitable LSs for different problems. A naive guess is that a good LS represents the problem structure and contains masks corresponding to the BBs of the problem. The $(\ell,1)$-MP and the (m,k)-MP represent the structures of the one-max problem and the $m-k$ trap problem, which we describe later in Section V. And the two LSs indeed outperform the other LSs. Hence, we believe that LSs that can represent the problem structure are suitable LSs. Throughout, we derive some general rules for suitable LSs for different problems.

IV. MODEL EFFICIENCY

While OM succeeds by reducing population noise and combining promising sub-solutions. The efficiency of OM depends on the LS it utilizes. The more masks the LS contains, the more number of function evaluations (NFEs) are required. For example, adopting nested masks such as LT is less likely to gain continuous improvement.

A. Definition of Model Efficiency

As the first step, we analyze the efficiencies of homogeneous LSs on homogeneous problems. A homogeneous LS can be partitioned into several parts. Each part contains several masks and different parts share no common variables. If all masks in one part cover only some variables of one BB, every time OM mixes variables with a mask, only one BB is affected. Hence, we can focus on how OM affects the BBs' quality instead of the total solution quality.

Suppose the variables are discrete, and each of them has ν possible values. A BB of size k has ν^k possible schemata, each with a corresponding fitness. We define the rank of each schema as follows.

Definition 4.1: The rank of a schema s is the number of possible fitness values that are smaller than the fitness value of s.

As an example, the rank of schemata of the 3-bit trap function is listed in Table II.

Consider applying OM to a pair of receiving and donating BB with a set of masks $\mathbb{M}=\{M_1,M_2,\ldots\}$, the improvement of the receiving BB quality could be quantified by the increase of its fitness rank. The required NFEs is smaller or equal to $|\mathbb{M}|$ since no function evaluation is required for some masks

Schemata	fitness	rank
000	0.9	2
001,010,101	0.6	1
011,101,110	0	0
111	1	3

TABLE II. RANKS OF THE SCHEMATA OF THE 3-BIT TRAP FUNCTION.

r : 000 d : 111		
$\mathbb{M}$	COST$(r,d,\mathbb{M})$	GAIN$(r,d,\mathbb{M})$
$\{\{1,2,3\}\}$	1	1
$\{\{1\},\{2,3\},\{1,2,3\}\}$	2	1

TABLE III. EXAMPLES OF THE COST AND GAIN FUNCTION. APPLYING DIFFERENT MASKS ON THE TWO SCHEMATA OF THE 3-BIT TRAP FUNCTION MIGHT RESULTS IN DIFFERENT COST AND GAIN. NOTICE THAT THE MASKS ARE ADOPTED IN THE ORDER AS THEY ARE WRITTEN.

if the donated bits are equal to the original ones. Hence, we define GAIN$(r,d,\mathbb{M})$ as the function of the rank improvement and COST$(r,d,\mathbb{M})$ as the function of the required NFEs when donating variables from a BB of schemata d to a BB of schemata r with a set of masks $\mathbb{M}$. For all r, d, $\mathbb{M}$, both GAIN$(r,d,\mathbb{M})$ and COST$(r,d,\mathbb{M})$ can be pre-calculated and recorded in a table. Table III shows the cost and gain for applying different masks on two schemata.

Let R,D denote random variables of schemata and $\mathbb{M}$ be one independent part of a homogeneous LS. We define cost-performance (CP) index of $\mathbb{M}$:

$$CP(\mathbb{M})=\frac{E[\text{GAIN}(R,D,\mathbb{M})]}{E[\text{COST}(R,D,\mathbb{M})]}. \tag{5}$$

$CP(\mathbb{M})$ summarizes the overall rank increase for NFEs cost by $\mathbb{M}$. A higher CP value indicates a more efficiency set of masks. Although CP is defined on an independent part of a LS, the efficiency of a LS is the same as the efficiency of its separable isomorphic parts. Note that when donating variables to an receiver with all the masks, GOMEA might choose different donors while ROMEA always choose the same donor. Hence, the definition of CP is more suitable for ROMEA.

B. Dual CP Value

We can calculate the CP values when running real EDAs by recording the distribution of all schemata in each generation. Then we can compare the efficiencies of different LSs by the CP values. However, it would be better if we could directly prove that one LS is always less efficient than another LS without running EDAs. For this purpose, we introduce the concept of dual CP.

Consider mixing a pair of BBs with schemata r, d with a set of masks $\mathbb{M}$. The probabilities of choosing (r,d) or (d,r) as $(\text{receiver},\text{donor})$ pair are both $Pr[r]Pr[d]$, where $Pr[*]$ is the ratio of the corresponding schemata in the population. Hence we define

$$CP_{dual}(r,d,\mathbb{M})=\frac{\text{GAIN}(r,d,\mathbb{M})+\text{GAIN}(d,r,\mathbb{M})}{\text{COST}(r,d,\mathbb{M})+\text{COST}(d,r,\mathbb{M})}, \tag{6}$$

where the probability $Pr[r]Pr[d]$ is canceled from both the numerator and the denominator. Note that dual CP value is only valid when COST$(r,d,\mathbb{M})$+COST$(d,r,\mathbb{M})>0$. Hence a dual pair (r,d) is a legal dual pair only if COST$(r,d,\mathbb{M})$+COST$(d,r,\mathbb{M})>0$.

The dual CP enables us to inspect the efficiency of a LS on each pair of schemata. The following lemma describes the relation of CP and dual CP:

Lemma 4.2: For a set of masks $\mathbb{M}$, if for all legal dual pairs r, d, $CP_{dual}(r, d, \mathbb{M}) < c$ where c is a constant, then $CP(\mathbb{M}) < c$.

Proof: Let D denote number of legal dual pairs and g_i, c_i, p_i denote the numerator, the denominator and the ratio of the i^{th} legal dual pair. We need to prove that if $g_i/c_i < c$ for i from 1 to D, then $(\sum_{i=1}^{D} p_i g_i / \sum_{i=1}^{D} p_i c_i) < c$. By observing that $g_i < c \times c_i$ for all i, we get $\sum_{i=1}^{D} p_i g_i < c \sum_{i=1}^{D} p_i c_i$. Hence, $(\sum_{i=1}^{D} p_i g_i / \sum_{i=1}^{D} p_i c_i) < c$. ■

In some cases, we could use Lemma 4.2 to prove that one LS is less efficient than another by inspecting the dual CP values of all legal dual pairs.

C. Population Sizing

Model efficiency could be regarded as the rate of population converging. However, the total NFEs is still affected by the population size. Some facet-wise models, such as initial-supply [5], decision making [4], [6] and model building [17], have been developed to model different bounds on population sizing for EDA success.

For OM, recognized BBs are compared separately, so fitness variance of other BBs do not affect the decision of choosing better BBs. Therefore, the population size is more possible to be bounded by initial-supply or model building. In this paper, to compare efficiencies of different LSs, we run EDAs with perfect model information. In other words, we do not consider model building.

As a facet-wise model, LS efficiency is not a direct predicator of total performance. Throughout, we discuss LS selection by considering both LS efficiency and population sizing although we only describe the latter quantitatively.

V. Analysis on Fully Separable Problems

By applying different LSs to different problems, we would like to see which LS is more suitable for each problem. Moreover, we would like to see if there are general rules of LS choosing. In this section, we analyze the onemax problem and the m-k trap problem.

A. Experiments Setting

In all our experiments, instead of learning LS from the population, the LSs required by OM are given manually. In the section, we test the m-k trap problem and the one-max problem with four LS, the $(\ell, 1)$-MP, the (m, k)-MP, the top-down (m, k)-LT and the bottom-up (m, k)-LT. In all the experiments, we use the binary tournament selection [7], full replacement as EDA operators. No mutation operator is used, and the termination condition is convergence of the population. If not specified explicitly, unbiased initial population is used.

Model	Required population	Required NFEs
$(\ell, 1)$-MP	316*	101497
(m, k)-MP, $k = 5$	316	49982
top-down (m, k)-LT, $k = 5$	264	129300
bottom-up (m, k)-LT, $k = 5$	179	133868

TABLE IV. Summary of applying the four LSs to a 180-bits m-k trap problem with $k = 5$. Notice that the $(\ell, 1)$-MP LS fails to solve the problem.

B. Applying Models on the Trap Problem

The result of applying the four LSs to the m-k trap problem is shown in Table IV and Figure 1. For the (m, k)-MP, we have k equal to the size of the ground truth BBs. For the (m, k)-LT, we consider the balanced (m, k)-LT. The required population is derived by running the bisection [14]. The population fails to converge to the optimum solution with $(\ell, 1)$-MP, so we use the same population size required by adopting the (m, k)-MP.

Since higher CP values indicate higher efficiency, Figure 1 shows that the (m, k)-MP is always more efficient than the other three LSs. The $(\ell, 1)$-MP leads to nearly no improvement at the end of the process. This is because most of the schemata are 00000 and 11111 at the late phase, each of which is the local optimum and the global optimum of the BBs. Both injecting a bit from 11111 to 00000 and injecting from 00000 to 11111 lead to no rank increase and thus all evaluations are wasted.

The (m, k)-LT also solves the trap problem, however, it is less efficient than the (m, k)-MP. However, the (m, k)-LT might still be more efficient than the (m, k)-MP in some cases. We could exhaustively list all possible functions and check the efficiencies of both LSs. For example, for $k = 3$, there are 8 possible schemata for a BB. Hence there are $8!$ possible functions if we consider only the ranks of fitness. Experiment results show that for all the $8!$ functions, the (m, k)-LT is always less efficient than the (m, k)-MP.

Note that the bottom-up (m, k)-LT is less efficient than the top-down (m, k)-LT since the former leads to $0^{(5)}$, the local optima of a BB, more frequently. However, we shall later see that bottom-up structure is usually a better choice if the masks represent the problem structure.

By the comparison of the four LSs, we conclude that for strongly connected variables that form a BB, we should have a mask that cover all the variables of the BB and we should avoid dividing that mask into sub-masks for efficiency concern.

C. Applying Models on One-max Problem

Next, we apply the LSs to the one-max problem. We use a biased initial population such that each bit is set to 0 with probability 0.9. For the (m, k)-MP and the (m, k)-LT, larger population size is required if we fix masks in every generation due to BB supply [5]. However, if a model builder is used to solve the one-max problem, the masks are not fixed in every generation. Thus, to simulate real model building, we random partition the problem variables into m chunks and let each independent part of the two LSs generate masks corresponding to one chunk.

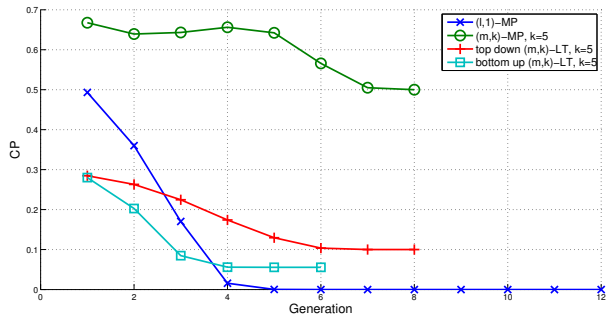

Fig. 1. CP values of different LSs to the m-k trap problem. Note that the convergence time are not all the same for the LSs.

As we see in Figure 2, the (m, k)-MP is always more efficient than the $(\ell, 1)$-MP. Moreover, the lager k is, the more efficient the (m, k)-MP is. For the MP series LSs, the cost of no improvement trial is always 1 while for larger k, the expectation gain is greater. Hence, efficiency of a MP is positive relative to k. However, from Table V we see that as k increases, the required population size increases accordingly, which results in higher NFEs. The population increases is like the punishment of linkage model complexity. For the one-max problem, it seems that concerning both model efficiency and model punishment, the most suitable MP is the $(\ell, 1)$-MP. However, if same population is used, a MP with larger k leads to fewer NFEs since it is more efficient.

The result of the one-max problem gives us some insight into more complex problem. For example, for the m-k trap problem, by treating optimal BBs as 1s and non-optimal BBs as 0s, it is then similar to the one-max problem. The initial ratio of the optimal BBs would be about $1/2^k$ if unbiased population is used. Therefore, we could extend our conclusion that to achieve optimal performance, we should use a LS with each mask containing the variables of one BB, and the masks should not be combined to form larger masks. By experiments, this extension is correct on the m-k trap problem.

Figure 2 indicates that the efficiencies of the top-down (m, k)-LT and the bottom-up (m, k)-LT are the lowest among all the LSs. The reason is that when a mask leads to improvement, the receiver is less likely to be improved again since it is of a better schemata. Hence, the probability that the other masks lead to improvement is lower. This implies that a tree structure might waste some NFEs. As we see in Table VI, for the top-down (m, k)-LT, there is no possible schemata pairs that leads to dual CP value larger than 0.5. By Lemma 4.1, we know the CP value of the top-down (m, k)-LT is always smaller than 0.5, which means the top-down (m, k)-LT is always less efficient than $(l, 1)$-MP. The same inference applies to the bottom-up (m, k)-LT.

Table V shows that the bottom-up (m, k)-LT requires relatively small NFEs comparing to the top-down (m, k)-LT due to the small population size it requires. Unlike the experiment for the trap problem, when each mask of the (m, k)-LT covers at least a BB of the problem, it is less possible to lead to a local optima by applying it. Hence, bottom-up structures are a better choice for LSs with tree structures if the masks represent the problem structures.

We summarize this section with three rules of LS selection for fully separable problems:

1) The most suitable LS should contain some masks, each covering a BB that is formed by strongly connected variables.
2) For each mask corresponding to a BB, further division of each mask leads to inefficiency and worse performance.
3) For each mask corresponding to a BB, further combination of each mask leads to higher model punishment and worse performance.

Model	Required population	Required NFEs
$(\ell, 1)$-MP	137	39356
(m, k)-MP, $k = 5$	181	45889
(m, k)-MP, $k = 9$	240	50013
top-down (m, k)-LT, $k = 5$	160	91393
top-down (m, k)-LT, $k = 9$	188	113440
bottom-up (m, k)-LT, $k = 5$	126	71393
bottom-up (m, k)-LT, $k = 9$	136	85139

TABLE V. SUMMARY OF APPLYING DIFFERENT LSS TO A 180-BITS ONE-MAX PROBLEM.

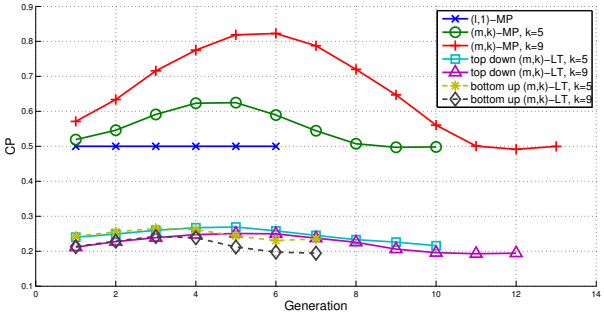

Fig. 2. CP values of applying different LSs to the one-max problem. Note that biased initial population is used to simulate the initial ratio of optimal BBs of complex problems.

	$(\ell, 1)$-MP	(m, k)-LT, $k = 5$	(m, k)-LT, $k = 9$
dual CP<0.5	0	9.98E-01	9.99E-01
dual CP$=0.5$	1	2.02E-03	7.64E-06
dual CP>0.5	0	0	0

TABLE VI. RATIO OF LEGAL DUAL PAIRS ON DIFFERENT CONDITION.

VI. ANALYSIS ON PROBLEMS WITH OVERLAP

Real world problems often have complex structures. By regarding strongly related variables as a BB, there might be overlap between BBs, which means some BBs share several common variables. A possible way of solving such problems is to combine two overlapping BBs together with a larger mask, which results in a larger population size. A better way might be using masks corresponding to the original two BBs and the larger BB as well. In this section, we analyze performances of different LSs to the designed atrap problem.

A. More General Linkage Sets

As stated, the abilities of MP, LT to describe overlap is limited. The problem structure of atrp can not be fully

represented by any of MP and LT. Hence, in the experiments, we compare the (m, k)-LT and some general LSs that describe the structure of the atrap problem.

From previous results, we know that small masks lead to inefficiency. Hence, we consider two properly pruned (m, k)-LT, the $(9, 5, 4)$-LS and the $(5, 4, 9)$-LS, where each number corresponds to the size of a mask. For this two LSs, one of the BB of an isomorphic part is not correctly covered by the mask of size 4. Hence, we further consider three LSs, the $(5, 5)$-LS, the $(9, 5, 5)$-LS, and the $(5, 5, 9)$-LS. The examples of the LSs are shown in Table VII. The order of the masks utilized by OM when calculating the CP values are as the order in the examples. However, to simulate real model building, for all these LSs, the order of the two smaller masks (of size 4 or 5) of each isomorphic part is randomly determined.

Underlying BBs: $\{1, 2, 3, 4, 5\}, \{5, 6, 7, 8, 9\}$	
$(5, 5)$-LS	
$\{1, 2, 3, 4, 5\}, \{5, 6, 7, 8, 9\}$	
$(9, 5, 4)$-LS	$(5, 4, 9)$-LS
$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}$	$\{1, 2, 3, 4, 5\}, \{6, 7, 8, 9\}$, $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$(9, 5^2)$-LS	$(5^2, 9)$-LS
$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $\{1, 2, 3, 4, 5\}\{5, 6, 7, 8, 9\}$	$\{1, 2, 3, 4, 5\}\{5, 6, 7, 8, 9\}$, $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

TABLE VII. EXAMPLES FOR THE GENERAL LSS. NOTICE THAT WE ONLY SHOW MASKS FOR AN ISOMORPHIC PART FOR THE LSS.

B. Applying Models on the Atrap Problem

The experiment results are shown in Table VIII and Figure 3. We observe (1) that $(5, 5)$-LS failed to solve the problem, and (2) that pruned LT is more efficient than (m, k)-LT and leads to better performance, and (3) that $(9, 5, 4)$-LS requires more NFEs than $(9, 5, 5)$-LS due to larger population size while the their efficiencies are close, and the result for $(5, 4, 9)$-LS and $(5, 5, 9)$-LS is the same, and (4) that for properly pruned LSs, bottom-up structure is better due to smaller population size.

From the observations, we summarize this section with the following rules for choosing LS for separable problems with overlap :

1) Combining masks corresponding to overlapping BBs is necessary and hence LT is more suitable than MP.
2) Properly pruned LT is more efficient than LT and hence is more suitable.
3) Considering both model efficiency and population sizing, LS reflecting the problem structure is more suitable than properly pruned LT
4) Bottom-up structure is a better choice for LSs reflecting the problem structure.

VII. CONCLUSION

We use the cost-performance index to measure the LS efficiency for OM. With some experiments, rules of an efficient LS are derived. In a LS, large mask usually leads to higher efficiency yet larger population size. On the other hand, smaller masks reduce population size while they usually cause inefficiency. Suppose the problem structure can be separated into several independent chunks and each chunk can not be

Model	Required population	Required NFEs
$(5, 5)$-LS	461*	339962
$(9, 5, 4)$-LS	509	115387
$(9, 5, 5)$-LS	461	106036
top-down (m, k)-LT, $k = 9$	470	288119
$(5, 4, 9)$-LS	371	101291
$(5, 5, 9)$-LS	337	91628
bottom-up (m, k)-LT, $k = 9$	322	295812

TABLE VIII. SUMMARY OF APPLYING THE SEVEN LSS TO A 180-BITS ATRAP PROBLEM WITH $k = 5$. NOTICE THAT $(5, 5)$-LS FAILS TO SOLVE THE PROBLEM.

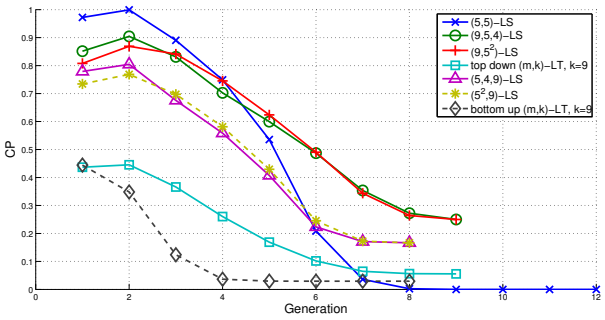

Fig. 3. CP values of applying different LSs to the one-max problem.

divided into non-overlapping sub-chunks. For each chunk, the LS should contain a mask corresponding to it, which we called a top mask. If all the variables of a chunk are all strongly related, the LS should contain no sub-masks of the corresponding top mask. In contrast, if a chunk can be further divided into some overlapping BBs, masks corresponding to each BB should be in the LS. To sum up, the LS that describes the problem structure leads to optimal performance.

One feature of OM is that it usually solves the problem with a smaller population than traditional EDAs. However, the NFEs are affected by the adopted LSs. Our investigation on the efficiency of LSs indicates that some efforts could be done to strike a balance between the efficiency and the robustness of the LSs. We believe that further study of population sizing and the efficiency of LSs is essential to advance the development of more competent mixing operators.

REFERENCES

[1] P. A. Bosman and D. Thierens. The roles of local search, model building and optimal mixing in evolutionary algorithms from a bbo perspective. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, pages 663–670, New York, NY, USA, 2011. ACM.

[2] K. Deb and D. E. Goldberg. Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2*, pages 93–108, 1993.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[4] D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, vol. 6:333–362, 1992.

[5] D. E. Goldberg, K. Sastry, and T. Latoza. On the supply of building blocks. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 336–342, 2001.

[6] G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 7–12, 1997.

[7] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In *ICGA*, pages 24–31, 1995.

[8] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 523–528, 1998.

[9] J. A. Lozano, P. Larrañaga, I. n. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. BOA: The bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 525–532, 1999.

[11] M. Pelikan, M. W. Hauschild, and D. Thierens. Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1005–1012, New York, NY, USA, 2011. ACM.

[12] M. Pelikan, K. Sastry, and E. Cantu-Paz, editors. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer, Berlin, 2006.

[13] E. Radetic, M. Pelikan, and D. E. Goldberg. Effects of a deterministic hill climber on hboa. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 437–444, New York, NY, USA, 2009. ACM.

[14] K. Sastry. Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

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

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

[17] T.-L. Yu, K. Sastry, D. E. Goldberg, and M. Pelikan. Population sizing for entropy-based model building in discrete estimation of distribution algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 601–608, 2007.