Optimal Mixing Evolutionary Algorithms

Dirk Thierens
Department of Computer Science
Utrecht University
3508 TB Utrecht
The Netherlands
Dirk.Thierens@cs.uu.nl

Peter A.N. Bosman
Centrum Wiskunde & Informatica (CWI)
P.O. Box 94079
1090 GB Amsterdam
The Netherlands
Peter.Bosman@cwi.nl

ABSTRACT

A key search mechanism in Evolutionary Algorithms is the mixing or juxtaposing of partial solutions present in the parent solutions. In this paper we look at the efficiency of mixing in genetic algorithms (GAs) and estimation-of-distribution algorithms (EDAs). We compute the mixing probabilities of two partial solutions and discuss the effect of the covariance build-up in GAs and EDas. Moreover, we propose two new Evolutionary Algorithms that maximize the juxtaposing of the partial solutions present in the parents: the Recombinative Optimal Mixing Evolutionary Algorithm (ROMEA) and the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA).

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Search.

General Terms

Algorithms, Performance, Experimentation.

Keywords

Genetic Algorithms, Estimation-of-Distribution Algorithms, Optimal Mixing, Linkage Tree Genetic Algorithm.

1. INTRODUCTION

A key property of GAs and EDAs is their ability to juxtapose partial solutions or substructures from different solutions to form new good solutions. This mixing is only efficient when the substructures are not disrupted too often by recombination or by the population sampling. When enough knowledge of the problem is available, mixing can be made efficient by designing the solution representation and/or the crossover operator or sampling distribution in an appropriate way. However, if there is not enough domain knowledge at hand, we have to induce this knowledge from a population of solutions. In this paper we first identify the differences between the mixing behavior of GAs and of EDAs. Next, we propose two algorithms - ROMEA and GOMEA - that have an optimal mixing efficiency.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

2. GA AND EDA MIXING

To study the mixing behavior of GAs and EDAs we focus on 4 substructures: *1*1*, *1*0*, *0*1*, and *0*0*. The mixing behavior is now characterized by computing the change in frequencies of these 4 substructures after applying a mixing operator: resp. uniform crossover and population sampling. Let us call p the frequency of substructure *1***, q the frequency of substructure ***1*, and r the frequency of substructure *1*1*. Note that $0 \leq r \leq \texttt{min}(p, q)$ The frequencies of the above 4 substructures are now given by:

P[*1*1*] = r	P[*1*0*] = p-r
P[*0*0*] = 1-p-q+r	P[*0*1*] = q-r

In the GA two solutions are paired and the substructures are either exchanged or not with equal probability. To compute the change in substructure frequencies we first need to compute the probability that specific substructures occur in a parent pair. The specific maiting pairs are shown in the first two columns (`Par1` and `Par2`) in Table 1. The probability a specific maiting pair will occur is given in the next column (`Prob`). Then, for a given pair we compute the probability with which each of the 4 substructures will be created. The results for each of the 4 substructures are shown in the columns `*1*1*`, `*0*0*`, `*1*0*`, and `*0*1*`. Finally, by summing all these weighted probabilities we obtain the expected frequencies of the 4 substructures after applying recombination. This end result is shown in Table 2. For instance, the first line in Table 2 shows that the frequency of the substructure *1*1* equals r before mixing. After uniform crossover (GA) is applied the frequency becomes $(pq + r)/2$, while after population sampling (EDA) is applied, the frequency becomes pq. In case of EDA sampling we can easily compute the expected frequencies directly, therefore there is no EDA column in Table 1.

Ideally, if the mixing is optimal then each mixing event would result with probability one in the creation of the *1*1* substructure (assuming that is the optimal substructure). In Section 5.4, we will introduce two algorithms, ROMEA and GOMEA, that achieve just that. Similar to GA, ROMEA pairs solutions before mixing them. The probability of a specific mating pair is thus the same as in the GA. So the second column in Table 1 (`Prob` under GA) also shows the mating probability for the pairs `Par1` and `Par2` for ROMEA. However, if the optimal values are present in the parent pair, the optimal substructure will always be created. Therefore,

the probability with which each of the 4 substructures will be created, is always 1 or 0. The results for each of the 4 substructures are shown again in the columns `*1*1*`, `*0*0*`, `*1*0*`, and `*0*1*`. GOMEA on the other hand is more similar to EDA as it uses gene-pool sampling. The last columns of Table 1 show the probability (`Prob`) of mating two specific substructures (`Par1` and `Par2`) when applying GOMEA.

The last 2 columns in Table 2 show the frequencies of the substructures after applying ROMEA, resp. GOMEA.

To compare GA versus EDA, and ROMEA versus GOMEA Figure 1 plots ratios between the frequencies of the *1*1* substructures for EDA vs. GA, and GOMEA vs. ROMEA in different scenarios. These frequencies are also shown on the top line of Table 2 but for ease of presentation, we assume here that the proportions of optimal values in both substructures are equal - this is, $p = q$. The plots are parameterized as EDA(p), GA(p, r), ROMEA(p,r), and GOMEA(p,r). The top left figure plots the ratio with $r = 0$, meaning that the substructure *1*1* is not yet present in the population. The ratio EDA(p)/GA(p,0) is always equal to 2, so the probability of a successful mixing event in this case is twice as high with EDA than with GA. In the top right figure, $r = p$, which represents the case that only the juxtaposed substructure *1*1* and *0*0* are present in the population. It is clear that the gene-pool sampling algorithms (EDA and GOMEA) now have a much higher probability of disrupting the juxtaposed substructures than the mating pairing algorithms (GA and ROMEA). Finally, the left and right bottom figures plot the *1*1* frequency as a function of r when $p = 0.1$. A pivotal point here is when $r = p^2 = 0.01$. This is the case of linkage equilibrium, where there is no covariance between the substructures. When $r < p^2$ the covariance becomes negative and the EDA (resp. GOMEA) have a higher probability to juxtapose the substructures than the GA (resp. ROMEA). However, when $r > p^2$ the covariance is positive - meaning that the frequency of the juxtaposed substructure has increased above the linkage equilibrium value - the GA (resp. ROMEA) have a higher probability of preserving the mixed substructures than the EDA (resp. GOMEA).

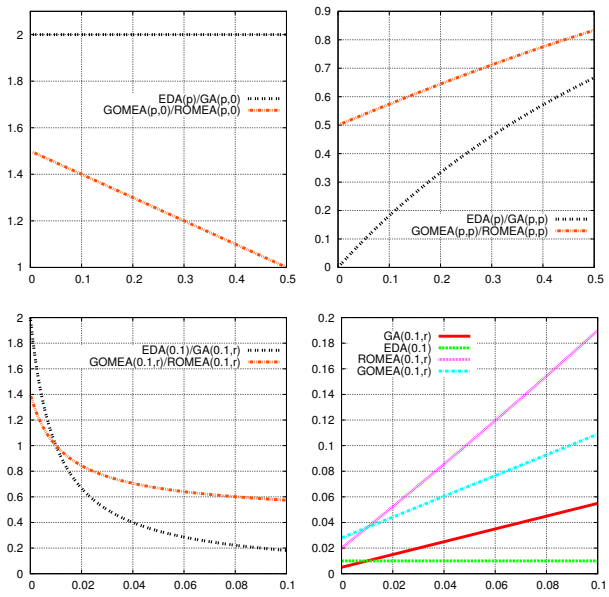

Figure 1: Ratios between frequencies of the juxtaposed substructure *1*1* (for details see Section 2).

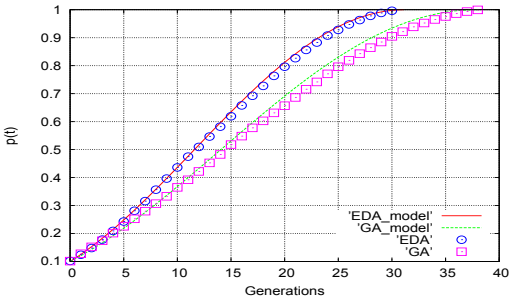

Figure 2: Convergence models and experimental results for EDA and GA on OneMax (l=100, Truncation threshold 50%, p(0)=0.1)

3. COVARIANCE BUILD-UP

Selection picks out the more fit individuals and thus reduces the variance of the population fitness. This reduction is not only caused by the change in allele frequency but also by the creation of negative covariances between the genes. In [6] we discussed how to model this for trunction selection when the fitness is binomially distributed as in the OneMax function. The population fitness variance can be computed by summing all elements from the genic covariance matrix: $\sigma^2(t) = \Sigma_{i=1}^{l}\Sigma_{j=1}^{l}\Theta_{ij}(t)p_{ij}(t) - p_i(t)p_i(t)$, with $p_i(t)$ the proportion of optimal values at position i. Separating the genic variance and covariance components $\sigma^2_{var}(t) = \Sigma_{i=1}^{l}\Theta_{ii}(t)$ and $\sigma_{cov}(t) = \Sigma_{i=1}^{l-1}\Sigma_{j=i+1}^{l}\Theta_{ij}(t)$ the population fitness variance becomes: $\sigma^2(t) = \sigma^2_{var}(t) + 2\sigma_{cov}(t)$.

Using truncation selection the selected parents represent one tail of the normal distribution. Truncating a normal distribution with variance σ^2 reduces this variance with a factor k such that $\sigma^2(t^s) = (1-k)\sigma^2(t)$, with $k = I(I-x)$, where I is the selection intensity and x is the corresponding deviation of the truncation point from the population mean.

The change in variance after selection is due to two phenomena: first the increase in optimal allele frequency $p(t)$, and second the creation of genic covariance. The allele-1 frequency increase in one generation however is rather small so the change in variance is primarily caused by the creation of the genic covariance. To make the analysis tractable we assume that we can neglect the effect of the allele frequency increase. Under this simplifying assumption the genic variance after selection $\sigma^2_{var}(t^s)$ can be approximated with the population fitness variance $\sigma^2(t)$ which allows us to calculate the genic covariance after selection $\sigma_{cov}(t^s)$ as: $\sigma^2(t^s) \approx \sigma^2_(t) + 2\sigma_{cov}(t^s) \approx (1-k)\sigma^2(t)$ or $\sigma_{cov}(t^s) \approx -\frac{k}{2}\sigma^2(t)$.

While selection introduces covariance between the gene values, crossover reduces this again with a decorrelation factor δ: $\sigma_{cov}(t+1) = \delta\sigma_{cov}(t)$. Crossover halves the covariance while EDA sampling reduces it to zero. To see this, recall that $\sigma_{ij} = E[x_i x_j] - E[x_i]E[x_j]$. Before crossover or EDA sampling we have $\sigma_{ij} = r - pq$, after crossover this becomes $\sigma_{ij} = (pq+r)/2 - pq = (r-pq)/2$, while after EDA sampling we get $\sigma_{ij} = pq - pq = 0$. This difference in reduction of the covariance causes the GA to mix less efficient than the EDA and therefore to require more generations until full convergence.

		GA					ROMEA				GOMEA				
Par1	**Par2**	**Prob.**	*1*1*	*0*0*	*1*0*	*0*1*	*1*1*	*0*0*	*1*0*	*0*1*	**Prob.**	*1*1*	*0*0*	*1*0*	*0*1*
*0*0*	*0*0*	$(1\text{-}p\text{-}q+r)^2$	0	1	0	0	0	1	0	0	(1+r-p-q)(1-p)(1-q)	0	1	0	0
*0*0*	*0*1*	(1-p-q+r)(q-r)	0	0.5	0	0.5	0	0	0	1	(1+r-p-q)(1-p)q	0	0	0	1
*0*0*	*1*0*	(1-p-q+r)(p-r)	0	0.5	0.5	0	0	0	1	0	(1+r-p-q)p(1-q)	0	0	1	0
*0*0*	*1*1*	(1-p-q+r) r	0.25	0.25	0.25	0.25	1	0	0	0	(1+r-p-q)pq	1	0	0	0
*0*1*	*0*0*	(q-r)(1-p-q+r)	0	0.5	0	0.5	0	0	0	1	(q-r)(1-p)(1-q)	0	0	0	1
*0*1*	*0*1*	$(q\text{-}r)^2$	0	0	0	1	0	0	0	1	(q-r)(1-p)q	0	0	0	1
*0*1*	*1*0*	(q-r)(p-r)	0.25	0.25	0.25	0.25	1	0	0	0	(q-r)p(1-q)	1	0	0	0
*0*1*	*1*1*	(q-r)r	0.5	0	0	0.5	1	0	0	0	(q-r)pq	1	0	0	0
*1*0*	*0*0*	(p-r)(1-p-q+r)	0	0.5	0.5	0	0	0	1	0	(p-r)(1-p)(1-q)	0	0	1	0
*1*0*	*0*1*	(p-r)(q-r)	0.25	0.25	0.25	0.25	1	0	0	0	(p-r)(1-p)q	1	0	0	0
*1*0*	*1*0*	$(p\text{-}r)^2$	0	0	1	0	0	0	1	0	(p-r)p(1-q)	0	0	1	0
*1*0*	*1*1*	(p-r)r	0.5	0	0.5	0	1	0	0	0	(p-r)pq	1	0	0	0
*1*1*	*0*0*	r(1-p-q+r)	0.25	0.25	0.25	0.25	1	0	0	0	r(1-p)(1-q)	1	0	0	0
*1*1*	*0*1*	r(q-r)	0.5	0	0	0.5	1	0	0	0	r(1-p)q	1	0	0	0
*1*1*	*1*0*	r(p-r)	0.5	0	0.5	0	1	0	0	0	rp(1-q)	1	0	0	0
*1*1*	*1*1*	r^2	1	0	0	0	1	0	0	0	rpq	1	0	0	0

Table 1: Frequencies of specific mating pairs and there mixing probabilities (for details see Section 2).

	Prob.	GA	EDA	ROMEA	GOMEA
*1*1*	r	(pq+r)/2	pq	$1+(1+r\text{-}p\text{-}q)^2\text{-}(1\text{-}p)^2\text{-}(1\text{-}q)^2$	pq(1-r)+r +p(q-r)(1-q)+q(p-r)(1-p)
*1*0*	p-r	p-(pq+r)/2	p(1-q)	$(1\text{-}q)^2\text{-}(1+r\text{-}p\text{-}q)^2$	(1-q)(p(1+r-p-q)-(r-p))
*0*1*	q-r	q-(pq+r)/2	(1-p)q	$(1\text{-}p)^2\text{-}(1+r\text{-}p\text{-}q)^2$	(1-p)(q(1+r-p-q)-(r-q))
*0*0*	1-p-q+r	1-p-q+(pq+r)/2	(1-p)(1-q)	$(1+r\text{-}p\text{-}q)^2$	(1+r-p-q)(1-p)(1-q)

Table 2: Frequencies of the substructures after different mixing algorithms (for details see Section 2).

Adding the covariance build-up to the convergence model of [6] we get: $p(t+1) - p(t) = \frac{I}{l}\sigma(t)\sqrt{1-k\delta}$. Solving this difference equation gives us the model:

$$p(t) = \frac{1}{2}(1 + \sin(\sqrt{\frac{1-k\delta}{l}}\ It + \arcsin(2p(0)-1))).$$

The number of generations g_{conv} to convergence can be computed by setting $p(g_{conv}) = 1$. The ratio of g_{conv} for GA ($\delta = 0.5$) versus EDA ($\delta = 0$) is equal to $\sqrt{1-k/2}$. For instance, for truncation selection with threshold 50% we have $I = 0.8$, $k = 0.64$, and $\sqrt{1-0.64/2} = 0.825$. The EDA thus converges about 20% faster than the GA as shown also experimentally in Figure 2.

4. STRUCTURES AND LEARNING

The idea of having a tunable model in an optimization algorithm to match the structure of the optimization problem most often comes down to identifying groups of variables that together make an important contribution to a solution's quality. Here we consider a general formulation for this type of dependency structure and describe three specific instances that we will use in our experiments.

4.1 Family of Subsets

The structures that we will use in this paper will all be a *family of subsets* (FOS), denoted $\mathcal{F}$. Mathematically, a FOS is a set of subsets of a certain main set S. A FOS is thus a subset of $\mathcal{P}(S)$, i.e. the powerset of S. In our case, the main set S is the set of all problem variable indices, i.e. $\{0, 1, \ldots, l-1\}$. A FOS $\mathcal{F}$ can be written as $\mathcal{F} = \{\boldsymbol{F}^0, \boldsymbol{F}^1, \ldots, \boldsymbol{F}^{|\mathcal{F}|-1}\}$ where $\boldsymbol{F}^i \subseteq \{0, 1, \ldots, l-1\}$, $i \in \{0, 1, \ldots, |\mathcal{F}|-1\}$. Furthermore, the structures we use will have families that are complete in the sense that every problem variable index is contained in *at least* one subset in $\mathcal{F}$, i.e. $\forall i \in \{0, 1, \ldots, l-1\} : \big(\exists j \in \{0, 1, \ldots, |\mathcal{F}|-1\} : i \in \boldsymbol{F}^j\big)$. In this manner, when performing variation following subsets in $\mathcal{F}$, no variable is left out of the process.

In EDAs [3] a probability distribution is estimated and new samples are drawn from this distribution. Typically these distributions have an underlying dependency structure similar to that of a FOS. If the subsets in the FOS are mutually exclusive (which is the case for the univariate structure and the marginal product structure below), a probability distribution can automatically be associated with the FOS, namely: $P(\boldsymbol{X}) = \prod_{i=0}^{|\mathcal{F}|-1} P(X_{\boldsymbol{F}^i})$ where a stochastic random variable X_i exists for each problem variable x_i.

4.2 Univariate Structure

The simplest structure is the univariate one, defined by $\boldsymbol{F}^i = \{i\}, i \in \{0, 1, \ldots, l-1\}$. There is only one configuration possible for this structure. Hence, no structure learning is required, causing variation operators based on this structure to have an extremely low computational complexity.

4.3 Marginal Product Structure

The marginal product (MP) structure is a FOS in which all subsets are mutually exclusive, i.e. each problem variable index is contained in *exactly* one subset. All variables in a single subset are then dependent on each other, whereas any two variables in different subsets are independent. Note that the univariate structure is a specific instance. The MP structure is defined by any FOS such that $\boldsymbol{F}^i \cap \boldsymbol{F}^j = \emptyset$.

The MP structure is best known for its use in the ECGA [1]. The method employed in ECGA to learn the structure is quite common in EDA literature. A greedy algorithm is used to optimize a statistical scoring metric. The scoring metric used in ECGA is the Minimum Description Length (MDL) metric and it should be minimized. The MDL met-

ric defines a trade-off between the quality of the fit of the estimated probability distribution and the number of parameters that is required for the fit. The quality of the fit is the negative log-likelihood, which, given a data set of size n, is identical to n times the entropy, i.e. $n\sum_{i=0}^{|\mathcal{F}|-1} H(X_{\boldsymbol{F}^i})$ with entropy being defined as $H(X_{\boldsymbol{F}^i}) = -\sum_{x\in\Omega(X_{\boldsymbol{F}^i})} P(X_{\boldsymbol{F}^i} = \boldsymbol{x})\log_2 P(X_{\boldsymbol{F}^i} = \boldsymbol{x})$ where $\Omega(X_{\boldsymbol{F}^i})$ denotes the sample space for random variables $X_{\boldsymbol{F}^i}$, i.e. all $2^{|\boldsymbol{F}^i|}$ binary strings of length $|\boldsymbol{F}^i|$. Typically, to estimate probabilities, maximum-likelihood (ML) estimates are used. For Cartesian sample spaces such as the binary one considered here, this amounts to counting frequencies and dividing by n. The number of parameters required using frequency tables is $\sum_{i=0}^{|\mathcal{F}|-1} 2^{|\boldsymbol{F}^i|} - 1$. Together, the MDL metric amounts to $(n\sum_{i=0}^{|\mathcal{F}|-1} H(X_{\boldsymbol{F}^i}))+(\sum_{i=0}^{|\mathcal{F}|-1} 2^{|\boldsymbol{F}^i|} - 1)$.

The greedy learning algorithm starts from the univariate structure and computes the decrease in MDL metric for all possible merges of two subsets $\boldsymbol{F}^i$ and $\boldsymbol{F}^j$. This decrease can be computed efficiently using only the variables that are involved and amounts to $n(H(X_{\boldsymbol{F}^i})+H(X_{\boldsymbol{F}^j})-H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}))+(2^{|\boldsymbol{F}^i|} - 1) + (2^{|\boldsymbol{F}^j|} - 1) - (2^{|\boldsymbol{F}^i\cup\boldsymbol{F}^j|} - 1)$. The merge that results in the biggest decrease in the MDL metric is chosen and the two subsets are replaced by their merged subset. If no merge operation exists that reduces the MDL metric further, the greedy search procedure stops.

4.4 Linkage Tree Structure

Whereas the MP structure expresses only a single level of dependence between problem variables, the linkage tree (LT) structure expresses a richer form of dependence [5]. In the MP model any two variables are either fully dependent or fully independent. In the LT model a variable can be part of multiple subsets. Therefore, any two variables may be dependent according to some subsets, but independent according to others. Specifically, the subsets in the LT model are ordered in a hierarchical fashion: for any subset $\boldsymbol{F}^i$ in the FOS that consists of more than one variable, there are two mutually exclusive subsets $\boldsymbol{F}^j$ and $\boldsymbol{F}^k$ in the FOS, both of which have less elements than $\boldsymbol{F}^i$ and their union is exactly $\boldsymbol{F}^i$, i.e. $\boldsymbol{F}^j\cap\boldsymbol{F}^k = \emptyset$, $|\boldsymbol{F}^j| < |\boldsymbol{F}^i|$, $|\boldsymbol{F}^k| < |\boldsymbol{F}^i|$ and $\boldsymbol{F}^j\cup\boldsymbol{F}^k = |\boldsymbol{F}^i|$. Furthermore, the hierarchical structure is complete in the sense that there exist two subsets in the FOS that together contain all problem variable indices.

The LT structure can be seen as the result of a hierarchical clustering procedure. Each problem variable is taken to be independent at first, i.e. clustering starts from the univariate structure. Then, similar to the greedy procedure for learning an MP, two subsets are combined. However, unlike for the MP model, both the combined subset as well as its constituent subsets are in the LT structure. Different from the learning procedure for the MP structure, the combining of subsets continues until only two subsets remain that together contain all problem variable indices.

The goal of the greedy algorithm here is slightly different from that in case of the MP structure. It is not the best trade-off between efficiency and quality of a single dependency structure but rather a complete hierarchical dependency clustering that is sought. For this reason, the complexity term in the MDL isn't needed, giving a metric change of $n(H(X_{\boldsymbol{F}^i}) + H(X_{\boldsymbol{F}^j}) - H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}))$ in which n can now be dropped because it is a constant (i.e. it is the population size), which gives the definition of mutual information: $I(X_{\boldsymbol{F}^i}, X_{\boldsymbol{F}^j}) = H(X_{\boldsymbol{F}^i}) + H(X_{\boldsymbol{F}^j}) - H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$. Following the greedy merge algorithm for the MP structure, this expression should still be maximized (maximum difference). However, because the notion behind building the LT structure is clustering, we want to look upon the difference as a distance measure and thus minimize it. Now, because $H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}) \geq \max\{H(X_{\boldsymbol{F}^i}), H(X_{\boldsymbol{F}^j})\}$ and thus $I(X_{\boldsymbol{F}^i}, X_{\boldsymbol{F}^j}) \leq H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$, one could use $H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}) - I(X_{\boldsymbol{F}^i}, X_{\boldsymbol{F}^j})$. However, this distance measure is biased by the number of variables in a subset. This can be overcome by dividing by $H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$, giving a distance measure of $1 - I(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})/H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$. A measure often associated with clustering in information theory is Variation of Information $VI(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}) = H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j}) - I(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$. Also VI is however biased toward subsets containing more variables. For this reason, often the scaled VI measure is used: $VI(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})/H(X_{\boldsymbol{F}^i\cup\boldsymbol{F}^j})$. This measure was also used in the first GA that used the linkage tree structure to perform variation, LTGA [5]. From the above however it can be seen that these concepts are *not* any different. In other words, the use of scaled VI to compute a linkage tree is no different than the use of the scaled mutual information and therefore very closely related to the partitioning that is computed in case of the MP structure.

In the original LTGA [5], the distance measure was computed using the definition of entropy for each subset. For large subsets of variables, i.e. high up in the linkage tree, this however requires counting the frequencies of instances for many variables. So, although use of the LT structure lifts the restriction of modelling only mutually exclusive subsets of variables as imposed by the MP structure, the computational complexity of configuring the LT structure is even larger. A faster clustering technique is the average linkage clustering or unweighted pair group method with arithmetic mean (UPGMA) and it involves only looking at distances between pairs of variables. In the case of the LT and the distance between two subsets of variables this comes down to $\frac{1}{|X_{\boldsymbol{F}^i}||X_{\boldsymbol{F}^j}|}\sum_{X\in X_{\boldsymbol{F}^i}}\sum_{Y\in X_{\boldsymbol{F}^j}} H(X,Y) - I(X,Y)$. Note that no normalization (dividing by $H(X,Y)$) is needed now because all atomic information-theoretic distance computations pertain to exactly two variables only. This distance measure can be computed very efficiently because all required information-theoretic measures between pairs of variables can be computed beforehand. Also, no more frequency counting for large sets of variables is required, making this model-building procedure even faster. Note that this remains true even in an EDA context under the assumption of ML estimates because sampling an instance for a set of variables is then identical to randomly picking a solution from the population and copying the values of variables that the instance needs to be sampled for. This measure has been studied before in the context of EDAs to speed up the configuration of the MP structure in ECGA [1]. However, in order to test whether the MDL score is still decreasing, a frequency count still needs to be performed for the variables contained in the merge that was deemed the most promising according to the UPGMA distance variant. All things combined, configuring the LT structure in this manner is even more efficient than configuring the MP structure in the typical ECGA manner.

To build the LT structure, the FOS is treated as a stack upon which first the complete univariate structure is pushed in a random order. Each time two sets are combined, the

```
EA
for i ∈ {0, 1, ..., n − 1} do
  P_i ← CreateRandomSolution()
  EvaluateFitness(P_i)
P ← TournamentSelection(P, n, 2)
while ¬ TerminationCriterionSatisfied() do
  LearnModelFromPopulation()
  VariationAndSelection()
```

Figure 3: Common EA outline.

joined set is also pushed on the stack. When traversing the LT structure, subsets are popped from the stack, meaning that the subsets are considered in reverse order of merging.

5. EVOLUTIONARY ALGORITHMS

5.1 Common EA Outline

In the remainder we denote the population by $\mathcal{P}$ and the set of offspring by $\mathcal{O}$. We further denote the population size by $n = |\mathcal{P}|$ and will assume $|\mathcal{O}| = |\mathcal{P}| = n$.

All algorithms in this paper belong to the class of evolutionary algorithms (EAs). Therefore, and to ensure a similar selection pressure in all algorithms, we first define a common core. The most important ingredients in any EA are selection and variation. Particular to the type of EAs that we are interested in, is the use of a configurable model on the basis of which the variation operators work. The model is learned from the population, after which variation is used to generate new solutions and finally survivor selection selects which solutions actually make it into the next generation. Although we acknowledge that the integration of local search techniques can be highly valuable, we are mostly concerned here with the workings of the variation operators in a black-box setting. As such, we initialize the population completely randomly. We define the method TournamentSelection$(\mathcal{A}, n, s)$ to select n solutions from set $\mathcal{A}$ through tournament selection with tournament size s. Using this definition, the common EA outline as used in this paper is presented in Figure 3.

5.2 Genetic Algorithm

Typical to the GA is the use of recombination on 2 parents to create 1 or 2 offspring. Once offspring solutions are fully constructed, their fitness values are evaluated. In the GA used in this paper, we generate n new solutions in this manner by randomly pairing 2 parents and generating 2 offspring solutions via recombination. In order not to lose track of good solutions created so far we combine the current population and the offspring and perform selection on these $2n$ solutions to select n survivors. To ensure convergence by logistic growth of the optimal solutions over multiple generations, we use tournament selection with a tournament size of 4. This way, each solution appears in a tournament twice and therefore the frequency of the best solution will always increase. Pseudo-code is given in Figure 4.

Recombination in GAs is classically of the crossover type where parts of the parent solutions are inherited by the offspring solutions. For the FOS structure this means that for each of the subsets, values for the variables in a subset are copied from the same, but randomly chosen parent. Note that when combined with the LT structure, this operation is *identical* to the use of the univariate structure. We

```
GA::VariationAndSelection()
π ← RandomPermutation({0, 1, ..., n − 1})
for i ∈ {0, 1, ..., n − 1} do
  (O_i, O_{i+1}) ← Recombine(P_{π_i}, P_{π_{i+1}})
  EvaluateFitness(O_i)
  EvaluateFitness(O_{i+1})
P ← TournamentSelection(P + O, n, 4)
```

Figure 4: GA variation and selection (n even).

```
GA::Recombine(p^0, p^1)
for i ∈ {0, 1, ..., |F| − 1} do
  if Random01() < 0.5 then
    o^0_{F^i} ← p^0_{F^i}; o^1_{F^i} ← p^1_{F^i}
  else
    o^0_{F^i} ← p^1_{F^i}; o^1_{F^i} ← p^0_{F^i}
Return(o^0, o^1)
```

Figure 5: GA variation for FOS model.

therefore only use the univariate and MP structures in combination with the GA. Furthermore, note that the use of the univariate structure gives the classically well-known uniform crossover operator. Pseudo-code is given in Figure 5.

5.3 Estimation-of-Distribution Algorithm

The EDA generates n new solutions by sampling a probability distribution. Similar to the GA, we then add these solutions to the population and perform tournament selection, see Figure 6. Sampling the probability distribution is fully dependent on the type of probability distribution. In this paper, we only consider distributions that directly follow from estimating joint distributions over the marginals imposed by the FOS. We perform estimations using the maximum-likelihood principle, i.e. marginal distributions are estimated by counting frequencies in the population for all possible instances of the variables. Probability distributions are automatically defined this way for the univariate structure and the MP structure, but not the LT structure. The univariate structure is found in various discrete EDAs such as UMDA, cGA and PBIL. The MP structure is typically found in the EDA known as ECGA [1]. In our experiments, the EDA with MP structure can therefore actually be considered to be exactly ECGA just as the EDA with univariate structure can be considered to be exactly UMDA or cGA. Pseudo-code for variation in EDAs is given in Figure 7.

5.4 Optimal Mixing Evolutionary Algorithm

Typical to the Optimal Mixing Evolutionary Algorithms (OMEAs) as introduced in this paper is the use of intermediate function evaluations to see if a certain operation during the construction of a solution was beneficial. This then can be seen as starting from an existing solution and trying iteratively to improve upon it. We therefore define the OMEA as trying to improve existing solutions. Here, be-

```
EDA::VariationAndSelection()
for i ∈ {0, 1, ..., n − 1} do
  O_i ← SampleDistribution()
  EvaluateFitness(O_i)
P ← TournamentSelection(P + O, n, 4)
```

Figure 6: EDA variation and selection.

EDA::SampleDistribution()

for $i \in \{0, 1, \ldots, |\mathcal{F}| - 1\}$ **do**
  $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow$ SAMPLESUBSETDISTRIBUTION$(P_{\boldsymbol{F}^i})$
  /* *Note: for ML estimates this is identical to* */
  /* $\boldsymbol{p} \leftarrow$ RANDOM$(\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n-1}\})$ */
  /* $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}_{\boldsymbol{F}^i}$ */
RETURN$(\boldsymbol{o})$

Figure 7: EDA variation for FOS model.

OMEA::VariationAndSelection()

for $i \in \{0, 1, \ldots, n-1\}$ **do**
  $\mathcal{O}_i \leftarrow \mathrm{OM}(\mathcal{P}_i)$
$\mathcal{P} \leftarrow$ TOURNAMENTSELECTION$(\mathcal{O}, n, 2)$

Figure 8: OMEA variation and selection.

cause the population size is equal to the number of offspring, this means that each solution undergoes OM (optimal mixing) variation. Because each offspring is at least as good as the parent that the OM variation started from, for survivor selection we can forget about the current population and focus only on the n offspring. To have the same convergence properties as the GA and the EDA, we lower the tournament selection size to 2. Note that the actual selection pressure is much higher as it is partly contained in the OM variation operators. Pseudo-code for OMEA is given in Figure 8.

Actual variation in OMEA follows the FOS structure and attempts to improve the current solution by changing a part of it as demarcated by the subsets in the FOS. We consider two types of OM variation: Recombinative Optimal Mixing (ROM) and Gene-pool Optimal Mixing (GOM). ROM and GOM can be seen as incremental variants of the variation operators in GA and EDA respectively. In ROM a single other parent is selected and copied to serve as a donor. Crossover is performed following the different subsets in the FOS. Although only one solution is checked for improvement (the one OM variation was started on), all information of the two parents is preserved as the donor copy undergoes crossover as well, receiving material from the solution undergoing OM variation. Care is taken not to re-evaluate a solution if nothing has changed. Because re-evaluations are potentially done frequently during OM variation, this check is vital compared to variation operators as in GA and EDA that only perform evaluation once a solution is fully constructed. Pseudo-code for ROM is given in Figure 9.

In the original LTGA [5] an operator similar to ROM was defined. However, there the donor was not a copy but truly

OMEA::ROM($\boldsymbol{x}$)

$\boldsymbol{o}^0 \leftarrow \boldsymbol{p}^0 \leftarrow \boldsymbol{x}$; $\mathit{fitness}[\boldsymbol{o}^0] \leftarrow \mathit{fitness}[\boldsymbol{p}^0] \leftarrow \mathit{fitness}[\boldsymbol{x}]$
$\boldsymbol{o}^1 \leftarrow \boldsymbol{p}^1 \leftarrow$ RANDOM$(\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n-1}\})$
for $i \in \{0, 1, \ldots, |\mathcal{F}| - 1\}$ **do**
  $\boldsymbol{o}^0_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}^1_{\boldsymbol{F}^i}$; $\boldsymbol{o}^1_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}^0_{\boldsymbol{F}^i}$
  if $\boldsymbol{o}^0_{\boldsymbol{F}^i} \neq \boldsymbol{o}^1_{\boldsymbol{F}^i}$ **then**
    EVALUATEFITNESS$(\boldsymbol{o}^0)$
    if $\mathit{fitness}[\boldsymbol{o}^0] > \mathit{fitness}[\boldsymbol{p}^0]$ **then**
      $\boldsymbol{p}^0_{\boldsymbol{F}^i} \leftarrow \boldsymbol{o}^0_{\boldsymbol{F}^i}$; $\mathit{fitness}[\boldsymbol{p}^0] \leftarrow \mathit{fitness}[\boldsymbol{o}^0]$; $\boldsymbol{p}^1_{\boldsymbol{F}^i} \leftarrow \boldsymbol{o}^1_{\boldsymbol{F}^i}$
    else
      $\boldsymbol{o}^0_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}^0_{\boldsymbol{F}^i}$; $\mathit{fitness}[\boldsymbol{o}^0] \leftarrow \mathit{fitness}[\boldsymbol{p}^0]$; $\boldsymbol{o}^1_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}^1_{\boldsymbol{F}^i}$
RETURN$(\boldsymbol{o}^0)$

Figure 9: ROM variation for FOS model.

OMEA::GOM($\boldsymbol{x}$)

$\boldsymbol{b} \leftarrow \boldsymbol{o} \leftarrow \boldsymbol{x}$; $\mathit{fitness}[\boldsymbol{b}] \leftarrow \mathit{fitness}[\boldsymbol{o}] \leftarrow \mathit{fitness}[\boldsymbol{x}]$
for $i \in \{0, 1, \ldots, |\mathcal{F}| - 1\}$ **do**
  $\boldsymbol{p} \leftarrow$ RANDOM$(\{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n-1}\})$
  $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{p}_{\boldsymbol{F}^i}$
  if $\boldsymbol{o}_{\boldsymbol{F}^i} \neq \boldsymbol{b}_{\boldsymbol{F}^i}$ **then**
    EVALUATEFITNESS$(\boldsymbol{o})$
    if $\mathit{fitness}[\boldsymbol{o}] > \mathit{fitness}[\boldsymbol{b}]$ **then**
      $\boldsymbol{b}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{o}_{\boldsymbol{F}^i}$; $\mathit{fitness}[\boldsymbol{b}] \leftarrow \mathit{fitness}[\boldsymbol{o}]$
    else
      $\boldsymbol{o}_{\boldsymbol{F}^i} \leftarrow \boldsymbol{b}_{\boldsymbol{F}^i}$; $\mathit{fitness}[\boldsymbol{o}] \leftarrow \mathit{fitness}[\boldsymbol{b}]$
RETURN$(\boldsymbol{o})$

Figure 10: GOM variation for FOS model.

another parent and both offspring were evaluated after each crossover operation. At the end, only the best offspring solution then is the result of the mixing operation. However, under the assumption that the learned structure is a good match with the problem structure this is only a source of inefficiency. One solution is always undesirable because it receives the material you are not interested in. In the original LTGA however, these "bad" solutions are evaluated anyway.

The EDA-equivalent OM variation operator is GOM. GOM is similar to ROM but instead of choosing a single donor parent for the entire variation procedure, it selects a new parent for each subset in the FOS. An alternative way of looking at GOM is to view it as the multi-parent recombination version of ROM. Pseudo-code is given in Figure 10. Note the similarity to the variation operation in EDA (Figure 7) in the case ML estimates are used (which is almost always the case in discrete EDAs). GOMEA can therefore also be seen as an Incremental-Subset-Sampling EDA (ISS-EDA).

6. EXPERIMENTS

6.1 Optimization problems

We consider three functions, all of which need to be maximized. The first is commonly known as onemax or bitcounting where fitness is simply the number of ones.

The second function is the mutually exclusive, additively decomposable composition of the well-known order-k deceptive trap functions. We consider subfunctions with $k = 5$:

$$f_{\text{Trap5}}(\boldsymbol{x}) = \sum_{i=0}^{(l/k)-1} f^{\text{sub}}_{\text{Trap-k}}\left(\sum_{j=ki}^{ki+k-1} x_j\right)$$

where

$$f^{\text{sub}}_{\text{Trap-k}}(u) = \begin{cases} 1 & \text{if } u = k \\ \frac{k-1-u}{k} & \textit{otherwise} \end{cases}$$

It is commonly known that the EA needs to detect and process the linkage groups pertaining to the deceptive subfunctions in order for optimization to proceed efficiently. Specifically, using univariate structures the minimally required population size and number of function evaluations scales up exponentially. It is clear that for this problem the MP structure is a perfect fit.

The third function is the nearest-neighbour overlapping, additively decomposable composition of predetermined, but completely random subfunctions of length k, which is essentially a NK-landscape [4]. This type of NK-landscape problem where the overlap between subfunctions is exactly

defined rather than randomly defined as in traditional NK-landscapes has the advantage that the optimum can be computed using dynamic programming [4], which is important when we want to use this type of function to determine the minimally required resources for an EA to find the optimum. We consider subfunctions of length 5 and the maximum overlap of 4 (corresponding to a shift of 1 where the next subfunction starts), but without wraparound:

$$f_{\text{NK-S1}}(\boldsymbol{x}) = \sum_{i=0}^{l-k} f_{\text{NK}}^{\text{sub}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$$

where $f_{\text{NK}}^{\text{sub}}\left(\boldsymbol{x}_{(i,i+1,\ldots,i+k-1)}\right)$ is a pre-determined, but randomly chosen value in $[0;1]$.

6.2 Setup

For each of the three optimization problems and problem lengths up to $l = 200$ we have determined the minimally required population size for GA, EDA, ROMEA and GOMEA in combination with each of the three FOS structures to solve the problem in 99 out of 100 independent runs. As indicated in Section 4 however, it doesn't make sense to combine GA and EDA with the LT structure as variation for these algorithms then becomes equivalent to using the univariate structure. We furthermore included the LTGA [5] in our experiments, combining it only with the LT structure, following its name and definition. However, we did change the learning procedure in LTGA to use the UPGMA mutual-information distance measure rather than the full-marginal scaled mutual-information distance measure used in the original LTGA. Note that this is also the LTGA version used in [2] where the UPGMA is called the pairwise distance measure. Furthermore, to ensure the same selection pressure as ROMEA and GOMEA, we added tournament selection with a tournament size of 2 to LTGA, similar as is done in ROMEA and GOMEA. In this way, the inefficiency in number of required function evaluations in the always-evaluate-both offspring in the LTGA variation procedure can be studied in isolation.

6.3 Results

The results for the minimally required population size, number of function evaluations and actual computing time in seconds are shown in Figures 11,12 and 13 respectively. Figure 11 nicely demonstrates existing knowledge about GAs and EDAs as well as the knowledge presented in this paper. As already known, the use of the univariate structure, like the uniform crossover operator in GAs, results in exponential scale-up behavior on problems with higher-order multivariate interactions such as the composition of deceptive trap functions and the overlapping NK landscapes. The results with the univariate structure also demonstrate how, due to faster mixing, EDAs are superior to GAs if the dependency structure used in the variation operator is configured to perfectly match the dependency structure in the problem. This phenomenon can be observed on the Onemax problem using the univariate structure. The reverse, hower, as a result of implicit multivariate dependency processing through 2-parent recombination, GAs are superior to EDAs if the dependency structure used in the variation operator is not correctly configured, can also be seen, for instance using the univariate structure on the trap functions.

Interestingly, and to the best of our knowledge, not demon-

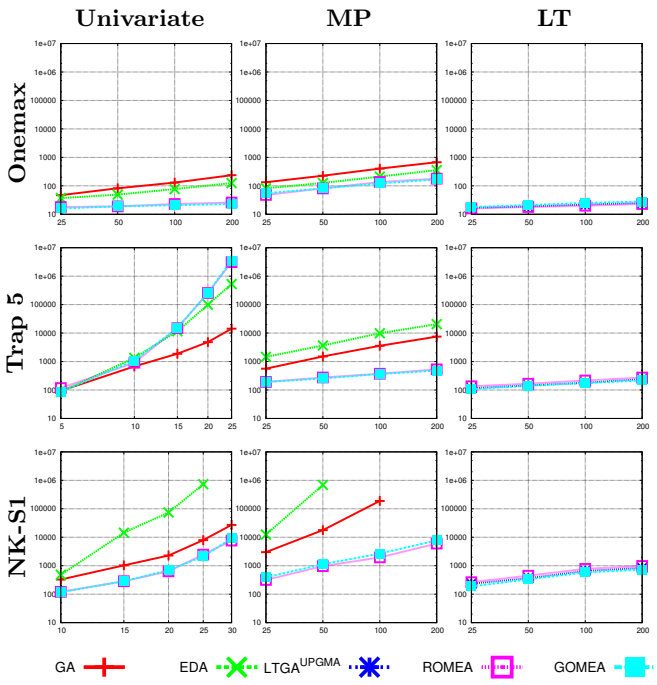

Figure 11: Minimally required population size ($n^{\min}$) to reach the optimum in at least 99 out of 100 runs.

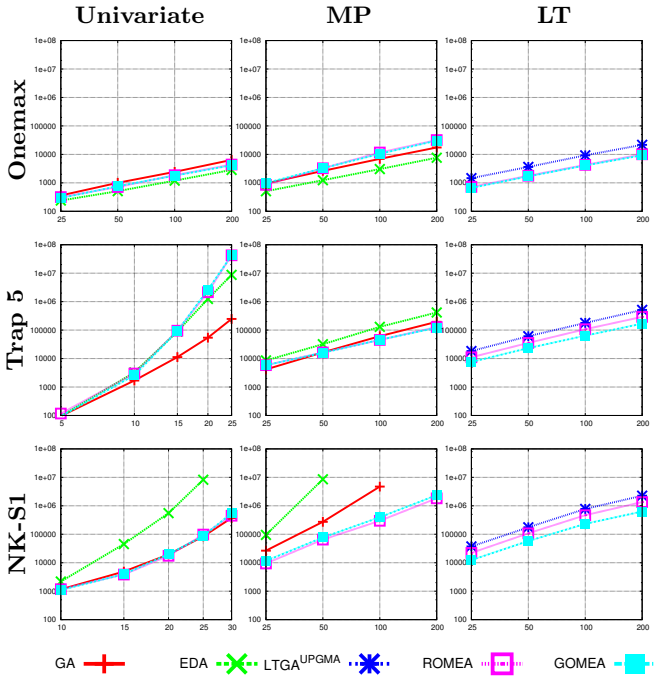

Figure 12: Required number of function evaluations averaged over 100 runs for population size $n^{\min}$.

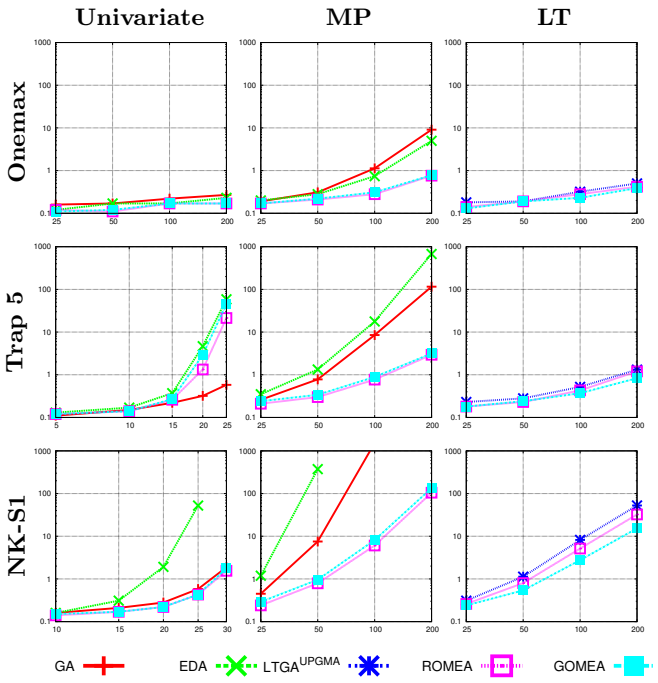

Figure 13: Required runtime in seconds averaged over 100 runs for population size $n^{\min}$.

strated before, the odds fall in favor of GAs rather than EDAs using the configurable MP structure that was originally introduced in an EDA (ECGA) rather than in a GA. This is because it takes time before the model is correctly configured. The more wrong the model still is, the more in favor the GA is. Once the model is configured correctly, in case of the MP structure and the current practice to learning it, the efficiency of the EDA cannot compensate anymore for the losses already sustained compared to the GA.

It is furthermore interesting to see that this phenomenon does not hold up under the incremental manner of variation as performed in the OMEAs. The reason for this is that the far more intense variation and re-evaluation results in an extreme type of selection pressure within a single generation and in an operator that acts close to something of a local search operator. As a result, very quickly the local structures such as the most promising instances for sets of dependent variables (i.e. the deceptive and global attractor in the deceptive trap functions) are found and identifying this structure requires only very few of such solutions. Therefore, not only is a far smaller population size required, regardless of the type of structure to be configured, also the model can be configured favorably much faster and the difference between recombinative and gene-pool variation becomes much smaller. Although with the MP structure the odds are still slightly in favor of recombinative mixing, using the LT structure, which can be configured favorably much faster and which is ultimately the structure we want to generally use, gene-pool variation is superior. Also, in terms of required actual running time the LT structure is vastly superior. Combined with the UPGMA mutual-information distance measure making the computing and storing of frequency counts for large marginals that involve many variables no longer needed, the LT structure can be configured very efficiently and effectively and is also exploited well by the OM manner of variation. For 200 variables, the use of an MP structure such as used in ECGA, even using OM variation ultimately results in an algorithm that requires a similar number of evaluations but approximately a factor 4 more in actual running time. Compared to the EAs that do not use the OM manner of variation, even with a similar number of evaluations the difference in required number of function evaluations becomes a factor of approximately 140 due to the far larger required population size.

Although subfunctions exist in the overlapping NK-landscape problem, clearly the MP structure is *not* a perfect fit. Although the LT structure doesn't appear to be a perfect fit either, having dependency sets of multiple sizes allows dependencies between blocks of variables to be expressed, which, when combined with incremental evaluation can allow for efficient construction of high-quality solutions. Indeed, all OMEAs with the LT structure can solve the overlapping NK-landscape problem. However, when the OMEAs are combined with the MP structure they are *also* capable of solving this problem. The incremental evaluation in OMEAs, or alternatively seen, the hillclimbing nature of OM variation on entire blocks in the MP structure rather than single variables in the univariate structure is clearly already highly beneficial. This supports the notion that it is not just finding a good configuration of a sufficiently complex structure, it is also the way in which this structural information is exploited upon creating new solutions that is of vital importance. OMEAs exploit this information more efficiently and effectively than GAs and EDAs.

Finally we note that the original LTGA, as a result of its always-evaluate-both offspring variation scheme, can indeed be seen to be less efficient than both ROMEA (by a factor $\geq$ 1.7) and GOMEA (by a factor $\geq$ 2.3) in terms of required number of function evaluations. This leads us to conclude that the most promising algorithm for future use, regarding all three criteria of required population size, number of required evaluations as well as computing time, is LT-GOMEA, from now on our new version of the linkage tree genetic algorithm (LTGA).

7. REFERENCES

[1] G. R. Harik, F. G. Lobo, and K. Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm. In M. Pelikan et al., *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, pages 39–61. Springer–Verlag, Berlin, 2006.

[2] M. Pelikan, M. Hauschild, and D. Thierens. Pairwise and problem-specific distance metrics in the linkage tree genetic algorithm. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO–2011)*. ACM Press.

[3] M. Pelikan, K. Sastry, and E. Cantú-Paz. *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*. Springer–Verlag, Berlin, 2006.

[4] M. Pelikan, K. Sastry, D. E. Goldberg, M. V. Butz, and M. Hauschild. Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO–2009)*, pages 851–858. ACM Press, 2009.

[5] D. Thierens. The linkage tree genetic algorithm. In *Proc. of Parallel Problem Solving from Nature (PPSN XI)*, pages 264–273, Springer, 2010.

[6] D. Thierens and D. E. Goldberg. Convergence models of genetic algorithm selection schemes. In *Proc. of Parallel Problem Solving from Nature (PPSN III)*, pages. 119–129. Springer, 1994.