Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithms

Ngoc Hoang Luong
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
Hoang.Luong@cwi.nl

Han La Poutré
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
Han.La.Poutre@cwi.nl

Peter A.N. Bosman
Centrum Wiskunde & Informatica (CWI)
Amsterdam, The Netherlands
Peter.Bosman@cwi.nl

ABSTRACT

The recently introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA), with a lean, but sufficient, linkage model and an efficient variation operator, has been shown to be a robust and efficient methodology for solving single objective (SO) optimization problems with superior performance compared to classic genetic algorithms (GAs) and estimation-of-distribution algorithms (EDAs). In this paper, we bring the strengths of GOMEAs to the multi-objective (MO) optimization realm. To this end, we modify the linkage learning procedure and the variation operator of GOMEAs to better suit the need of finding the whole Pareto-optimal front rather than a single best solution. Based on state-of-the-art studies on MOEAs, we further pinpoint and incorporate two other essential components for a scalable MO optimizer. First, the use of an elitist archive is beneficial for keeping track of non-dominated solutions when the main population size is limited. Second, clustering can be crucial if different parts of the Pareto-optimal front need to be handled differently. By combining these elements, we construct a multi-objective GOMEA (MO-GOMEA). Experimental results on various MO optimization problems confirm the capability and scalability of our MO-GOMEA that compare favorably with those of the well-known GA NSGA-II and the more recently introduced EDA mohBOA.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance, Experimentation

Keywords

Multi-objective optimization; Optimal Mixing; Linkage Tree Genetic Algorithm; Clustering

GECCO'14, July 12–16, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598261.

1. INTRODUCTION

A multi-objective (MO) optimization problem of m objectives $f_i(\boldsymbol{x}), i \in \{0, 1, \ldots, m-1\}$, is defined as, without loss of generality, to maximize all $f_i(\boldsymbol{x})$'s. The objective value vector of a solution $\boldsymbol{x}$ is $\boldsymbol{f}(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$. A solution $\boldsymbol{x}^0$ *dominates* a solution $\boldsymbol{x}^1$ (denoted $\boldsymbol{x}^0 \succ \boldsymbol{x}^1$) if and only if $f_i(\boldsymbol{x}^0) \geq f_i(\boldsymbol{x}^1), \forall i \in \{0, 1, \ldots, m-1\} \wedge \boldsymbol{f}(\boldsymbol{x}^0) \neq \boldsymbol{f}(\boldsymbol{x}^1)$. A solution $\boldsymbol{x}^0$ is Pareto optimal if and only if there does not exist a solution $\boldsymbol{x}^1$ such that $\boldsymbol{x}^1 \succ \boldsymbol{x}^0$. The Pareto-optimal set $\boldsymbol{\mathcal{P}_S}$ of the problem at hand is the set of all Pareto-optimal solutions. The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ is the set of the objective value vectors of all Pareto-optimal solutions. The goal of MO optimization is to find a set of non-dominated solutions whose objective value vectors constitute a good approximation of the Pareto-optimal front [4].

It is known that MO evolutionary algorithms (MOEAs) are an effective methodology for solving MO optimization problems [4]. Most MOEAs studies focus on the capability to find a good approximation of the Pareto-optimal front. The quality of approximations is often assessed based on both proximity to the optimal front (i.e. as close as possible) and diversity along the front (i.e. as well-spread as possible) [4]. Commonly studied MOEAs, such as the Nondominated Sorting Genetic Algorithm II (NSGA-II) [5] and the improved Strength Pareto Evolutionary Algorithm (SPEA2) [13], have been demonstrated to be effective in achieving this two-fold goal for a wide range of problems. However, the important issue of scalability is often overlooked [10], which is typically highly dependent on the algorithms' capability for efficient and effective linkage learning. It has been shown that without detecting and exploiting the dependencies between problem variables (e.g. using classic crossover and mutation operators), MOEAs cannot solve some decomposable problems efficiently [7, 10]. Addressing scalability, there exist MO estimation-of-distribution algorithms (MOEDAs), such as the Multi-objective Adapted Maximum-Likelihood Model (MAMaLGaM) [1] for continuous variables and the Multi-objective Hierarchical Bayesian Optimization Algorithm (mohBOA) [10] for discrete variables. Similar to the advantages of EDAs in single objective (SO) optimization, MOEDAs attempt to bring these advantages to MO optimization by replacing classic variation operators with model-based variation operators [7, 9, 10].

Although EDAs (and MOEDAs) are robust optimizers, probabilistic model building procedure typically comes at the cost of larger population sizes and more CPU time as a result of large computational complexity for model building [9]. Furthermore, the estimation of complete probability

distributions may be unnecessary if linkage information by itself suffices to perform variation effectively [12]. Such information is processed using an intensive, but effective, mixing procedure in the recently introduced class of Gene-pool Optimal Mixing Evolutionary Algorithms (GOMEAs) [12]. GOMEAs have been found to efficiently and reliably solve a variety of well-known SO benchmark problems, typically requiring far smaller population sizes and having better scalability in required function evaluations as compared to classic GAs and EDAs [12]. Moreover, certain classes of linkage models in GOMEA can be learned in $\mathcal{O}(nl^2)$ time [6] whereas learning comparable higher-order models in EDAs typically requires $\mathcal{O}(nl^3)$ time (where l is the problem size and n is the population size). It is these strengths of GOMEAs that we aim to transfer to MO optimization.

Studies on MOEAs have underlined important features for improving the capability of solving MO optimization problems. State-of-the-art MOEAs are characterized by implementations of the elitism concept in the context of MO optimization (i.e. when there exist multiple equally good trade-off solutions on the non-dominated Pareto front) [3]. A common elitism realization is the use of a secondary population, termed the elitist archive, for retaining non-dominated solutions found during the search. An archive can be beneficial because the sizes for the main population may be smaller than the number of solutions on the Pareto front and some non-dominated solutions can be lost due to selection [8].

The goal of MO optimization is two-fold: finding an approximation set of non-dominated solutions that is close to the Pareto-optimal front (i.e. proximity) and as diverse as possible (i.e. diversity, especially in the objective space) [4]. Standard MOEAs steer the population toward the optimal front while trying to preserve the diversity by different mechanisms, such as the selection based on crowding distance in NSGA-II [5] or the environmental selection in SPEA2 [13]. However, it has been showed that these mechanisms are insufficient for achieving a good scalability and that different parts of the optimal front should be processed separately [10]. State-of-the-art MOEDAs therefore often implement mixture probability distributions by clustering the selected solutions in the objective space and building a linkage model for each cluster separately (e.g. in mohBOA [10] and in MAMaLGaM [1]). Studies [10, 11] have noted the difficulty for finding the entire optimal front of some decomposable problems, especially the *extreme* regions of the optimal front, as for the studied problems the niches on the extremes become exponentially smaller than the niches in the *middle*. Furthermore, for MO optimization in general, selection tries to exploit all objectives simultaneously, thus reducing the pressure towards approaching the optimal front [1]. This problem was solved in MAMaLGaM-X$^+$ [1] by adding a separate SO optimizer for each objective and injecting the best solutions found by these SO optimizers into the elitist archive. In this paper, we adopt the idea of clustering from MAMaLGaM-X$^+$ but we adapt the process to fit well with the mechanism of GOMEA.

Having discussed the importance of elitist archiving, clustering, and linkage learning, in the remainder we will design and assemble these components into MO-GOMEA. Section 2 will describe these key components of MO-GOMEA. We presents benchmark problems and their characteristics in Section 3. Section 4 exhibits and discusses the experimental results. Finally, Section 5 concludes the paper.

```
MO-GOMEA  //population size n, k clusters
 1  t ← 0; t^NIS ← 0
 2  for i ∈ {0, 1, ..., n − 1} do
 3    P_i ← CreateRandomSolution()
 4    EvaluateFitness(P_i)
 5    A^t ← UpdateElitistArchive(P_i)
 6  while ¬TerminationCriteriaSatisfied do
 7    t ← t + 1
 8    {C_0, C_1, ..., C_{k−1}} ← ClusterPopulation(P)
 9    for j ∈ {0, 1, ..., k − 1} do
10      S_j ← TournamentSelection(C_j)
11      F_j ← LearnLinkageModel(S_j)
12    for i ∈ {0, 1, ..., n − 1} do
13      C_j ← DetermineCluster(P_i, {C_0, C_1, ..., C_{k−1}})
14      if ¬IsExtremeCluster(C_j) then
15        O_i ← MO-OptimalMixing(P_i, C_j, F_j, A^t)
16      else
17        O_i ← SO-OptimalMixing(P_i, C_j, F_j, A^t)
15    P ← O = {O_0, O_1, ..., O_{n−1}}
16    if f(A^t) ≠ f(A^{t−1}) then
17      t^NIS ← 0
18    else
19      t^NIS ← t^NIS + 1
```

Figure 1: Pseudo code for MO-GOMEA

2. MULTI-OBJECTIVE GOMEA

MO-GOMEA is started by randomly initializing a population $\mathcal{P}$ of n candidate solutions. All n solutions are evaluated to obtain their objective values. The population $\mathcal{P}$ is then clustered (in the objective space) into k clusters $\mathcal{C}_j$'s ($j \in \{0, 1, \ldots, k-1\}$) of equal sizes. For each cluster $\mathcal{C}_j$, selection is performed separately to obtain the corresponding selection set $\mathcal{S}_j$. From each $\mathcal{S}_j$, a separate linkage model $\mathcal{F}_j$ is learned. Finally, for each solution in the population $\mathcal{P}$, the cluster $\mathcal{C}_j$ that it belongs to is determined, and then variation is performed on the solution using the linkage relations captured by the corresponding linkage model $\mathcal{F}_j$ (see Section 2.4) where only the other solutions in the same cluster $\mathcal{C}_j$ are used as potential donors. This transforms each solution in the population into an offspring solution. These offspring complete replace the population. The pseudo-code for the outline of MO-GOMEA is given in Figure 1.

2.1 Elitist Archive

We use a basic, but effective, implementation of the elitist archive. We denote the elitist archive in generation t by $\mathcal{A}^t$ in decision-variable space and by $\boldsymbol{f}(\mathcal{A}^t)$ in objective space. Every newly generated solution is checked to see if it can be added into the archive. If the new solution is dominated by any archive member, it will be discarded. If it is a new non-dominated solution, it will be added into the archive, and archive members that are dominated by this new solution will be removed. In the case that there exists an archive member with the same objective values, the existing solution will be replaced by the new one if such replacement results in a diversity improvement for the archive in decision-variable space. We use a simple diversity metric for a solution: the Hamming distance to the nearest neighbor in the archive. Between the existing archive member and the new solution, the one having greater value for this metric will be chosen. How to ensure elitism for a bounded archive size is not our focus in this paper. Moreover, because of the nature of our benchmarks and experiment settings, we can assume that the elitist archive size is as large as the Pareto-optimal front.

2.2 Clustering

We use balanced k-leader-means clustering as previously used in MAMaLGaM [1] to cluster the population into k clusters of equal sizes. Note that clustering is performed in the objective space. First, a heuristic is used to select k leader solutions that are as well spread as possible. To do so, the first leader is the solution with maximum value in an arbitrary objective. The nearest-leader distances of all remaining solutions are computed as the distances to this first leader. The solution with the largest distance is chosen as the next leader. For every remaining solution, its nearest-leader distance is then updated if the distance to this new leader is smaller than the previous value. This process is repeated until k leaders are obtained. Second, k-means clustering is performed with the k leaders as the initial cluster means. Third, the distance from every solution to every final cluster mean is computed. Each cluster is then expanded to contain exactly c closest solutions, starting from each cluster mean. It was previously suggested to use $c = \frac{2}{k}|\boldsymbol{\mathcal{P}}|$, where $\boldsymbol{\mathcal{P}}$ are the solutions being clustered, resulting in overlap between neighboring clusters [1]. This reduces the probability that some solutions are not covered by any clusters. Moreover, it is beneficial to have equal-sized clusters so that a comparable amount of resources is used to handle each part of the Pareto-optimal front, to support obtaining an evenly-spread set of non-dominated solutions.

Instead of clustering the selection set as in mohBOA [10] and MAMaLGaM [1], in MO-GOMEA the whole population is clustered first, and selection is then performed for each cluster separately. This is done because MO-GOMEA does not generate offspring solutions by sampling new solutions from the learned models as is the case in MOEDAs. Instead, the variation operator in MO-GOMEA is used to improve each solution in the population in a more local-search like fashion (see Section 2.4). To do so, each solution in the population needs to be associated with a cluster.

Clustering helps to handle different parts of the Pareto-optimal front separately. This can be of major importance, especially for the extreme regions of the front where solutions maximize a single objective. This is because solutions from different extreme regions typically differ a lot. Moreover, in some problems the number of available solutions in the extreme regions can be much smaller than in the middle regions of the optimal front [11]. The additional use of separate SO optimizers to specifically obtain solutions in extreme regions of the front as used in MAMaLGaM-X$^+$ has been shown to be highly beneficial [1]. Although these separate optimizers can be tied in with the MOEA by running them generationally-synchronously parallel, putting the best solutions found by the separate optimizers into the elitist archive, and having the elitist archive participate in variation, such interaction between the populations of the SO and MO optimizers is still very limited. Therefore, instead of using external SO optimizers, in MO-GOMEA we use only a single population. In every generation, for each objective, in MO-GOMEA the cluster having the largest mean value in that objective is designated to perform only SO optimization in that objective during variation. If a single cluster happens to have the largest mean in more than one objective, we choose an objective arbitrarily. Moreover, while the selection procedure for learning linkage models in middle clusters is based on the Pareto-domination concept, the selection for extreme clusters is based on the single-objective only.

2.3 Linkage Learning

Linkage learning is performed for each cluster separately. Moreover, similar to the SO GOMEA, linkage learning is performed on a selection set that is obtained using tournament selection with tournament size 2 to have a beneficial bias in the model toward better fitness values.

Let $L = \{0, 1, \ldots, l-1\}$ be the set of indices of all l decision variables. To capture the interactions between these decision variables, GOMEAs use a general linkage model termed as the Family Of Subset (FOS). A FOS $\mathcal{F}$ consists of subsets of set L, i.e. $\mathcal{F} = \{\boldsymbol{F}^0, \boldsymbol{F}^1, \ldots, \boldsymbol{F}^{|\mathcal{F}|-1}\}$ where $\boldsymbol{F}^i \subseteq \{0, 1, \ldots, l-1\}$, $i \in \{0, 1, \ldots, |\mathcal{F}|-1\}$. A FOS $\mathcal{F}$ is thus a subset of the powerset of L, i.e. $\mathcal{F} \subseteq \mathcal{P}(L)$. Every subset $\boldsymbol{F}^i$ can be seen as a linkage group of problem variables that exhibit some degree of joint dependency and that should thus be copied jointly together when performing variation. Various GOMEA instances exist with different FOS structures [2, 12]. Here, we employ the Linkage Tree (LT) structure, which is the most commonly used in literature. In SO optimization, the GOMEA variant with LT is also known as the Linkage Tree Genetic Algorithm (LTGA) [12].

The LT contains all singleton subsets, i.e. $\boldsymbol{F}^i = \{i\}, i \in \{0, 1, \ldots, l-1\}$, capturing decision variables as being fully independent. The LT also contains combinations of variables, organized in a tree-like fashion. A branch node of the LT is a bivariate or multivariate subset $\boldsymbol{F}^i$, which is created by combining two subsets $\boldsymbol{F}^j$ and $\boldsymbol{F}^k$ such that $\boldsymbol{F}^j \cap \boldsymbol{F}^k = \emptyset, |\boldsymbol{F}^j| < |\boldsymbol{F}^i|, |\boldsymbol{F}^k| < |\boldsymbol{F}^i|$ and $\boldsymbol{F}^j \cup \boldsymbol{F}^k = \boldsymbol{F}^i$. The root node, which contains all the decision variable indices, is the set L itself and is discarded from the LT as it does not generate any new offspring solution when doing recombination. The LT has $2l-2$ linkage groups.

A full LT can be constructed by a procedure called Unweighted Pair Grouping Method with Arithmetic-mean (UPGMA [6]). Starting with all the singleton groups of univariate subsets (i.e. leaf nodes), multivariate groups (i.e. branch nodes) are created by consecutively combining two closest groups until the group containing all variable indices (i.e. root node) is created. Here, we use Mutual Information (MI) as the distance metric for UPGMA (i.e. a higher MI value indicates a closer distance). An optimal implementation of UPGMA exists that has $\mathcal{O}(nl^2)$ time complexity [6].

2.4 Optimal Mixing

Classic EAs use blind crossover and mutation to create offspring solutions. EDAs sample the learned probability distribution to generate new solutions. GOMEAs perform an intensive mixing procedure aimed at efficiently exploiting the FOS linkage model to improve all population members, one by one; the resulting solutions are offspring solutions.

Aimed at covering the whole Pareto-optimal front by discerning and exploiting different regions, the population is clustered in MO-GOMEA and for each cluster $\mathcal{C}_j$, a dedicated linkage model $\mathcal{F}_j$ is learned. Therefore, before improving a population member, we need to determine which cluster that solution belongs to. Although the clustering algorithm used here nicely constructs equally sized clusters, some solutions in $\mathcal{P}$ might actually not be covered by any cluster and some solutions may be covered by more than one cluster. Uncovered solutions are assigned to the cluster with the nearest mean. In case of multiple cluster assignments, ties are broken randomly. Every cluster thereby can be used to improve a more or less equal number of solutions.

Given cluster C_j that an existing solution $\boldsymbol{x}$ belongs to, and the corresponding FOS $\mathcal{F}_j$ of that cluster, MO-GOMEA changes $\boldsymbol{x}$, that we also refer to as the parent solution, incrementally by the Optimal Mixing (OM) procedure into an offspring solution as follows. First, the offspring solution $\boldsymbol{o}$ is created by cloning $\boldsymbol{x}$ and a backup $\boldsymbol{b}$ of $\boldsymbol{o}$ is created. We then traverse the linkage groups in FOS $\mathcal{F}_j$. For every $\boldsymbol{F}^i \in \mathcal{F}_j$, a donor solution $\boldsymbol{p}$ is randomly chosen from the same cluster C_j. The values of the problem variables whose indices are indicated by the linkage group $\mathcal{F}_i$ are copied from the donor $\boldsymbol{p}$ to the current solution $\boldsymbol{o}$. The objective values of this partially-altered solution $\boldsymbol{o}$ are evaluated and are compared with the backed-up state $\boldsymbol{b}$. If such mixing results in an improvement (i.e. the altered solution dominates the backed-up state $\boldsymbol{o} \succ \boldsymbol{b}$) or an equally good solution (i.e. the altered solution has the same objective values as the backed-up state $\boldsymbol{f}(\boldsymbol{o}) = \boldsymbol{f}(\boldsymbol{b})$) or a side step (i.e. the altered solution may not dominate the backed-up state but it is not dominated by any solutions in the elitist archive $\mathcal{A} \not\succ \boldsymbol{o}$), the changes are accepted and recorded as the new backup. Otherwise, the current solution is reverted to its backed-up state.

It can happen that all the mixing steps of OM do not improve nor at least change the parent solution or that there exist significant plateaus in the problem causing solutions to be changed back and forth. In SO GOMEA this problem was tackled by using a procedure termed Forced Improvement (FI) [2]. FI is essentially a second round of OM but the donor solution then is always the currently best found solution. In [2], FI is triggered when a parent solution is not changed after going through OM or when the number of subsequent generations that the best solution did not change, termed no-improvement stretch (NIS), exceeds the threshold $1 + \lfloor \log_{10}(n) \rfloor$. Here, we implement an MO version for FI through the following modifications. First, NIS is redefined as the number of consecutive generations that the Pareto front (in the objective space) formed by non-dominated solutions in the elitist archive stayed the same. Second, a new donor solution is now selected randomly from the elitist archive for each linkage group. Third, because FI aims to strictly improve the parent solution, a mixing step in the FI phase is only accepted if it results in a direct domination (i.e. $\boldsymbol{o} \succ \boldsymbol{b}$) or a Pareto-front improvement (i.e. a truly new non-dominated solution is found: $\mathcal{A}^t \not\succ \boldsymbol{o} \wedge \boldsymbol{f}(\boldsymbol{o}) \notin \boldsymbol{f}(\mathcal{A}^t)$). Similar to the SO version, rather than traversing the whole linkage tree, the FI procedure is stopped as soon as the first change is accepted. If the parent solution is still unchanged after FI, we simply replace it by a solution randomly chosen from the elitist archive. Pseudo-code is given in Figure 2.

OM in MO-GOMEA can be seen as a direct extension of OM in SO-GOMEA by replacing SO improvement checking with the Pareto-domination checking in every mixing step, and using multiple solutions from the elitist archive instead of a single best solution for the FI phase. MO-OM is employed for improving solutions that are determined as belonging to the middle region clusters. As discussed earlier however, having a mechanism that puts extra pressure on the individual objectives can be highly beneficial because the Pareto-domination improvement may not give enough pressure to find the extreme solutions (the solutions that maximize a single objective). Therefore, to improve solutions that belong to extreme region clusters, we employ SO-OM and SO-FI of the SO-GOMEA variant as reported in [2]. In other words, if a parent solution $\boldsymbol{x}$ belongs to the extreme

```
MO-OptimalMixing(x, C, F, A^t)
 1  b ← o ← x; f(b) ← f(o) ← f(x); changed ← false
 2  for i ∈ {0, 1, ..., |F| − 1} do
 3    p ← Random(C)
 4    o_{F^i} ← p_{F^i}
 5    if o_{F^i} ≠ b_{F^i} then
 6      f(o) ← EvaluateFitness(o)
 7      A^t ← UpdateElitistArchive(o)
 8      if (o ≻ b) or (f(o) = f(b)) or (A^t ⊁ o) then
 9        b_{F^i} ← o_{F^i}; f(b) ← f(o); changed ← true
10      else
11        o_{F^i} ← b_{F^i}; f(o) ← f(b)
12  if ¬changed or t^{NIS} > 1 + ⌊log10(n)⌋ then
13    changed ← false
14    for i ∈ {0, 1, ..., |F| − 1} do
15      p ← Random(A^t)
16      o_{F^i} ← p_{F^i}
17      if o_{F^i} ≠ b_{F^i} then
18        f(o) ← EvaluateFitness(o)
19        if (o ≻ b) or (A^t ⊁ o and f(o) ∉ f(A^t)) then
20          b_{F^i} ← o_{F^i}; f(b) ← f(o); changed ← true
21        else
22          o_{F^i} ← b_{F^i}; f(o) ← f(b)
23        A^t ← UpdateElitistArchive(o)
24      if changed then breakfor
25  if ¬changed then
26    p ← Random(A^t); o ← p; f(o) ← f(p)
```

Figure 2: Pseudo code for multi-objective Optimal Mixing

cluster that is associated with optimizing single objective f_i, $\boldsymbol{x}$ will be improved by OM in the SO manner, and the solution $\boldsymbol{x}_{f_i}^{best}$ that stores the current maximum observed value for f_i will be considered as the single donor solution in FI. Note that, however, the NIS concept is still defined in the overall MO optimization context with regard to the Pareto front formed by the elitist archive members.

3. EXPERIMENTAL SETTINGS

3.1 Benchmark Problems

3.1.1 Zeromax - Onemax

Zeromax - Onemax [10] has two objectives that are defined over a binary string $\boldsymbol{x}$ of l bits as follows:

$$f_{\text{Onemax}}(\boldsymbol{x}) = \sum_{i=0}^{l-1} x_i; \quad f_{\text{Zeromax}}(\boldsymbol{x}) = l - f_{\text{Onemax}}(\boldsymbol{x}) \quad (1)$$

Objective f_{Onemax} counts the number of bits set to 1 while f_{Zeromax} counts the number of bits set to 0, making the objectives conflicting. Every candidate solution is a Pareto-optimal solution; any increase in the number of bits set to 1 will be a decrease in the number of bits set to 0, and vice versa. The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ of Zeromax - Onemax consists of $l+1$ points $\boldsymbol{\mathcal{P}}_F = \{(i, l-i) \mid i \in \{0, 1, \ldots l\}\}$, on a straight line. A point on $\boldsymbol{\mathcal{P}_F}$ can correspond to many different solutions, with the exception of the two extreme points that correspond to exactly a string of all 1s (maximizing Onemax) and a string of all 0s (maximizing Zeromax). It can be seen that the niches of the solutions having objective values in the middle regions of the front $\boldsymbol{\mathcal{P}_F}$ are exponentially larger than the niches in the extreme regions of $\boldsymbol{\mathcal{P}_F}$.

3.1.2 Trap-5 - Inverse Trap-5

Trap-5 - Inverse Trap-5 also has two objectives. Trap-5 is the well-known mutually exclusive, additively decomposable composition of the order-5 deceptive subfunctions:

$$f_{\text{Trap-5}}(\boldsymbol{x}) = \sum_{i=0}^{(l/5)-1} f^{\text{sub}}_{\text{Trap-5}}\left(\sum_{j=5i}^{5i+4} x_j\right) \quad (2)$$

where

$$f^{\text{sub}}_{\text{Trap-5}}(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4-u & \text{if } u < 5 \end{cases} \quad (3)$$

Inverse Trap-5 comprises the same partitions as Trap-5 but each partition is evaluated "inversely":

$$f^{\text{sub}}_{\text{Inverse-Trap-5}}(u) = \begin{cases} 5 & \text{if } u = 0 \\ u-1 & \text{if } u > 0 \end{cases} \quad (4)$$

A candidate solution for which each partition has one objective subfunction that evaluates to 5 is Pareto-optimal. The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ consists of $l/5+1$ points $\boldsymbol{\mathcal{P}}_F = \{(5i + 4(l/5 - i), 5(l/5 - i) + 4i) \mid i \in \{0, 1, \dots l/5\}\}$, on a straight line. It is well-known known that in order to solve additively decomposable trap functions, it is crucial to preserve the linkage relations between problem variables during variation. Therefore, this benchmark is often used as a demonstration for the importance of linkage learning.

3.1.3 Leading Ones Trailing Zeros (LOTZ)

The Leading Ones Trailing Zeros (LOTZ) problem consists of two objectives: maximizing the number of consecutive bits set to 1 at the beginning and maximizing the number of consecutive bits set to zero at the end.

$$f_{\text{LO}}(\boldsymbol{x}) = \sum_{i=0}^{l-1}\prod_{j=0}^{i} x_j; \quad f_{\text{TZ}}(\boldsymbol{x}) = \sum_{i=0}^{l-1}\prod_{j=i}^{l-1}(1-x_j) \quad (5)$$

A candidate solution that consists of a substring of all 1s followed by a substring of all 0s, i.e. having the form of $11\dots100\dots0$, is Pareto-optimal. The two extreme solutions are all 1s (maximizing Leading Ones) and all 0s (maximizing Trailing Zeros). The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ of LOTZ consists of $l+1$ points $\boldsymbol{\mathcal{P}}^i_F = \{(i, l-i) \mid i \in \{0, 1, \dots l\}\}$, on a straight line. However, different from the Zeromax - Onemax problem, any point on Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ of LOTZ corresponds to exactly one Pareto-optimal solution.

3.1.4 Multi-objective MAXCUT

Weighted MAXCUT is defined over a weighted undirected graph $G = (V, E)$, where $V = (v_0, v_1, \dots, v_{l-1})$ is the set of l vertices, and E is the set of edges (v_i, v_j) with corresponding weights w_{ij}'s. A maximum cut is a partition of l vertices into two disjoint subsets A and $B = V \setminus A$ such that the combined weight of all edges (v_i, v_j) having $v_i \in A$ and $v_j \in B$ is maximized. A cut can be encoded as a binary string $\boldsymbol{x}$ of l bits, in which each bit x_i corresponds to a vertex, and all 0-valued bits indicate vertices of set A while all 1-valued bits indicate vertices of set B. The objective function of the weighted MAXCUT problem is defined as follows

$$f_{\text{weighted MAXCUT}}(\boldsymbol{x}) = \sum_{(v_i,v_j)\in E} \begin{cases} w_{ij} & \text{if } x_i \neq x_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

We construct an MO version of the weighted MAXCUT problem by optimizing a different MAXCUT instance in each objective. The instances have identical vertices but different edge weights. In our setup, each MAXCUT instance is a fully connected graph having $\frac{1}{2}l(l-1)$ edges.

3.2 Performance Indicator

To compare the performance of different MO optimizers, we employ the $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}$ performance indicator, also known as inverse generational distance [1]:

$$\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}(\mathcal{S}) = \frac{1}{|\boldsymbol{\mathcal{P}_F}|}\sum_{\boldsymbol{f}^0 \in \boldsymbol{\mathcal{P}_F}} \min_{\boldsymbol{x}\in\mathcal{S}}\{d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}^0)\} \quad (7)$$

where $\boldsymbol{\mathcal{P}_F}$ is the Pareto-optimal front, $\mathcal{S}$ is the final approximation front (i.e. the outcome of that optimizer), and $d(\cdot,\cdot)$ computes the Euclidean distance. The $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}$ performance indicator is the average distance from a point in $\boldsymbol{\mathcal{P}_F}$ to its nearest point in $\mathcal{S}$. A smaller value of $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}$ indicates a better performance result, and the value of 0 is achieved if and only if the approximation set equals the Pareto-optimal front. Note that calculating the average distance in "reversed order" yields a different indicator $\boldsymbol{D_{\mathcal{S} \to \mathcal{P}_F}}$, also known as generational distance. However, obtaining a low $\boldsymbol{D_{\mathcal{S} \to \mathcal{P}_F}}$ score does not mean that a good approximation front has been found since having a single Pareto-optimal point already gives a value of 0. The $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}}$ indicator is more useful because it balances both proximity (i.e. how close $\mathcal{S}$ is to the front $\boldsymbol{\mathcal{P}_F}$) and diversity (i.e. how well-spread $\mathcal{S}$ is along the front $\boldsymbol{\mathcal{P}_F}$) of the approximation set $\mathcal{S}$.

For the problems Zeromax - Onemax, Trap-5 - Inverse Trap-5, and LOTZ, $\boldsymbol{\mathcal{P}_F}$ is known, as described in the previous section. For MO MAXCUT problem however, we can only construct $\boldsymbol{\mathcal{P}_F}$ for small problem sizes $l \in \{6, 12, 25\}$ using enumerative methods. For $l \in \{50, 100\}$, we can only obtain reference fronts by consulting the results of [2] as follows. We use the maximum population sizes of LTGA, that reliably solve the single-objective MAXCUT, reported in [2] to set as the cluster size for MO-GOMEA. We then run 5 different instances of MO-GOMEA with different number of clusters $k \in \{1, 3, 5, 7, 10\}$. Each MO-GOMEA instance is run 100 times, each time with a budget of 20 million function evaluations. We take the Pareto front of all results in all these runs over all 5 instances of MO-GOMEA to be $\boldsymbol{\mathcal{P}_F}$ in this case. A degree of reliability can be taken from the fact that the optimal extreme points were always found to be in the final reference fronts $\boldsymbol{\mathcal{P}_F}$ so constructed by using a dedicated SO MAXCUT optimal solver from literature.

3.3 Experimental Setup

We compare the performance of MO-GOMEA with an commonly used instance of NSGA-II (2-point crossover with probability 0.9 and bit-flipping mutation with probability $1/l$). We also consider the best results obtained by the EDAs (MO-)UMDA and mohBOA as reported in [10]. We want to assess scalability, i.e. how the population size requirements and the required number of evaluations grow as the problem size increases. For the problems Zeromax - Onemax, Trap-5 - Inverse Trap-5, and LOTZ, the Pareto-optimal fronts are known, so to this end, we perform bisection. Bisection is a binary-search inspired procedure that aims to find the minimally required population size to solve a problem instance reliably. Here, we define reliable as achieving $\boldsymbol{D_{\mathcal{P}_F \to \mathcal{S}}} = 0$ (i.e. the entire Pareto-optimal front is found) in all 100 out of 100 independent runs. We perform 10 independent bisection searches for every problem size $l \in \{25, 50, 100, 200, 400\}$.

For MO-MAXCUT, because we do not know the Pareto-optimal front for certain, it is more difficult to perform bisection. Moreover, MO-MAXCUT is a (NP)-hard problem, for which polynomial scalability may not be expected, so

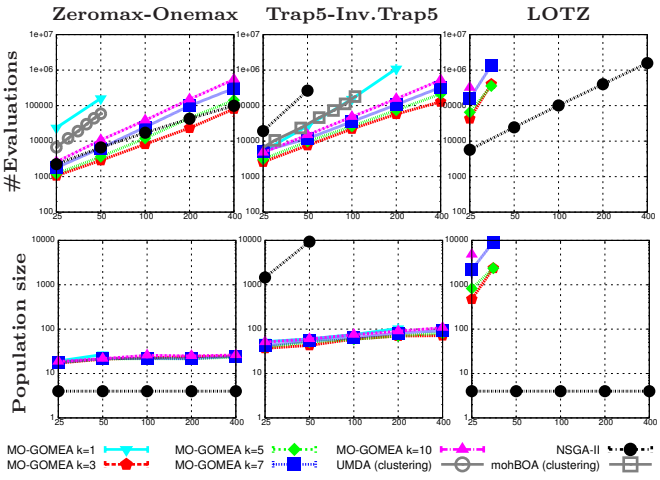

Figure 3: Scalability of MO-GOMEAs and competing optimizers. Horizontal axis: problem size.

we wish to observe how good of an approximation can be achieved. Therefore, we set the population sizes for MO-GOMEA and NSGA-II by consulting the results in [2], which solved SO-MAXCUT reliably. For each problem size, we use the average required population size of LTGA as the cluster size in MO-GOMEA and as the base population size in NSGA-II. We run MO-GOMEA with the number of clusters $k \in \{1, 3, 5, 7, 10\}$. Similarly, we run NSGA-II with the base population size scaled by $1, 2, 4, 8,$ and 16 times. For every MAXCUT problem instance $l \in \{6, 12, 25, 50, 100\}$, we perform 100 independent runs. We determine performance on the basis of the average convergence graph of the $\boldsymbol{D}_{\mathcal{P}_F \to \mathcal{S}}$ indicator. The budgets of function evaluations are also set by consulting [2]. For every problem size, we set the maximum number of evaluations to be 10 times the average number of evaluations that LTGA required to solve SO-MAXCUT (we multiply by 10 because we have a MO-GOMEA instance with at most $k = 10$ clusters). Note that we always consider 1 function evaluation to include evaluating both objectives.

4. RESULTS & DISCUSSIONS

4.1 Zeromax - Onemax

Figure 3 shows that MO-GOMEA with clustering (i.e. $k > 1$) outperforms UMDA with clustering in terms of number of evaluations as well as scalability. Note that, in [10], UMDA with clustering did not have an elitist archive and the required population size was not reported. It can be inferred however, that in order to cover all points of the Pareto-optimal front, the population size of UMDA without elitist archive must be at least as large as the Pareto-optimal front. Our MO-GOMEA instances require a smaller population size that grows quite slowly as the problem size increases. It can furthermore be observed that for solving this problem we do not really need many clusters in MO-GOMEA because all problem variables are independent from each other in all regions of the search space. That explains why MO-GOMEA with a number of clusters $k = 3$ has the best performance, and as we increase the number of clusters ($k = 5, 7, 10$), the number of evaluation also increases accordingly, but the scalability is relatively the same. However, MO-GOMEA without clustering ($k = 1$), and thus without internal SO optimizers, solves the problem much

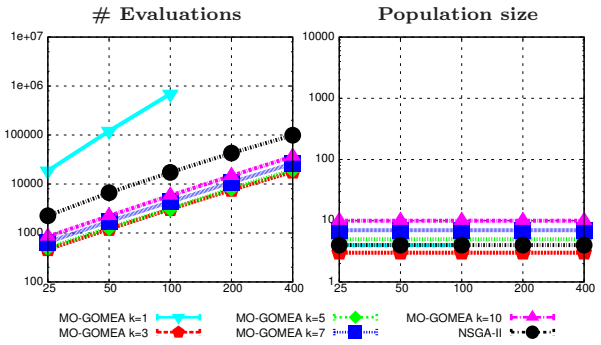

Figure 4: Scalability of MO-GOMEAs with mutation on Zeromax-Onemax. Horizontal axis: problem size.

less efficiently. This is because the 2 extreme solutions (maximizing Zeromax vs. maximizing Onemax) are in very small niches that contain only 1 solution. Without clustering to distribute enough resources and without SO optimizers to approach these extreme regions efficiently, it is much harder to cover the entire Pareto-optimal front. In [10], an NSGA-II version without an elitist archive was found to hardly be able to solve the problem up to 50 problem variables. Here, being equipped with an elitist archive, NSGA-II can solve all Zeromax - Onemax instances with the smallest possible population size for NSGA-II of 4 individuals and with a performance much better than UMDA. This confirms the necessity of elitist archiving for scalable MO optimization.

NSGA-II does not have clustering nor SO optimizers. The fact that it can solve all Zeromax - Onemax of various problem sizes with such a small population size (and with slightly better scalability than MO-GOMEA) can be explained by its use of bit-wise mutation, which is not by default employed in MO-GOMEA. To validate this, we equip MO-GOMEA with a simple mutation operator: the mixing step at singleton linkage groups is replaced by randomly assigning a 0 or a 1 value (i.e. not genepool-based). The performance of MO-GOMEAs with this mutation operator is shown in Figure 4. With mutation, indeed the MO-GOMEAs now have a scalability relatively similar to NSGA-II. Moreover, the MO-GOMEAs need a fewer number of evaluations. The fact that MO-GOMEA with mutation but without clustering ($k = 1$, and thus without SO optimizers) still has worse performance indicates that clustering and internal SO optimizers are important features of MO-GOMEA. It is however also a result of the fact that mutation here is key to finding the extreme points. This however requires time in terms of generations and all higher-order mixing operations attempted by MO-GOMEA that follow from use of the full LT in the mean time are not useful. It would therefore be interesting in future work to look at techniques to filter out superfluous FOS sets as was recently done for the SO case [2].

4.2 Trap-5 - Inverse Trap-5

Figure 3 clearly demonstrates that MO-GOMEA outperforms both mohBOA and NSGA-II. Different from the Zeromax - Onemax benchmark of all-independent problem variables, Trap-5 - Inverse Trap-5 requires linkage learning to detect and preserve linkage relations between problem variables during recombination processes. Linkage learning is not employed in NSGA-II and even though it uses a recombination operator (two-point crossover) that is linkage friendly because we have encoded the trap functions tightly,

it is still highly inefficient in finding the optimum. The strengths of the linkage model and optimal mixing procedure of GOMEA, which are known to contribute to its superior performance in solving the SO version of Trap-5 [12], is successfully extended to the MO realm. Given a properly learned linkage model, the OM procedure helps MO-GOMEA achieve successful mixing events which respect the dependencies between problem variables. The intensive local-search-inspired characteristic of the OM procedure moreover results in MO-GOMEA requiring only a small and slow-growth population size, similarly to the SO case [12].

The best results are again obtained with MO-GOMEA with 3 clusters. Using more clusters again leads to similar scalability but requires larger numbers of evaluations. This is again because the Pareto-optimal front of Trap-5 - Inverse Trap-5 has the same linkage structure (i.e. concatenated groups of 5 inter-dependent problem variables) in both objectives, resulting in redundant behavior when adding more clusters. However, MO-GOMEA with $k = 1$ clearly performs worse (but still better than NSGA-II) because the extreme regions of the front have again very small niches (similar to Zeromax - Onemax). Figure 3 also shows the performance of the MOEDA mohBOA with restricted tournament replacement and clustering, which had the best performance reported in [10]. Here, we obtain similar behavior for $k = 1$ but we obtain clearly better results for $k > 1$. Because in [10] mohBOA did not have an elitist archive, it is difficult to directly compare MO-GOMEA and mohBOA, which we did not re-implement. However, it has been reported in [12] that, in SO optimization for Trap-5, GOMEA had better performance than a typical EDA. Even if we assume that the probabilistic model building and sampling in mohBOA are as effective as the LT building and the OM procedure in MO-GOMEA, the elitist archive and internal SO optimizers cause MO-GOMEA to have better scalability.

4.3 LOTZ

Figure 3 shows that LOTZ is a challenge for MO-GOMEA. The reason is that we aim to cover the entire Pareto-optimal front reliably while LOTZ has a peculiar linkage structure. Only substrings of consecutive 1 bits at the beginning (Leading Ones) and the substring of consecutive 0 bits at the end (Trailing Zeros) contribute to the objective values. This means that all the mixing events of 0s and 1s in the middle are useless and can be considered as *noise*, affecting the effectiveness of linkage learning in MO-GOMEA. Moreover, the LT model fails to capture efficiently the type of operations needed to solve LOTZ. What further makes LOTZ a complicated problem is that selection pressure based on Pareto dominance makes it very difficult to obtain the extreme solutions of LOTZ (i.e. a string of all 1s and a string of all 0s). For example, a string $\boldsymbol{x}$ beginning with a 0 will be dominated by any strings beginning with a 1 and ending with a 0. This string $\boldsymbol{x}$ will therefore soon be deleted from of the population. The OM of GOMEA cannot improve $\boldsymbol{x}$ while still keeping the leading 0 bit because OM will prefer any dominating solutions $\boldsymbol{x}' \succ \boldsymbol{x}$ with a leading 1 bit. Even the SO version of OM in the extreme clusters cannot efficiently preserve this leading 0 bit because the leading 0 bit is only useful when it is combined with the substring of trailing 0s to create the extreme solution of all 0s. Figure 3 shows however that NSGA-II has little difficulty in solving LOTZ, requiring a population size of only 4. This is due to the

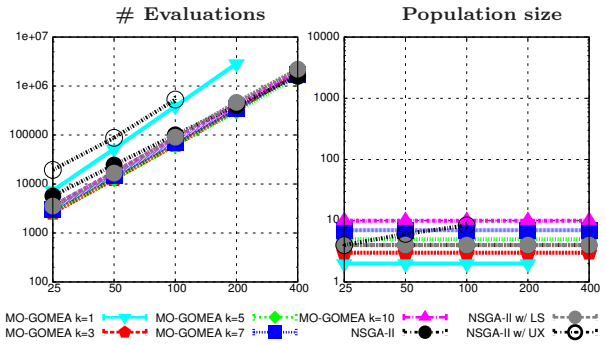

Figure 5: Scalability of MO-GOMEAs with local search and NSGA-IIs on LOTZ. Horizontal axis: problem size.

mutation operator of NSGA-II. It can be inferred from Figure 3 that, e.g. in order to solve LOTZ of 400 bits, NSGA-II needs to be run over than 300,000 generations, waiting for mutation to flip the right bit at the right time to obtain a Pareto-optimal solution. It should further be noted that the two-point crossover operator of NSGA-II has a favorable search bias here because it can preserve and recombine very large substrings of all 1s or all 0s at the beginning and end of a solution. Figure 5 validates this by showing that NSGA-II with uniform crossover performs worse.

LOTZ clearly poses additional challenges, which for MO-GOMEA include finding more appropriate linkage models and filtering useless subsets. However, LOTZ can still be solved fairly efficiently using a simple local search (LS) operator. Following the typical design of genetic local search, we apply LS at the end of every generation on each offspring solution by traversing the variables of a solution in a random order and flipping every variable, followed by a Pareto-improvement check. If a flip does not result in a solution dominated by the previous state nor dominated by the beginning state, it is accepted. Otherwise, it is reverted to the previous state. Figure 5 shows the performance of MO-GOMEA and NSGA-II with this LS. LS greatly improves the performance of MO-GOMEA, but not that of NSGA-II in terms of number of evaluations. These results underline just how beneficial it can be to use a proper LS operator to bring back diversity that might have gotten lost.

4.4 Multi-objective MAXCUT

Although Zeromax - Onemax, Trap-5 - Inverse Trap-5 and LOTZ are interesting benchmark problems with known Pareto-optimal fronts, they are fairly artificial in the sense that the Pareto-optimal fronts always have the shape of a straight line. We therefore also perform experiments the MO-MAXCUT problems where the Pareto-optimal front can be shaped very differently. The convergence graphs in Figure 6 show that for $l = 6$, NSGA-II performs slightly better than MO-GOMEA. For $l = 12$, only NSGA-II 16×, the one with the largest population size, outperforms MO-GOMEA. As the problem size increases however, the picture starts to change. For $l = 25$, only NSGA-II 16× has a convergence result as good as MO-GOMEA with $k = 5, 7$, but it is still outperformed by MO-GOMEA with $k = 10$. For larger problem sizes, $l = 50, 100$, MO-GOMEAs with clustering ($k > 1$) distinctly outperform all NSGA-II instances. The facts that MO-GOMEA with clustering exhibits better convergence for problem sizes $l > 25$ and that the bigger the problem, the wider the performance gap between the various

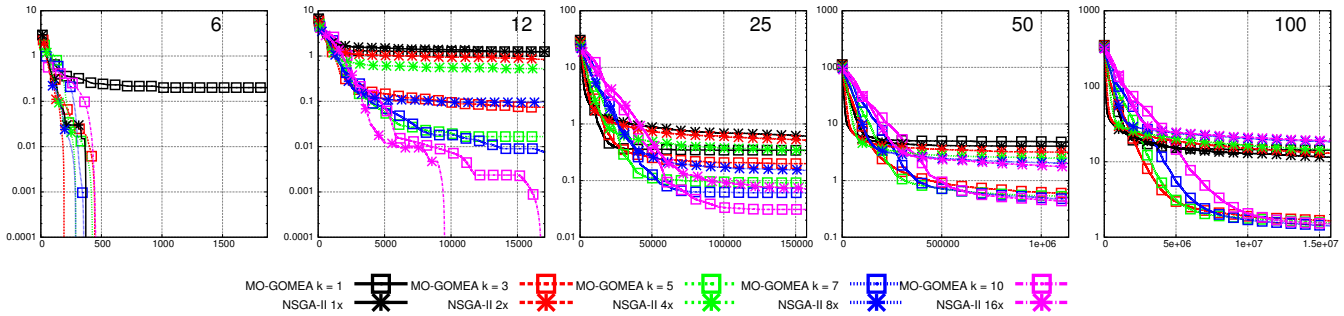

Figure 6: Average performance of instances of MO-GOMEA and instances of NSGA-II on multi-objective MAXCUT problems of different problem sizes. Horizontal axis: number of evaluations (both objectives per evaluation). Vertical axis: $\boldsymbol{D}_{\boldsymbol{\mathcal{P}}_{\boldsymbol{F}} \to \mathcal{S}}$

MO-GOMEAs $k > 1$ and NSGA-IIs becomes, indicate the intrinsic superior scalability of MO-GOMEA over NSGA-II. MO-GOMEA without clustering ($k = 1$) has performance results relatively the same as those of NSGA-IIs, which do not have clustering either, on most problem sizes ($l > 6$). This emphasizes again the importance of clustering to handle different parts of the Pareto-optimal front separately. Also, although we did not perform bisection to find the minimally required population size for each instance (because the entire true Pareto-optimal front is unknown), it can be inferred, to a certain degree, from MAXCUT $l = 12, 25, 50$ that the more clusters MO-GOMEA have the better the convergence result is. This is because, different from the other benchmark problems, the structures of the objectives may now differ, causing different structures to require to be exploited along the front, which is supported by clustering.

5. CONCLUSIONS

In this paper, we have presented the MO-GOMEA framework. We have shown that for the combination with the LT FOS model, superior scalability for solving different classes of MO optimization problems can be achieved as compared to classic GAs (i.e. NSGA-II) and even state-of-the-art EDAs (i.e. mohBOA). Our experimental results further support that the key features of scalable MO optimizers that we identified and incorporated into MO-GOMEA are indeed responsible for the observed performance. These features are: an elitist archive to keep track of the Pareto front, clustering to process different regions of the Pareto front differently, linkage learning and an efficient mechanism for exploiting the learned linkage relations to generate offspring solutions. An elitist archive, even in a basic implementation, is beneficial for solving MO optimization problems. Clustering and the incorporation of an explicit, dedicated bias for SO optimization embedded in extreme clusters help MO-GOMEA to have better coverage of the Pareto-optimal front and to find extreme solutions more efficiently. Finally, the efficient and effective linkage learning and exploitation contribute to the scalability of MO-GOMEA in solving MO optimization problems. The experimental results have further indicated that diversity loss can easily play an important role in MO optimization and particularly for MO-GOMEA, for which reason diversity improving mechanisms such as mutation and local search should be studied in more detail in the future. Moreover, especially when multiple clusters are used, there is a clear need for filtering superfluous subsets in the FOS models and further improve scalability.

6. REFERENCES

[1] P. A. N. Bosman. The anticipated mean shift and cluster registration in mixture-based edas for multi-objective optimization. In *Proc. of the Genetic and Evolutionary Comp. Conf. - GECCO-2010*, pages 351–358. ACM, 2010.

[2] P. A. N. Bosman and D. Thierens. More concise and robust linkage learning by filtering and combining linkage hierarchies. In *Proc. of the Genetic and Evolutionary Comp. Conf. - GECCO-2013*, pages 359–366. ACM, 2013.

[3] C. Coello Coello, G. Pulido, and E. Montes. Current and future research trends in evolutionary multiobjective optimization. In *Information Processing with Evolutionary Algorithms*, Advanced Information and Knowledge Processing, pages 213–231. Springer London, 2005.

[4] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., NY, USA, 2001.

[5] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evol. Computation*, 6(2):182–197, 2002.

[6] I. Gronau and S. Moran. Optimal implementations of UPGMA and other common clustering algorithms. *Information Processing Letters*, 104(6):205 – 210, 2007.

[7] N. Khan, D. E. Goldgerg, and M. Pelikan. Multi-objective bayesian optimization algorithm. IlliGAL Report No. 2002009. Technical report, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2002.

[8] J. D. Knowles and D. Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Trans. on Evol. Computation*, 7(2):100–116, 2003.

[9] M. Laumanns and J. Ocenasek. Bayesian optimization algorithms for multi-objective optimization. In *Parallel Problem Solving from Nature - PPSN VII*, pages 298–307. Springer, 2002.

[10] M. Pelikan, K. Sastry, and D. E. Goldberg. Multiobjective hboa, clustering, and scalability. In *Proc. of the Genetic and Evolutionary Comp. Conf. - GECCO-2005*, pages 663–670. ACM, 2005.

[11] K. Sastry, M. Pelikan, and D. E. Goldberg. Decomposable problems, niching, and scalability of multiobjective estimation of distribution algorithms. IlliGAL Report No. 2005004. Technical report, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2005.

[12] D. Thierens and P. A. N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO-2011*, pages 617–624. ACM, 2011.

[13] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100. CIMNE, Barcelona, Spain, 2002.