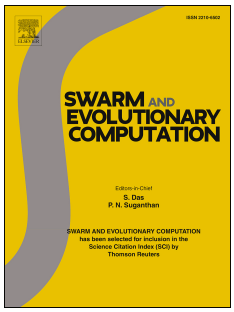
Accepted Manuscript



Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm with the interleaved multi-start scheme

Ngoc Hoang Luong, Han La Poutré, Peter A.N. Bosman

PII: S2210-6502(17)30476-5

DOI: 10.1016/j.swevo.2018.02.005

Reference: SWEVO 358

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 15 June 2017

Revised Date: 14 December 2017

Accepted Date: 7 February 2018

Please cite this article as: N.H. Luong, H. La Poutré, P.A.N. Bosman, Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm with the interleaved multi-start scheme, *Swarm and Evolutionary Computation BASE DATA* (2018), doi: 10.1016/j.swevo.2018.02.005.

Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm with the Interleaved Multi-start Scheme

Ngoc Hoang Luong[a,*], Han La Poutré[a,b], Peter A.N. Bosman[a]

[a]*Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands*
[b]*Delft University of Technology (TU Delft), Delft, The Netherlands*

Abstract

The Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm (MO-GOMEA) has been shown to be a promising solver for multi-objective combinatorial optimization problems, obtaining an excellent scalability on both standard benchmarks and real-world applications. To attain optimal performance, MO-GOMEA requires its two parameters, namely the population size and the number of clusters, to be set properly with respect to the problem instance at hand, which is a non-trivial task for any EA practitioner. In this article, we present a new version of MO-GOMEA in combination with the so-called Interleaved Multi-start Scheme (IMS) for the multi-objective domain that eliminates the manual setting of these two parameters. The new MO-GOMEA is then evaluated on multiple benchmark problems in comparison with two well-known multi-objective evolutionary algorithms (MOEAs): Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D). Experiments suggest that MO-GOMEA with the IMS is an easy-to-use MOEA that retains the excellent performance of the original MO-GOMEA.

Keywords: evolutionary algorithms, multi-objective optimization, linkage learning, optimal mixing, parameter settings, scalability

1. Introduction & Background

1.1. Multi-objective Combinatorial Optimization and Evolutionary Algorithms

Solving combinatorial optimization problems involves finding an optimal solution with respect to some objective function from a discrete set of feasible alternatives. Non-trivial problems often fall into the NP-hard class, where no polynomial-time deterministic algorithms are known so far. Metaheuristics, therefore, play an important role in providing high quality solutions within a reasonable computing time. Moreover, many real-world problems involve two or more conflicting objectives (i.e., the objective function is then a vector function), where a *utopian* solution that optimizes all objectives at the same time does not exist. Instead, the optimum of such a multi-objective problem is a set of equally favorable trade-off solutions that are all optimal in the sense that an improvement in any objective degrades other objectives. This article focuses on multi-objective combinatorial optimization problems.

A multi-objective optimization problem consists of m objective functions $f_i(\boldsymbol{x})$, $i \in \{0, 1, \ldots, m-1\}$ that, without loss of generality, all need to be maximized. We assume that solutions for the combinatorial optimization problem at hand involve l decision variables that comprise a discrete search space. In particular, we focus on Cartesian search spaces, meaning that for each variable we have a domain (e.g., a set of integers) and the space of entire solutions is the Cartesian product of these individual domains. We here restrict each variable domain to the binary domain $\mathbb{B} = \{0, 1\}$, but all methodologies presented in this article can be

*Corresponding author

easily extended to higher cardinality (see, e.g., [1]). A solution $\boldsymbol{x}$ can then be represented as a binary vector $\boldsymbol{x} = (x_0, x_1, \ldots, x_{l-1}) \in \boldsymbol{\Omega} = \times_{i=0}^{l-1}\mathbb{B}$. The objective value vector of $\boldsymbol{x}$ is $\boldsymbol{f}(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$.

Optimality in multi-objective optimization is defined by employing Pareto concepts (see, e.g., [2]). A solution $\boldsymbol{x}^0$ Pareto *dominates* a solution $\boldsymbol{x}^1$ (denoted $\boldsymbol{x}^0 \succ \boldsymbol{x}^1$) if and only if $f_i(\boldsymbol{x}^0) \geq f_i(\boldsymbol{x}^1), \forall i \in \{0, 1, \ldots, m-1\}$ and $\boldsymbol{f}(\boldsymbol{x}^0) \neq \boldsymbol{f}(\boldsymbol{x}^1)$. A solution $\boldsymbol{x}^0$ is Pareto optimal (or Pareto non-dominated) 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. Because the number of solutions on the Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ can be numerous (or even infinite in case of continuous optimization), it often suffices to find a good approximation of the Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$. The quality of the approximation front is evaluated based on both its proximity to the optimal front (i.e., as close as possible) and its diversity along the front (i.e., as well-spread as possible) [3].

Evolutionary algorithms (EAs) have been popular approaches for tackling multi-objective optimization problems [2, 4, 5]. Many EAs are population-based algorithms, which are well-suited in obtaining a whole approximation set of multiple non-dominated solutions in one run instead of having to run a single point-based algorithm, e.g., hill climbing, as many times as the desired number of solutions. Furthermore, EAs normally do not require problem-specific knowledge, which is hardly available in the context of black-box optimization, in their operation mechanism, resulting in their wide applicability to various problem domains. Problem-specific knowledge, if available, can be straightforwardly incorporated into EA operators (e.g., recombination or mutation) to enhance the search performance. Well-known multi-objective evolutionary algorithms (MOEAs), such as the Nondominated Sorting Genetic Algorithm II (NSGA-II) [6], the improved Strength Pareto Evolutionary Algorithm (SPEA2) [7], and the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) [8], have demonstrated their effectiveness in solving many real-world multi-objective problems. Nevertheless, there is still room for improvement in MOEA research. In this article, we particularly consider two of the important issues in MOEA design and application: *algorithm scalability* and *parameter settings*.

1.2. Scalability, Linkage Learning, and Gene-pool Optimal Mixing

Scalability requires that optimization algorithms maintain their effectiveness and efficiency when the problem size increases. Scalability thus corresponds to the general algorithmic notion of computational complexity. For combinatorial EAs, their scalability is typically highly dependent on their capability for mixing and preserving building blocks (i.e., good partial solutions) in the population to create new solutions [9]. Simple variation operators (e.g., uniform/1-/2-point crossover and mutation) of classic EAs normally need to be customized by problem-specific expert knowledge so that they do not disrupt building blocks too often during the optimization process. For example, it has been shown that Genetic Algorithm (GA) with simple variation operators cannot efficiently solve large instances of the Distribution Network Expansion Planning (DNEP) problem, i.e., the combinatorial optimization problem that involves choosing suitable electrical devices to economically and properly reinforce the capacity of an electricity distribution network regarding the predicted increase in electricity consumption [1]. The crossover and mutation operators of GA need to be modified to take into account specific features of electricity networks in order to obtain good DNEP solutions within acceptable computing time. Similarly, NSGA-II with simple variation operators fails to achieve good approximations for the Pareto-optimal fronts of the multi-objective DNEP problems [10].

However, problem-specific knowledge is not always available, as in the context of black-box optimization, or might be too complicated to be effectively exploited. In such situations, information about building blocks can be inferred from the working populations of EAs by linkage learning. Linkage learning is a procedure in which problem variables having some degree of dependency are detected, and they can then be considered as building blocks when new solutions are generated. Certain decomposable problems cannot be scalably solved by MOEAs that do not take into account the dependencies between problem variables (e.g., MOEAs that employ only classic recombination and mutation operators), requiring an exponential amount of computing budget for problems of large sizes [11]. Furthermore, it has been shown that effective

exploitation of linkage information helps EAs successfully tackle hard real-world combinatorial problems such as DNEP even in a black-box setting [1].

Estimation-of-distribution algorithms (EDAs) [12] address the scalability issue by replacing classic variation operators with model-based variation operators. In every generation, EDAs generate offspring solutions by sampling from a probabilistic model that encodes the distribution of promising (parent) solutions that currently exist in the population. Such probabilistic models often contain certain information about the dependencies among problem variables, ensuring offspring solutions are created with respect to the learned dependency structure. Notable EDAs for the multi-objective domain are the Multi-objective Adapted Maximum-Likelihood Model (MAMaLGaM) [13] for continuous variables and the Multi-objective Hierarchical Bayesian Optimization Algorithm (mohBOA) [11] for discrete variables.

Building a probabilistic model, however, is non-trivial and requires certain computing overheads in each generation. Furthermore, EDAs employing models of higher orders typically require larger population sizes so that the distribution of promising solutions can be properly learned [14]. It has been shown that such estimations of complete probability distributions are computationally expensive and might be redundant if solution variations can be effectively performed using only linkage information [9]. To this end, Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) [9] employs the so-called Optimal Mixing (OM) operator to intensively exploit the learned linkage information to iteratively improve existing candidate solutions, effectively creating offspring solutions of higher, or at least equal, quality. The linkage tree, a commonly-used linkage model for GOMEA, can be learned in $\mathcal{O}(Nl^2)$ time [15], compared to the typical $\mathcal{O}(Nl^3)$ time for EDAs employing probabilistic models of comparable complexity (where N is the population size and l is the number of problem variables). GOMEA requires far smaller population sizes and achieve better scalability in terms of the required numbers of function evaluations compared to classic GAs and EDAs in solving different benchmark problems [9] and real-world optimization problems [1]. The single-objective GOMEA has been extended with essential components of MOEAs, resulting in the multi-objective version, termed MO-GOMEA, which has been shown to retain the excellent performance of the original algorithm when tackling multi-objective problems [16, 10]. The design of MO-GOMEA is described in Section 2.

1.3. Objective-space Clustering

The quality of an approximation set of non-dominated solutions is often assessed based on its proximity and diversity in relation to the Pareto-optimal front [3, 17]. MOEAs try to achieve this two-fold goal by steering the population toward the Pareto-optimal front and preserving the diversity (especially in the objective space) at the same time. To this end, environmental selection methods are often employed, e.g., selection mechanisms based on both Pareto dominance and crowing distance (as in NSGA-II [6]) or density measure (as in SPEA2 [7]). However, such selection operators are insufficient to scalably obtain good approximation sets for some classes of decomposable problems (e.g., Zeromax-Onemax or Trap5-Inverse Trap5) [11, 18]. Also, it has been shown that NSGA-II did not manage to obtain approximation sets that are well-spread along Pareto-optimal fronts of the real-world multi-objective DNEP problem [10]. We note that linkage learning alone cannot ensure obtaining well-spread approximation sets because solutions in different regions of a Pareto-optimal front typically have different features and a single linkage model would not be able to capture all the specific structures of the whole front [16]. For example, a DNEP solution that minimizes the investment cost typically differs a lot from a DNEP solution that maximize the electricity network reliability.

State-of-the-art MOEDAs typically address this issue by employing mixture probability distributions. In every generation, promising candidate solutions are clustered, often in the objective space, and a probabilistic model is constructed for each cluster separately (e.g., in mohBOA [11] and in MAMaLGaM [13]); all the learned models are then used to generate offspring solutions. Similarly, MO-GOMEA [16] performs linkage learning separately for each cluster, and then use the learned linkage model of each cluster to improve the quality of the solutions in that cluster. Such cluster-based operation mechanisms have been found to be highly important to the scalability of MO-GOMEA, and setting the number of clusters can have significant impact on its performance [16].

1.4. Parameter Settings and the Interleaved Multi-start Scheme (IMS)

Although MO-GOMEA has shown excellent scalability compared to other MOEAs in solving different benchmark and real-world problems, practitioners are required to properly set its two most important parameters, namely the population size and the number of clusters [16]. In general, parameter settings characterize the capacity of an EA when solving a problem instance, and a certain capacity is needed for each EA to obtain results of a certain quality within a certain probability. For certain types of multi-objective problems, it has been proved that only population-based MOEAs can obtain the Pareto-optimal fronts efficiently and single individual-based MOEAs fail to achieve the same goal [19]. While the population holds a crucial role in any EAs, especially MOEAs, it is notoriously difficult to determine the proper population sizes for solving the problem instance under concern beforehand in practice. On the one hand, EAs cannot solve problems well using populations of inadequate sizes. On the other hand, if the population size is too big, EAs may overly diversify their search efforts and the allowed budget of fitness evaluations or running time is used up before good solutions are obtained. For example, it has been shown for the real-world DNEP problem, different EAs have different minimally-required population sizes for each problem instance [20]. Furthermore, too small population sizes result in premature convergence while too large population sizes incur unnecessarily large numbers of DNEP solution evaluations, which are computationally expensive as each evaluation involves power flow computations for the electricity network under concern [20].

Similarly, for MOEAs with cluster-based operation like MAMaLGaM and MO-GOMEA, if too few clusters are employed, certain parts of Pareto-optimal fronts might not be effectively obtained due to lack of dedicated search bias. If too many clusters are used, the population size would be excessively large as each cluster needs a certain size to operate properly, resulting in over-diversification of the search. Practitioners often need to run EAs many times with different parameter settings in a trial-and-error manner.

The key issue is to match the algorithm capacity to the problem complexity. Different EAs have different population sizing requirements for each class of optimization problems. Computing the proper population size of an EA for an arbitrary problem instance, $N = f(l)$ in which l is the number of variables, however, is not trivial. For simple problems, e.g., Onemax, $f(l) = O(\log l)$, but for much harder problems, $f(l)$ may need to be much bigger, up to exponential for NP-hard problems. Also, the optimal parameter setting is not possible to be determined beforehand especially in the context of black-box optimization, where problem knowledge is not available or too complicated to be efficiently exploited.

A different approach is to adapt the population size (and other parameter settings) along the optimization run as the hardness of the problem instance at hand is better recognized during the run. A population sizing-free mechanism has been proposed in [21, 22] to eliminate the population size setting for single-objective EAs with promising results. In essence, multiple populations of different sizes are operated at the same time in an interleaved fashion, in which populations of larger sizes are started later while populations of smaller sizes are allowed to perform more generations. Populations of smaller sizes are terminated when they converge or when a larger population obtains a better average fitness value, which indicates that the smaller populations are not necessary to be run anymore. This interleaved multi-start scheme (IMS), however, is not straightforward to be employed for MOEAs because populations of MOEAs hardly converge to a single solution and it is uninformative to compute and compare average fitness values in the multi-objective domain. There exist certain works (e.g., see [23]) to customize the IMS for NSGA-II but the results are primitive and the scalability issue is entirely overlooked.

In this article, we investigate how to customize the IMS for MO-GOMEA. First, we adapt the IMS to eliminate the requirement of setting the population size and the number of clusters of MO-GOMEA, enhancing the usability of MO-GOMEA. Second, we perform experiments to assess the performance of MO-GOMEA with the IMS on different unconstrained and constrained benchmark problems. For the comparison purpose, we also conduct experiments with an NSGA-II version and an MOEA/D version combined with our adapted IMS. We acknowledge that this approach is not necessarily optimal and many approaches to parameter tuning [24] and control [25] exist. However, the IMS approach is an easy-to-use framework which can be straightforwardly incorporated with any population-based EA and has been found to give high-quality results when combined with GOMEA [1]. Thus, the IMS approach can enhance the usability of MOEAs such as MO-GOMEA, facilitating the transferable use of MOEAs by practitioners inside and

```
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      F_j ← LearnLinkageModel(C_j)
11    for i ∈ {0, 1, ..., N − 1} do
12      C_j ← DetermineCluster(P_i, {C_0, C_1, ..., C_{k−1}})
13      if ¬IsExtremeCluster(C_j) then
14        O_i ← Multi-Objective-GenepoolOptimalMixing(P_i, C_j, F_j, A^t)
15      else
16        O_i ← Single-Objective-GenepoolOptimalMixing(P_i, C_j, F_j, A^t)
17    P ← O = {O_0, O_1, ..., O_{N−1}}
18    if f(A^t) ≠ f(A^{t−1}) then
19      t^NIS ← 0
20    else
21      t^NIS ← t^NIS + 1
```

Figure 1: Pseudo code for MO-GOMEA

outside the field.

The remainder of the article is organized as follows. For the purpose that this article can act as a standalone work, section 2 will re-address the key components of MO-GOMEA. Section 3 introduces all benchmark problems that we use to perform experiments. Section 4 shows how we customize the IMS for MO-GOMEA to eliminate the settings of the population size and the number of clusters parameters. Section 5 demonstrates the performance of MO-GOMEA with the IMS and the influence of mutation operators. Section 6 compares the performance of MO-GOMEA with NSGA-II and MOEA/D in the IMS framework and further discusses the IMS in a broader multi-objective optimization context. Section 7 discusses what makes MO-GOMEA with the IMS a scalable and practical MOEA. Section 8 concludes the article.

2. Multi-objective GOMEA

MO-GOMEA starts with a population $\mathcal{P}$ of N randomly-uniformly-generated candidate solutions. All initial solutions are evaluated for objective values. A clustering method is then employed to partition the population $\mathcal{P}$ (in the objective space) into k clusters $\mathcal{C}_j$'s ($j \in \{0, 1, \ldots, k-1\}$) of equal sizes (see Section 2.2). Linkage learning is performed in each cluster $\mathcal{C}_j$ to obtain a separate linkage model $\mathcal{F}_j$ (see Section 2.3). Finally, each existing (parent) solution $\boldsymbol{x}$ is evolved into a new (offspring) solution in an iterative manner by using the linkage model $\mathcal{F}_j$ of the cluster $\mathcal{C}_j$ to which $\boldsymbol{x}$ belongs to guide the variation operators (see Section 2.4). The offspring solutions make up the population for the next generation. MO-GOMEA is outlined in Figure 1.

2.1. Elitist Archive

The Pareto-optimal front might contain a numerous (or infinite) number of non-dominated solutions while the working population of MOEAs is normally limited. Many non-dominated solutions are, therefore, omitted due to the environmental selection or inadvertently discarded due to the stochastic EA operators [26]. It is thus beneficial to maintain a secondary population, termed the elitist archive, that is dedicated to keeping track of the overall status of the Pareto front of non-dominated solutions found during the search. MO-GOMEA employs an adaptive elitist archive implementation as presented in [27]. In general, a newly-generated solution is added into the archive if it is not dominated by any existing solution

in the archive. If there exists an archived solution having similar objective value vector, the new solution will replace the old one if such replacement increases the diversity of the archive in the decision space [16].

2.2. Clustering

In every generation, MO-GOMEA performs an efficient clustering method, called k-leader-means clustering, to partition the population into k clusters having equal sizes of c solutions in the objective space (for details, see [13]). Equal-sized clusters ensure that the same amount of search effort is used to approach each part of the Pareto-optimal front. We employ $c = \frac{2}{k}|\mathcal{P}|$, where $\mathcal{P}$ are the solutions being clustered, as in [13]. This large value of c makes neighboring clusters overlap, which reduces the chance that some solutions are not included in any cluster and increase the probability of obtaining an evenly-spread approximation set.

2.3. Linkage Learning

In every generation, a separate linkage model is learned for each cluster based on the solutions belonging to that cluster. MO-GOMEA employs the linkage tree (LT) structure as the linkage model to capture the dependencies among problem variables. Let L denote the set of indices of all l problem variables, i.e., $L = \{0, 1, \ldots, l-1\}$. An LT $\mathcal{F}$ can be represented as a Family of Subsets (FOS) of L, i.e., $\mathcal{F} = \{\boldsymbol{F}^0, \boldsymbol{F}^1, \ldots, \boldsymbol{F}^{|\mathcal{F}|-1}\}$ where $\boldsymbol{F}^i \subseteq L$. Each subset $\boldsymbol{F}^i$ can be considered as a linkage group of problem variables that exhibit some dependency relation. Such dependencies should be considered when performing variations (i.e., creating offspring solutions) because respecting linkage structures is important to the efficiency of EAs if the problem instance at hand truly exhibits these dependencies.

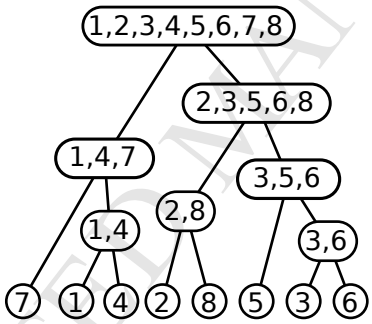


Figure 2: An example linkage tree with $l = 8$ problem variables.

An LT $\mathcal{F}$ can be constructed for each cluster in a bottom-up procedure called the Unweighted Pair Grouping Method with Arithmetic-mean (UPGMA) with an optimal implementation having $\mathcal{O}(Nl^2)$ time complexity [15]. First, all leaf nodes of $\mathcal{F}$ are singleton linkage groups, where each contains a single problem variable index, i.e., $\boldsymbol{F}^i = \{i\}, i \in L$. Then, intermediate nodes of $\mathcal{F}$ are created by iteratively merging two linkage groups that are closest to each other on the basis of some distance metric. Each time, a new linkage group is added into $\mathcal{F}$, and while its two constituent linkage groups are still kept in $\mathcal{F}$, they are not considered for further merging. That is, each intermediate node associates with a linkage group $\boldsymbol{F}^i$, where $\boldsymbol{F}^i = \boldsymbol{F}^j \cup \boldsymbol{F}^k$, $\boldsymbol{F}^j \cap \boldsymbol{F}^k = \emptyset$, $|\boldsymbol{F}^j| < |\boldsymbol{F}^i|$, and $|\boldsymbol{F}^k| < |\boldsymbol{F}^i|$. The merging procedure continues until the root node containing all problem variable indices is created. MO-GOMEA employs the Mutual Information (MI), which is computed over the solutions belonging to the same cluster, as the distance metric for UPGMA, where higher MI values denote closer distances. Figure 2 shows an example LT concerning 8 problem variables.

2.4. Gene-pool Optimal Mixing

MO-GOMEA performs a variation operator called Gene-pool Optimal Mixing (GOM) to improve each existing (parent) solution in a step-wise manner, transforming them into offspring solutions. GOM is guided by the linkage information encoded in the learned LT of each cluster. If an existing solution belongs to more than one cluster (due to the balanced k-leader-means clustering) or if it does not belong to

any cluster, it will be improved by using the LT of the cluster which has the closest Euclidean distance from the cluster mean to that solution.

The step-wise mechanism of GOM includes iterative improvement checks. How the improvement check is performed depends on which cluster the parent solution belongs to. For an m-objective problem, $k > m$ clusters needs to be employed, and in every generation, after performing clustering, m clusters will be assigned as *extreme* clusters while the remainders are *middle* clusters. An extreme cluster is the cluster that has the largest mean value of an objective (assuming all objectives are to be maximized), and such extreme cluster is dedicated to approaching the extreme region of the Pareto-optimal front corresponding to that objective. For middle clusters, GOM performs improvement checks on the basis of the Pareto dominance relation. For extreme clusters, GOM perform improvement checks on the basis of improvements in terms of the corresponding objective , i.e., single-objective improvements. Figure 3 illustrates this cluster-based operation of MO-GOMEA.

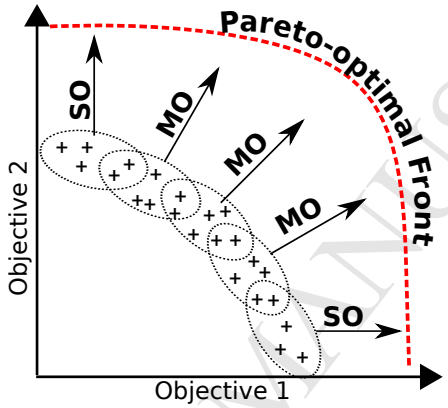


Figure 3: The cluster-based operation of MO-GOMEA. Different clusters approach different parts of the Pareto-optimal front. **MO**: Multi-objective GOM with improvement checks on the basis of Pareto dominance. **SO**: Single-objective GOM with improvement checks on the basis of the corresponding objective.

2.4.1. Multi-objective GOM

Let $\boldsymbol{p}$ be an existing (parent) solution that is to be improved, C_j be the cluster that $\boldsymbol{p}$ belongs to, and $\mathcal{F}_j$ be the LT learned over the solutions in C_j. First, the offspring solution $\boldsymbol{o}$ and a backup $\boldsymbol{b}$ are cloned directly from $\boldsymbol{p}$. Each linkage group in the LT $\mathcal{F}_j$ is then iterated in a random order. For every $\boldsymbol{F}^i \in \mathcal{F}_j$, another existing solution in C_j is randomly chosen as the donor solution $\boldsymbol{d}$. The problem variable values are then copied from $\boldsymbol{d}$ to the current $\boldsymbol{o}$ only at the indices indicated by $\boldsymbol{F}$, i.e., $\boldsymbol{F}^i$ acts as a crossover mask in the mixing. The partially-altered solution $\boldsymbol{o}$ is evaluated for its objective values and is compared against the backed-up $\boldsymbol{b}$. The changes are accepted and also updated into the backup $\boldsymbol{b}$ if any of the following conditions is satisfied: 1) the altered solution dominates the backup $\boldsymbol{o} \succ \boldsymbol{b}$; 2) the altered solution has the same objective values as the backup $\boldsymbol{f}(\boldsymbol{o}) = \boldsymbol{f}(\boldsymbol{b})$; 3) the altered solution is not dominated by any elitist archive member $\mathcal{A} \not\succ \boldsymbol{o}$. Otherwise, the changes are restored to the backup states. This procedure is repeated until all linkage groups in the LT $\mathcal{F}_j$ are iterated, and an offspring is then fully constructed, replacing the original parent $\boldsymbol{p}$ in the next generation.

An additional procedure called Forced Improvement (FI [16, 28]) is invoked when GOM does not manage to change a parent solution. FI is a second round of GOM but the donor solutions are selected randomly from the elitist archive. Furthermore, FI aims for strict improvement and only accepts the mixing step if the altered solution dominates the backup $\boldsymbol{o} \succ \boldsymbol{b}$, or if the altered solution is a truly new non-dominated solution $\mathcal{A}^t \not\succ \boldsymbol{o} \wedge \boldsymbol{f}(\boldsymbol{o}) \notin \boldsymbol{f}(\mathcal{A}^t)$. FI terminates as soon as a mixing step is accepted instead of iterating the whole LT. If FI does not manage to change the parent solution $\boldsymbol{p}$, it will be replaced by a random elitist archive member. It might happen that a plateau in the search landscape causes variable values of parent solutions to be changed back and forth without improving their objective values, and thus FI will not be

invoked. This can be partly recognized when the Pareto front formed by the elitist archive stays the same for too many generations, making a no improvement stretch (NIS). To mitigate such effect, FI is also invoked when NIS exceeds the threshold $1 + \lfloor \log_{10}(N) \rfloor$, which has been found to give good results in the single-objective domain [28]. The GOM and FI procedures are outlined in Figure 4.

```
MULTIOBJECTIVE-GENEPOOLOPTIMALMIXING(p, C, F, A^t)
 1  b ← o ← p; f(b) ← f(o) ← f(p); changed ← false
 2  for i ∈ {0, 1, ..., |F| − 1} do
 3    d ← PICKRANDOMDONORFROMCLUSTER(C)
 4    o_{F^i} ← d_{F^i}
 5    if o_{F^i} ≠ b_{F^i} then
 6      f(o) ← EVALUATEFITNESS(o); updated ← false; improved ← false
 7      if [A^t ⊁ o and f(o) ∉ f(A^t)] or
 8        [f(o) ∈ f(A^t) and ISDIVERSITYINCREASEDBYADDING(A^t, o)] then
 9          updated ← true; improved ← true
10      else if o ≻ b or f(o) = f(b) or A^t ⊁ o then
11          improved ← true
12      if updated then
13          A^t ← UPDATEELITISTARCHIVE(o)
14      if improved then
15          b_{F^i} ← o_{F^i}; f(b) ← f(o); changed ← true
16      else
17          o_{F^i} ← b_{F^i}; f(o) ← f(b)
18  if ¬changed or t^{NIS} > 1 + ⌊log_10(N)⌋ then
19    changed ← false
20    for i ∈ {0, 1, ..., |F| − 1} do
21      d ← PICKRANDOMDONORFROMELITISTARCHIVE(A^t)
22      o_{F^i} ← d_{F^i}
23      if o_{F^i} ≠ b_{F^i} then
24        f(o) ← EVALUATEFITNESS(o); updated ← false; improved ← false
25        if A^t ⊁ o and f(o) ∉ f(A^t) then
26            updated ← true; improved ← true
27        else if f(o) ∈ f(A^t) and ISDIVERSITYINCREASEDBYADDING(A^t, o) then
28            updated ← true
29        if o ≻ b then
30            improved ← true
31        if updated then
32            A^t ← UPDATEELITISTARCHIVE(o)
33        if improved then
34            b_{F^i} ← o_{F^i}; f(b) ← f(o); changed ← true
35        else
36            o_{F^i} ← b_{F^i}; f(o) ← f(b)
37      if changed then breakfor
38  if ¬changed then
39    d ← RANDOM(A^t); o ← d; f(o) ← f(d)
```

Figure 4: Pseudo code for multi-objective Optimal Mixing

2.4.2. Single-objective GOM

Pareto dominance improvements might not put enough pressure to efficiently approach the extreme regions of the Pareto-optimal front. MO-GOMEA thus explicitly operates the extreme clusters dedicatedly in the single-objective mechanism similarly to the original single-objective GOMEA [28]. If a parent solution $\boldsymbol{p}$ belongs to an extreme cluster with the corresponding objective f_i, GOM will perform improvement checks in terms of that objective only. Similarly, FI employs the solution $\boldsymbol{x}_{f_i}^{best}$ that currently has the maximum value in the objective f_i as the sole donor solution. The NIS counter, however, is still considered in terms of Pareto front improvements.

3. Benchmark Problems and Performance Evaluation

3.1. Problems

Because we will use benchmark problems throughout subsequent sections to test and decide on various design choices for MO-GOMEA, we present all problems that we consider in this article in this section first. We also describe how the Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ of each problem instance can be obtained, which is used to evaluate the performance of MOEAs (see Section 3.2).

3.1.1. Scalable Benchmark Problems

Zeromax-Onemax [11].

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

f_{Onemax} maximizes the number of 1-valued bits while f_{Zeromax} maximized the number of 0-valued bits, making every possible solution a Pareto-optimal solution. The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ has $l+1$ points $\boldsymbol{\mathcal{P}}_F = \{(i, l-i) \mid i \in \{0, 1, \ldots l\}\}$ lying on a straight line in the objective space. Each point on $\boldsymbol{\mathcal{P}_F}$ can be the image (i.e., the objective value vector) of multiple solutions, except for the two extreme points that map to exactly the all-1 string, that maximizes Onemax, and the all 0-string, that maximizes Zeromax. The niches of candidate solutions mapping to the extreme regions of $\boldsymbol{\mathcal{P}_F}$ are exponentially smaller than the niches of those map to the middle regions [11].

Trap5 - Inverse Trap5 [11].

Trap5 is the additively decomposable composition of the order-5 deceptive subfunctions such that the subfunctions are mutually exclusive.

$$f_{\text{Trap5}}(\boldsymbol{x}) = \sum_{i=0}^{(l/5)-1} f^{\text{sub}}_{\text{Trap5}}\left(\sum_{j=5i}^{5i+4} x_j\right) \tag{2}$$

where

$$f^{\text{sub}}_{\text{Trap5}}(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4-u & \text{if } u < 5 \end{cases} \tag{3}$$

Inverse Trap5 is evaluated in an opposite manner:

$$f^{\text{sub}}_{\text{Inverse-Trap5}}(u) = \begin{cases} 5 & \text{if } u = 0 \\ u-1 & \text{if } u > 0 \end{cases} \tag{4}$$

The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ has $l/5+1$ points $\boldsymbol{\mathcal{P}}_F = \{(5i+4(l/5-i), 5(l/5-i)+4i) \mid i \in \{0, 1, \ldots l/5\}\}$, lying on a straight line. Candidate solutions that have any subfunction evaluated to the value 5 in either Trap5 or Inverse Trap5 are Pareto-optimal solutions. Trap5 is often used to benchmark the ability of EAs in recognizing and respecting the linkage structures when performing variations.

Leading Ones Trailing Zeros (LOTZ).

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

$f_{\text{LO}}(\boldsymbol{x})$ maximizes the number of subsequent 1 bits at the beginning of $\boldsymbol{x}$ while $f_{\text{TZ}}(\boldsymbol{x})$ maximized the number of subsequent 0 bits at the end of $\boldsymbol{x}$. The Pareto-optimal front $\boldsymbol{\mathcal{P}_F}$ has $l+1$ points $\boldsymbol{\mathcal{P}}^i_F = \{(i, l-i) \mid i \in \{0, 1, \ldots l\}\}$, on a straight line, lying on a straight line. Candidate solutions that is composed of an all-1 substring concatenated to an all-0 substring is Pareto-optimal. The two extremes solutions are the all-1 string, that maximizes Leading Ones, and the all-0 string, that maximizes Trailing Zeros.

3.1.2. Multi-objective weighted MAXCUT

Definition.

Let $G = (V, E)$ a weighted undirected graph, where $V = (v_0, v_1, \ldots, v_{l-1})$ is the set of l vertices, and E is the set of edges (v_i, v_j) with associated weights w_{ij}'s. The weighted MAXCUT problem is defined as finding a maximum cut, which is a partition of l vertices into two disjoint subsets A and $B = V \setminus A$ such that the total weight of all edges (v_i, v_j) having $v_i \in A$ and $v_j \in B$ is maximized. We represent a cut as an l-bit binary string $\boldsymbol{x}$. Each bit x_i corresponds to a vertex $i \in \{0, 1, \ldots, l-1\}$, such that $x_i = 0$ indicates that the vertex $v_i \in A$ and $x_i = 1$ indicates that the vertex $v_i \in B$. Solving the weighted MAXCUT problem seeks to maximize the following objective function:

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

The multi-objective weighted MAXCUT problem is formed by solving a different MAXCUT instance for each objective. The instances have identical vertices but different edge weights.

Problem Instances.

Our benchmark consists of 4 bi-objective weighted MAXCUT instances of size $l = 12, 25, 50, 100$. Each instance is a fully connected graph having $\frac{1}{2}l(l-1)$ edges. The edge weights are set by following the approach described in [28]. This test set was also used previously in benchmarking MO-GOMEA [16]. For the 12-vertex and 25-vertex problem instances, we can easily obtain the true Pareto-optimal fronts $\boldsymbol{\mathcal{P}_F}$ by the enumeration method. For the 50-vertex and 100-vertex problem instances, because $\boldsymbol{\mathcal{P}_F}$'s are unknown, we replace them by the reference sets reported in [16], which were obtained based on running multiple experiments with well-considered parameter settings.

3.1.3. Multi-objective Knapsack

Problem Definition.

The multi-objective knapsack problem involves l items and m knapsacks. Each knapsack k has a specific capacity c_k. Each item i has a weight $w_{i,k}$ and a profit $p_{i,k}$ corresponding to each knapsack k. A solution of the knapsack problem can be encoded as a binary string $\boldsymbol{x} = (x_0, x_1, \ldots, x_{l-1}) \in \{0,1\}^l$, where each bit x_i corresponds to an item i, and $x_i = 1$ indicates that item i is selected. Selecting an item i in the multi-objective knapsack context means the item i is placed in every knapsack. A feasible solution is a selection of items such that the total weight does not exceed the capacity of any knapsack. The objectives are maximizing the profits of all knapsacks at the same time as follows.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x})) \\ \text{where} \quad & f_k(\boldsymbol{x}) = \sum_{i=0}^{l-1} p_{i,k} x_i \quad k = 0, \ldots, m-1 \\ \text{subject to} \quad & \sum_{i=0}^{l-1} w_{i,k} x_i \leq c_k \quad k = 0, \ldots, m-1 \end{aligned}$$

Constraint Handling.

If a solution violates the capacity constraint of any knapsack, we use the repair mechanism proposed in [29] to iteratively remove items until all constraints are satisfied. The item removal order follows the principle that the items with the lowest maximum profit/weight ratio should be discarded first. The maximum profit/weight ratio r_i of an item i is calculated as follows [29].

$$r_i = \max_{k=0}^{m-1} \left\{ \frac{p_{i,k}}{w_{i,k}} \right\}$$

This repair mechanism tries to satisfy the capacity constraints while diminishing the overall profit as little as possible by considering all knapsacks when removing items. We name it the multi-objective repair.

This multi-objective repair works well for approaching middle regions of the Pareto-optimal front, where trade-off solutions that balance all objectives are located. It can be intuitively noticed that Pareto-optimal solutions in the extreme regions, which maximize the profit of a specific knapsack, can be approached more efficiently if they are targeted by employing a repair mechanism dedicated to that knapsack. A repair function dedicated to a knapsack k also iteratively removes items until all constraints are satisfied but the item removal order biases toward knapsack k, such that items with lowest profit/weight ratio $r_{i,k} = p_{i,k}/w_{i,k}$ should be discarded first. We name such dedicated repair functions single-objective repairs. The question is which repair function should be used for a specific infeasible solution. The answer is straightforward for MO-GOMEA: middle-region clusters should employ the multi-objective repair while each extreme-region cluster should employ the single-objective repair that corresponds to its target objective.

Problem Instances.
We use the bi-objective knapsack problem instances of 100, 250, 500, 750 items proposed by [29]. True Pareto-optimal fronts $\mathcal{P}_{\boldsymbol{F}}$'s of the 100-, 250-, and 500-item problem instances have been reported in [29]. For the 750-item problem instance, $\mathcal{P}_{\boldsymbol{F}}$ is unknown and we replace it by a reference set, which is created by combining all Pareto fronts obtained by all optimizers in all runs.

3.2. Performance Evaluation

We evaluate the performance of different MO-GOMEA variants based on two criteria: front quality and scalability. The latter can be studied if the optimum of a problem is known and can be reached. Otherwise, more performance indicators are needed to describe the quality of the resulting front.

3.2.1. Performance Indicators

The result of an MOEA solving a multi-objective problem is a so-called *approximation set* $\mathcal{S}$ of non-dominated solutions that approximates the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$. In this work, the contents of the elitist archive when the optimization algorithm terminates are considered as the approximation set obtained by that algorithm. We employ 4 performance indicators to assess the quality of result approximation sets.

1. **Front Occupation (FO)** indicates the number of solutions in an approximation set $\mathcal{S}$. A larger FO value is preferable.
2. **Maximum Spread (MS)**, in the bi-objective case, measures the Euclidean distance between the 2 extreme solutions of an approximation set $\mathcal{S}$ in the objective space [30]. This indicator relates to the diversity of the approximation set. A larger MS value is preferable.
3. **Generational Distance (GD)** measures the average distance in the objective space from a solution in an approximation set $\mathcal{S}$ and to its nearest solution in the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$.

$$GD(\mathcal{S}, \mathcal{P}_{\boldsymbol{F}}) = \frac{1}{|\mathcal{S}|} \sum_{\boldsymbol{f}^0 \in \mathcal{S}} \min_{\boldsymbol{x} \in \mathcal{P}_{\boldsymbol{F}}} \{d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}^0)\} \tag{7}$$

   where $d(\cdot, \cdot)$ computes the Euclidean distance. GD relates to the proximity of an approximation set $\mathcal{S}$ to the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$. A smaller GD value is preferable and the best GD value 0 indicates that all solutions in $\mathcal{S}$ are Pareto optimal, i.e., all their objective value vectors belong to the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$. However, the GD indicator does not take into account the diversity: the best GD value 0 can be obtained with an approximation set $\mathcal{S}$ ($|\mathcal{S}| = 1$) that contains only a single Pareto-optimal solution in $\mathcal{P}_{\boldsymbol{F}}$.
4. **Inverted Generational Distance (IGD)** measures the average distance in the objective space from a solution in the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$ to its nearest solution in an approximation set $\mathcal{S}$ [3].

$$IGD(\mathcal{S}, \mathcal{P}_{\boldsymbol{F}}) = \frac{1}{|\mathcal{P}_{\boldsymbol{F}}|} \sum_{\boldsymbol{f}^0 \in \mathcal{P}_{\boldsymbol{F}}} \min_{\boldsymbol{x} \in \mathcal{S}} \{d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}^0)\} \tag{8}$$

   A smaller IGD value is preferable and the best IGD value 0 indicates that all solutions in the Pareto-optimal front $\mathcal{P}_{\boldsymbol{F}}$ are found. The IGD indicator takes into account both proximity (i.e., how close $\mathcal{S}$

is to the front $\mathcal{P}_{\boldsymbol{F}}$) and diversity (i.e., how well-spread $\mathcal{S}$ is along the front $\mathcal{P}_{\boldsymbol{F}}$) of the approximation set $\mathcal{S}$. We note that a slightly smaller value for this indicator in itself doesn't necessarily mean that the corresponding approximation front found by an MOEA is better than the one found by another MOEA [31]. However, in this work, we perform the trend analysis of algorithm performance in terms of the distance between the obtained approximation set and the Pareto-optimal front over time. Therefore, we put more emphasis on the IGD indicator in evaluating the performance of MOEAs, and the most important issue is the convergence of this IGD value to 0.

In order to compute the GD and IGD indicators, the true Pareto-optimal fronts are required. The number of solutions in $\mathcal{P}_{\boldsymbol{F}}$ can be infinite (e.g., in the continuous case). However, in this work, we focus on combinatorial multi-objective optimization. All the test problems have finite Pareto-optimal fronts, which are available for the benchmark purpose. The Pareto-optimal fronts $\mathcal{P}_{\boldsymbol{F}}$ of the problem Zeromax-Onemax, Trap5-Inverse Trap5, and LOTZ can be analytically calculated as described in Section 3.1.1. The Pareto-optimal fronts of MAXCUT and knapsack problem instances can be approximated as in Sections 3.1.2 and 3.1.3, respectively, and can be found in the Supplementary Materials.

3.2.2. Scalability

We perform the trend analysis of time required to obtain the Pareto-optimal fronts over problem sizes. We evaluate the performance of each optimizer in solving these 3 scalable benchmark problems by measuring the average number of evaluations until the whole Pareto-optimal fronts are obtained (i.e., when $IGD = 0$) over 100 independent runs. On the other hand, the true structural decompositions of MAXCUT and knapsack problem instances are unknown. Because MAXCUT and knapsack are also (NP-)hard problems, it is not expected that every single run of any optimizer can obtain all points on the Pareto-optimal fronts in polynomial time. Therefore, we can only observe how good of an approximation can be achieved.

For MAXCUT and knapsack problem instances, each optimizer is run 100 times independently and the IGD indicator values are recorded at some intervals during the optimization process until the allowed budget of fitness evaluations is used up. The average IGD indicator values over these 100 runs indicate the average convergence performance of that optimizer. To support our conclusion from experimental results, we perform the Mann-Whitney-Wilcoxon statistical hypothesis test for equality of medians with $p < \alpha = 0.05$ to see whether the final result obtained by one optimizer is statistically different from that of another optimizer.

4. Combining MO-GOMEA with the Interleaved Multi-start Scheme (IMS)

We employ the IMS to eliminate the requirement of setting the population size and the number of clusters parameters of MO-GOMEA. We propose different configurations for IMS and perform experiments to verify their performance. Based on the experimental results, we pick the configuration that has the most similar performance compared to MO-GOMEA where the population size is set to its ideal value for each problem. We employ two scalability benchmark problems, namely Zeromax-Onemax and Trap5-Inverse Trap5. The true Pareto-optimal fronts of these two problems can be easily calculated, which is convenient for comparing performance of different MOEAs. We do not employ the LOTZ problem here for benchmarking because, as reported in [16], it is much more efficient to solve LOTZ by mutation or Pareto-front local search, which do not exist in the standard version of MO-GOMEA. We will return to LOTZ in Section 5 when we discuss the influence of mutation operators.

4.1. Eliminating the Population Size Parameter Setting

4.1.1. Customizing the IMS for the Population Size Parameter

A parameter-less Genetic Algorithm (P-GA) was proposed in [21] and [32], in which multiple populations P_i's of different sizes N_i's operate in an interleaved fashion as follows. P-GA starts with the first population P_0 of some small size N_0. Next, every population P_{i+1} is created by doubling the size of the previous population P_i, i.e., $N_{i+1} = 2N_i$ for $i \geq 0$. For every base $b = 4$ generations of population P_i, P-GA runs 1 generation of population P_{i+1}. In other words, population P_{i+1} executes a generational step

every b-th generation of population P_i. When a population P_i converges, it will be terminated because, for the sake of simplicity, P-GA does not employ mutation operators. Converged populations, therefore, cannot explore the search space any further. Additionally, because smaller populations are given more fitness evaluations, if the average fitness of a population P_i is less than that of a larger population P_j ($j > i$) which is started later, then that population P_i is regarded as costly and inefficient, and should be terminated as well. Such interleaved multi-start scheme (IMS) has also been experimented in [22, 33] but with the generation base $b = 2$ so that every population is given the same number of fitness evaluations. Figure 5 shows the pseudo-code of the scheme.

```
INTERLEAVED MULTI-START SCHEME (IMS)
 1  P_0 ← INITIALIZEPOPULATION(N_0)
 2  generations[0] ← 0
 3  max_population_index ← 0
 4  i ← 0
 5  while WITHINALLOWEDCOMPUTINGBUDGET() do
 6    EXECUTEONEGENERATION(P_i)
 7    generations[i] ← generations[i] + 1
 8    if generations[i] mod b = 0 then
 9      i ← i + 1
10      if i > max_population_index then
11        P_i ← INITIALIZEPOPULATION(2 × N_{i−1})
12        generations[i] ← 0
13        max_population_index ← i
14    else
15      i ← 0
```

Figure 5: Interleaved Multi-start Scheme (IMS) [32]

While the above IMS works well for single-objective optimization, it requires some adaptations for the multi-objective domain. First, we do not necessarily need to employ the termination criterion of converged populations because MOEA populations normally do not converge to a single solution. Second, we do not employ the termination criterion of small populations based on average fitness values. Calculating the average fitness value for a population is uninformative in the multi-objective context. All populations of MO-GOMEA are thus kept running without termination of small populations. However, we will further discuss this issue in Section 7. Third, instead of establishing a race among populations as in P-GA, we share the elitist archive among all running populations so that the final approximation set is contributed from all populations of different sizes.

4.1.2. Results

We put MO-GOMEA into the IMS with two configurations for the generation base $b = 2$ and $b = 4$. For the sake of simplicity, we fix the number of clusters $k = 5$, which was previously found to give good results [16]. We employ these 2 MO-GOMEA variants to solve Zeromax-Onemaz and Trap5-Inverse Trap5 with different sizes $l = 25, 50, 100, 200, 400$. For every problem instance, we run each MO-GOMEA variant 100 times independently and obtain the number of fitness evaluations until the whole Pareto-optimal front is found in each run. The Mann-Whitney-Wilcoxon test with $p < \alpha = 0.05$ is performed to check for statistical significance.

Figure 6 shows the performance of MO-GOMEA with the IMS having two different generation bases $b = 2$ and $b = 4$ compared to the MO-GOMEA using the optimal population size settings reported in [16]. The IMS with base 4 is slightly better than the IMS with base 2 on solving Zeromax-Onemax ($p = 0.97$ for problem size $l = 25$, $p < 0.001$ for other problem sizes), but the differences are small and their scalability graphs are similar. The IMS with base 2 is significantly better than the IMS with base 4 in solving Trap5-Inverse Trap5 ($p < 0.001$ for all problem sizes). This is due to the fact that diversity maintenance is of higher importance for multi-objective EAs compared to single-objective EAs. Smaller generation bases are faster in introducing larger populations with more diverse information. Both MO-GOMEAs with the IMS having base 2 and base 4 show some overhead compared to the MO-GOMEA using the optimal populations as a

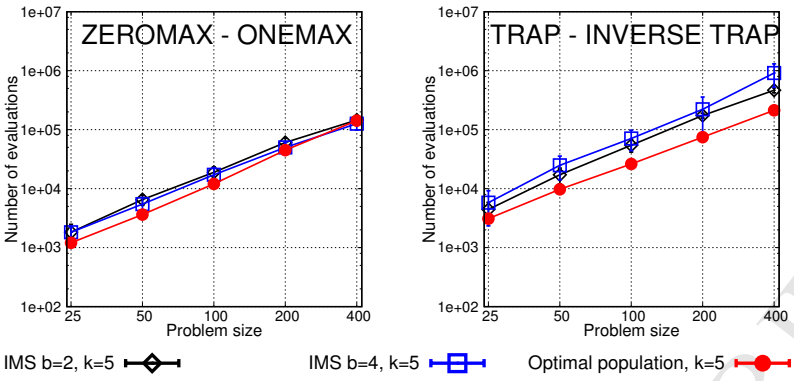


Figure 6: Experiments on eliminating the population size parameter for MO-GOMEA. Horizontal axis: Problem size. Vertical axis: The number of evaluations until the whole Pareto-optimal front is obtained: mean & standard devidation.

few generations are needed before the populations with proper sizes are initialized. Based on the results in Figure 6, we choose the IMS with base 2 for all experiments with MO-GOMEA in the remaining sections.

4.2. Eliminating the Number of Clusters Parameter Setting

4.2.1. Customizing the IMS for the Number of Clusters Parameter

In this section, we aim to let IMS control the number of clusters k of MO-GOMEA. Every population P_i is characterized by its population size N_i and the number of clusters k_i. We propose and experiment with three different IMS configurations for k as follows.

1. The first mechanism $[k = 1 \rightarrow +m]$ starts with the initial population P_0 of some small size N_0 (we here use $N_0 = 8$) and the number of clusters $k_0 = 1$ (i.e., no population clustering). For every odd-indexed population P_i ($i = 1, 3, 5, \ldots$), the population size N_i doubles the size of the preceding population P_{i-1} while the number of clusters k_i remains the same, i.e., $N_i = 2N_{i-1},\ k_i = k_{i-1}$. For every even-indexed population P_i ($i = 2, 4, 6, \ldots$), the population size N_i equals the size of the preceding population P_{i-1} while the number of clusters k_i increases by m clusters (m is the number of objectives), i.e., $N_i = N_{i-1},\ k_i = k_{i-1} + m$.
2. The second mechanism $[k = m + 1 \rightarrow +(m + 1)]$ starts with the initial population P_0 of size N_0 and the number of clusters $k_0 = m + 1$ (m is the number of objectives). It has been shown that MO-GOMEA works better when employing population clustering with single-objective optimization for m extreme clusters and multi-objective optimization for middle clusters [16], which means every population should have at least $m + 1$ clusters. For every odd-indexed population P_i ($i = 1, 3, 5, \ldots$), the population size N_i doubles the size of the preceding population P_{i-1} while the number of clusters k_i remains the same, i.e., $N_i = 2N_{i-1},\ k_i = k_{i-1}$. For every even-indexed population P_i ($i = 2, 4, 6, \ldots$), the population size N_i equals the size of the preceding population P_{i-1} while the number of clusters k_i increases by $(m + 1)$ clusters (m is the number of objectives), i.e., $N_i = N_{i-1},\ k_i = k_{i-1} + (m + 1)$.
3. The third mechanism $[k = m + 1 \rightarrow +1]$ starts with the initial population P_0 of size N_0 and the number of clusters $k_0 = m + 1$ (m is the number of objectives). For every succeeding population P_i ($i = 1, 2, 3, \ldots$), the population size N_i doubles the size of the preceding population P_{i-1} while the number of clusters k_i increases by 1 cluster, i.e., $N_i = 2N_{i-1},\ k_i = k_{i-1} + 1$.

4.2.2. Results

Figure 7 shows the experimental results on solving Zeromax-Onemax and Trap5-Inverse Trap5 with the population-sizing-free MO-GOMEA (IMS $b = 2$) with the fixed $k = 5$ and its three MO-GOMEA variants having different adaptive mechanisms for k. For every problem instance, we run each optimizer 100 times independently and obtain the number of evaluations until the whole Pareto-optimal front is found in each

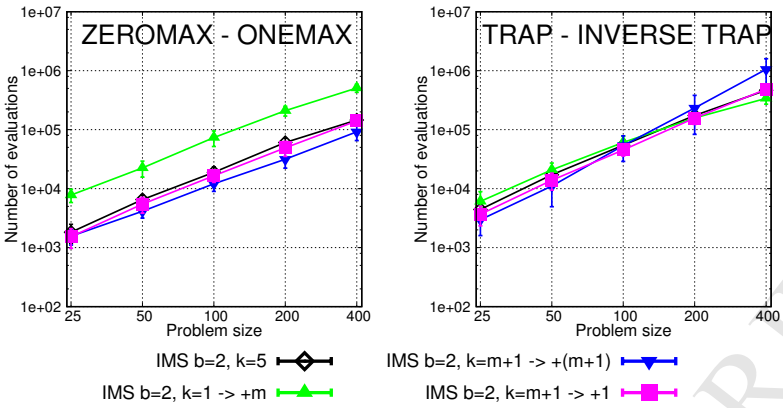


Figure 7: Experiments on eliminating the population size parameter for MO-GOMEA. Horizontal axis: Problem size. Vertical axis: The number of evaluations until the whole Pareto-optimal front is obtained: mean & standard deviation.

run. The differences in the numbers of evaluations for all MO-GOMEA variants solving each problem instance are found to be statistically significant ($p < 0.001$, except in the cases of Zeromax-Onemax $l = 25$, where $p = 0.91$, and Trap5-Inverse Trap5 $l = 100$, where $p = 0.08$). The first variant MO-GOMEA $1 \rightarrow +m$ has a slightly better performance in solving Trap5-Inverse Trap5 but the worst performance in case of Zeromax-Onemax. The second variant MO-GOMEA $m{+}1 \rightarrow +(m{+}1)$, on the other hand, performs slightly better in solving Zeromax-Onemax but it has the worst scalability in case of Trap5-Inverse Trap5. The third variant MO-GOMEA $m + 1 \rightarrow +1$ has balanced results on all test problems and also the most similar performance with the base MO-GOMEA $k = 5$. Therefore, we suggest that the $m + 1 \rightarrow +1$ configuration is the most suitable mechanism (among all methods tested here) to let IMS control the number of clusters parameter of MO-GOMEA. From now on, we refer to this MO-GOMEA variant with the IMS $b = 2$ and $k = m + 1 \rightarrow +1$ as the standard implementation of MO-GOMEA.

5. Performance of the MO-GOMEA with the IMS and the Influence of Mutation Operators

In this section, we conduct experiments on all benchmark problems (as presented in Section 3) to study the performance of our newly created MO-GOMEA with the IMS and the influence of the use of different mutation operators.

5.1. Design of Mutation Operators

MO-GOMEA works well without any mutation operators. However, empirical results showed that mutation could be beneficial to the performance of MO-GOMEA on some problems [16]. We here implement mutation as an additional component that practitioners can easily switch on or off as desired. At every mixing step during the GOM procedure, mutation could be performed after copying values from a donor to the current solution and before fitness evaluation of the intermediate solution. Mutation is applied independently with some probability p_m only on the problem variables indicated in the linkage set $\boldsymbol{F}^i$ used at the mixing step under concern. We propose 2 mutation operators: weak mutation and strong mutation. Weak mutation uses a fixed mutation probability $p_m = 1/l$ (l is the number of problem variables). Strong mutation uses an adaptive mutation probability $p_m = 1/l_{\boldsymbol{F}^i}$, where $l_{\boldsymbol{F}^i} = \left|\boldsymbol{F}^i\right|$ is the number of problem variables in a linkage set $\boldsymbol{F}^i$, i.e., the linkage set corresponding to the positions in the current solution whose values have just been replaced by those copied from the donor.

5.2. Scalable Benchmark Problems

Figure 8 shows that MO-GOMEA with the strong mutation operator is the fastest solver for Zeromax-Onemax (the performance gaps are found to be statistically significant with $p < 0.001$ for all problem

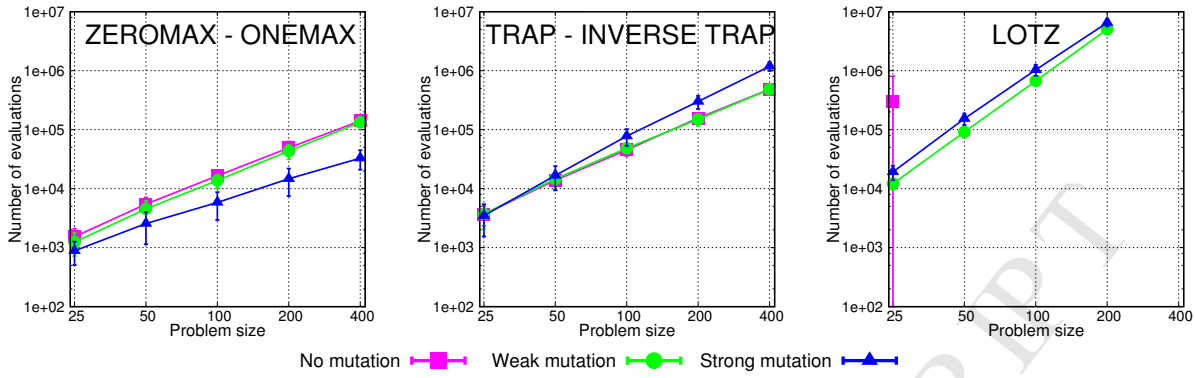


Figure 8: Performance of MO-GOMEA without mutation and with weak/strong mutation on scalable benchmark problems. Horizontal axis: Problem size. Vertical axis: The number of evaluations until the whole Pareto-optimal front is obtained: mean & standard deviation.

sizes). As all problem variables are independent from each other, linkage learning is not required to solve Zeromax-Onemax and a mutation operator with high mutation probability p_m will help obtain the whole Pareto-optimal front quicker by simply performing bit-flips to try out different alternatives without disrupting any building blocks. Such aggressive mutation operators, however, worsen the performance of MO-GOMEA on problems that require linkage learning such as Trap5-Inverse Trap5 (the differences are statistically significant with $p < 0.05$ for all problem sizes). Weak mutation has little influence on the scalability of MO-GOMEA in solving Zeromax-Onemax and Trap5-Inverse Trap5 because its mutation probability becomes increasingly smaller as the problem size increases. While MO-GOMEA with weak mutation performs slightly better than MO-GOMEA without mutation in solving Zeromax-Onemax ($p < 0.05$ for all problem sizes), there is no statistically significant difference between the two optimizers in solving Trap5-Inverse Trap5 ($p = 0.57, 0.15, 0.93,$ and 0.72 for problem sizes $l = 25, 50, 200,$ and 400, respectively).

Figure 8 shows that MO-GOMEA without mutation operators has difficulties in obtaining the whole Pareto-optimal fronts of LOTZ problem instances, especially in approaching 2 extreme solutions (i.e., the all-1 string and the all-0 string). As discussed in [16], only leading 1 bits and trailing 0 bits contribute to the objective values during the optimization process. It can be seen that all solutions containing trailing 1 bits and leading 0 bits can easily be dominated and replaced by any solutions ending with a 0 and beginning with a 1. Those trailing 1s and leading 0s, however, are essential to the construction of the extreme solutions at the later stage of the optimization process when they meet the leading 1s and trailing 0s. GOM, therefore, cannot find any donor with trailing 1s nor leading 0s remaining in the populations when necessary. This problem, however, can be alleviated by employing mutation operators. Both weak and strong mutation operators significantly improve the performance of MO-GOMEA on solving the LOTZ problem by bringing back the prematurely disappeared bit values. For LOTZ, MO-GOMEA with weak mutation performs slightly better than the variant with strong mutation (the differences are statistically significant with $p < 0.001$ for all problem sizes).

While these 3 benchmarks are convenient for scalability benchmarking because all Pareto-optimal solutions can be computed analytically, their Pareto-optimal fronts always have the shape of a straight line and they therefore do not resemble real-world optimization problems. In the following section, we will consider the MAXCUT and knapsack problems, which can be used to model many real-world problems.

5.3. MAXCUT Results

Table 1 shows the values of the 4 performance indicators which are evaluated based on the result approximation sets obtained at the end of 100 runs of each MO-GOMEA variant solving 4 MAXCUT problem instances. For small problem instances $l = 12, 25$, all performance indicator values are the same or the differences are statistically insignificant ($p > 0.31$). MO-GOMEA with strong mutation obtains the best IGD

Table 1: MAXCUT problems. Means and standard deviations (in brackets) of the 4 performance indicators evaluated on the result approximation sets of 100 runs of MO-GOMEA without mutation, MO-GOMEA with weak mutation, and MO-GOMEA with strong mutation. The best mean value of each indicator for each problem instance is presented in bold.

Instance	Algorithm	FO	MS	GD	IGD
12	MO-GOMEA no mutation	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MO-GOMEA weak mutation	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MO-GOMEA strong mutation	**6** (0.0)	**1.414214** (0.000000)	**0.000000**(0.000000)	**0.000000** (0.000000)
25	MO-GOMEA no mutation	**17** (0.1)	**1.414214** (0.000000)	**0.000000** (0.000000)	0.000030 (0.000297)
	MO-GOMEA weak mutation	**17** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MO-GOMEA strong mutation	**17** (0.1)	**1.414214** (0.000000)	0.000029 (0.000287)	0.000059 (0.000411)
50	MO-GOMEA no mutation	**42** (0.8)	**1.414214** (0.000000)	0.000764 (0.000749)	0.001001 (0.000535)
	MO-GOMEA weak mutation	**42** (0.8)	**1.414214** (0.000000)	**0.000655** (0.000696)	**0.000811** (0.000515)
	MO-GOMEA strong mutation	**42** (0.8)	**1.414214** (0.000000)	0.001116 (0.000771)	0.001252 (0.000620)
100	MO-GOMEA no mutation	122 (3.8)	1.394845 (0.051454)	0.003229 (0.002474)	0.004044 (0.002234)
	MO-GOMEA weak mutation	122 (3.9)	1.387425 (0.040047)	**0.002812** (0.001962)	0.003729 (0.001915)
	MO-GOMEA strong mutation	**123** (3.1)	**1.413599** (0.035315)	0.002899 (0.003248)	**0.003324** (0.002986)

value for the case $l = 100$ ($p < 0.01$) while the variant with weak mutation obtains the best IGD value for the case $l = 50$ ($p = 0.01$). The differences in FO, MS, and GD values obtained by different MO-GOMEA variants are either negligible or statistically insignificant.

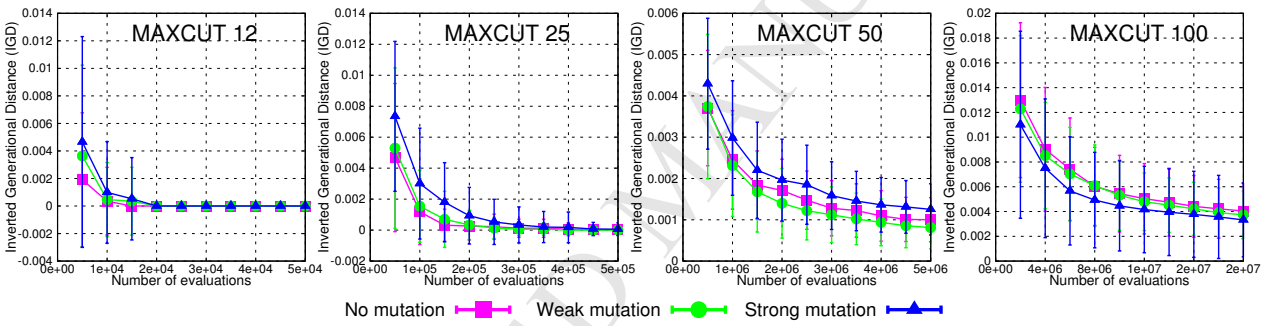


Figure 9: Average IGD convergence performance of MO-GOMEA without mutation and with weak/strong mutation on MAXCUT. Horizontal axis: number of evaluations (both objectives per evaluation). Vertical axis: IGD values: mean & standard deviation.

Figure 9 shows the average IGD convergence performance of MO-GOMEA and the influence of weak and strong mutation in solving 4 MAXCUT problem instances over time (in terms of the number of evaluations). The weak mutation operator has little or no influence on the performance of MO-GOMEA, and the differences in the final IGD results are found to be statistically insignificant in the cases of $l = 12, 25, 100$ ($p > 0.31$). The strong mutation operator shows no considerable impact in the cases of $l = 12, 25$, improves the convergence in the case $l = 100$ but slightly worsens the performance in the case $l = 50$. Nevertheless, all the performance gaps between MO-GOMEA without mutation and MO-GOMEA with mutation are small. In these MAXCUT problem instances, the linkage information-guided solution recombination of GOM is effective enough for MO-GOMEA to explore the search space without requiring any mutation operators.

5.4. Knapsack Results

Table 2 shows the values of the 4 performance indicators which are evaluated based on the result approximation sets obtained at the end of 100 runs of each MO-GOMEA variant solving 4 knapsack problem instances. For the cases $l = 100, 250$, MO-GOMEA with weak mutation obtains the best FO values, but the differences are not significant ($p > 0.18$), and the best IGD values, where the differences are statistically significant ($p < 0.03$). For the cases $l = 500, 750$, MO-GOMEA without mutation obtains the best FO and IGD values ($p < 0.001$). In almost all cases, MO-GOMEA without mutation also achieves the best GD scores. MO-GOMEA with strong mutation obtains the best MS values ($p < 0.001$, except for case $l = 100$, where

Table 2: Knapsack problems. Means and standard deviations (in brackets) of the 4 performance indicators evaluated on the result approximation sets of 100 runs of MO-GOMEA without mutation, MO-GOMEA with weak mutation, and MO-GOMEA with strong mutation. The best mean value of each indicator for each problem instance is presented in bold.

Instance	Algorithm	FO	MS	GD	IGD
100	MO-GOMEA no mutation	105 (3.7)	1.408707 (0.020036)	0.001456 (0.000408)	0.002504 (0.000598)
	MO-GOMEA weak mutation	**106** (3.1)	1.410399 (0.013204)	**0.001234** (0.000324)	**0.002183** (0.000386)
	MO-GOMEA strong mutation	93 (4.3)	**1.411914** (0.022381)	0.003044 (0.000783)	0.004580 (0.000694)
250	MO-GOMEA no mutation	259 (14.6)	1.269258 (0.035736)	**0.004828** (0.000646)	0.005905 (0.000656)
	MO-GOMEA weak mutation	**260** (13.0)	1.301241 (0.033348)	0.004942 (0.000718)	**0.005699** (0.000668)
	MO-GOMEA strong mutation	175 (11.5)	**1.364569** (0.034394)	0.015667 (0.001537)	0.015894 (0.001653)
500	MO-GOMEA no mutation	**369** (19.6)	1.214116 (0.029429)	**0.008235** (0.000864)	**0.009419** (0.000913)
	MO-GOMEA weak mutation	356 (16.1)	1.243492 (0.024079)	0.009474 (0.000788)	0.009995 (0.000682)
	MO-GOMEA strong mutation	242 (16.1)	**1.335861** (0.026243)	0.028869 (0.002394)	0.029176 (0.002596)
750	MO-GOMEA no mutation	**523** (24.0)	1.309775 (0.020059)	**0.007297** (0.000632)	**0.008014** (0.000697)
	MO-GOMEA weak mutation	481 (20.2)	1.325757 (0.021018)	0.009740 (0.000828)	0.010362 (0.000978)
	MO-GOMEA strong mutation	339 (20.8)	**1.382429** (0.018137)	0.032002 (0.002080)	0.034977 (0.002497)

$p > 0.66$), but the IGD values of its result approximation sets are the worst among all variants ($p < 0.001$ for all cases), i.e., its result fronts are far from the reference Pareto-optimal fronts.

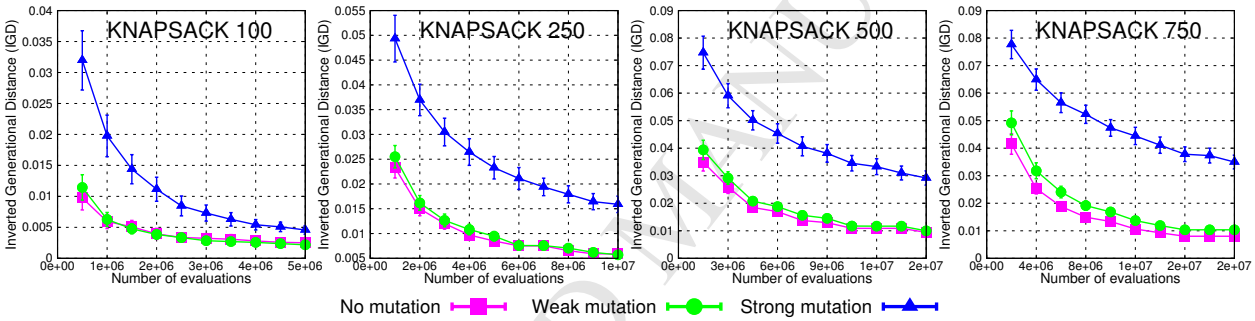


Figure 10: Average IGD convergence performance of MO-GOMEA without mutation and with weak/strong mutation on knapsack. Horizontal axis: number of evaluations (both objectives per evaluation). Vertical axis: IGD values: mean & standard deviation.

Figure 10 shows the average IGD convergence performance of MO-GOMEA and the influence of mutation operators in solving 4 multi-objective knapsack problem instances over time (in terms of the number of evaluations). MO-GOMEA without mutation has the best convergence performance in the cases of $l = 500$ and 750. The weak mutation operator exhibits little effects while the strong mutation operator severely degrades the performance of MO-GOMEA. Mutation operators with high mutation rates p_m can disrupt important building blocks acquired by the linkage learning and GOM procedures.

While mutation is a crucial variation operator of other MOEAs, it is not a required component for MO-GOMEA. Among all benchmark results, mutation operators only substantially deliver positive improvements when solving the LOTZ problem. Therefore, we conclude that the standard version of MO-GOMEA will not contain any mutation operator. Instead, the type of mutation and the probability of mutation should be, much like local search operators, considered as highly problem-specific and, therefore, are not included as a default parameter of MO-GOMEA.

6. Comparison with other MOEAs and the Influence of Terminating (Inefficient) Small Populations

6.1. NSGA-II and MOEA/D with the IMS

For comparison purposes, we extend NSGA-II [6] and MOEA/D [8] with the IMS that we employed for MOEAs in Section 4. We conduct experiments for NSGA-II and MOEA/D with both IMS generation base $b = 2$ and $b = 4$ as we do not yet know which values would be more appropriate for the operation of

NSGA-II and MOEA/D. Because the standard versions of NSGA-II and MOEA/D do not employ objective-space population clustering, they do not have the number-of-clusters parameter as in MO-GOMEA. While mutation is not a required component in MO-GOMEA, it is crucial for the search mechanism of NSGA-II [16]. Therefore, we keep the mutation operator with probability $p_m = 1/l$ (l is the number of problem variables) in NSGA-II and MOEA/D as in their original implementations [6, 8]. While the original NSGA-II does not have an elitist archive, for fair comparison, we equip both NSGA-II and MOEA/D with an elitist archive similar to the one of MO-GOMEA.

6.2. A Termination Criterion for (Inefficient) Small Populations

Research on population sizing-free EAs for single-objective optimization, as in [21, 32], emphasized the importance of terminating smaller populations when they are found to be less efficient compared to a larger population. More specifically, when a population of size N_i reaches an average fitness at least as good as the average fitness of the population of size N_{i-1}, all populations of size N_k with $k < i$ will be terminated [32]. It is not directly clear however that this requirement transfers to the multi-objective domain, especially for MO-GOMEA. This is because the elitist archive is shared among all populations and the Forced Improvement phase in MO-GOMEA causes substantial interaction with the elitist archive, and thus substantial interaction between populations that may increase the efficiency of smaller populations. In this section, we therefore explicitly wish to study the impact of stopping smaller populations in MO-GOMEA. It is more difficult, however, to derive a proper metric to measure and compare the qualities of different populations during a multi-objective optimization process. The average fitness of a population is not a suitable metric in the multi-objective optimization context because the fitness value of each candidate solution involves multiple conflicting objectives. The hypervolume performance indicator [29] is also not a suitable metric for the termination checking purpose here. A population P_{i-1} of size N_{i-1} can have a smaller hypervolume value than the population P_i of size N_i but P_{i-1} is not necessarily worse than P_i if they are approaching different regions of the Pareto-optimal front.

A termination condition was developed for IMS to be used with NSGA-II in [23], but experiments were very limited: only one test problem with 2 real-valued decision variables was used and the scalability issue was entirely omitted. We here employ the Pareto dominance relation to determine when a small population should be terminated due to inefficiency. A population P_i of size N_i is considered to be inefficient compared to a population P_j of size $N_j > N_i$ if the Pareto front formed by P_i is totally Pareto-dominated by the Pareto front formed by P_j or if all points on the Pareto front formed by P_i also exist on the Pareto front formed by P_j. We incorporate this condition into NSGA-II, MOEA/D, and MO-GOMEA, and perform the experiments again to observe its influence on the optimizers.

6.3. Experimental Results

6.3.1. Scalable Benchmark Problems

Figure 11 shows the average number of evaluations (over 100 independent runs) spent by NSGA-II, MOEA/D, and MO-GOMEA variants on scalable benchmark problems until the whole Pareto-optimal fronts are obtained. MO-GOMEA variants clearly outperform NSGA-II and MOEA/D variants on solving the Trap5-Inverse Trap5 problem. Trap functions can only be efficiently solved by optimizers that have linkage learning abilities, which NSGA-II and MOEA/D do not employ. For the Zeromax-Onemax problem, where all variables are independent and linkage learning is not necessary, MO-GOMEA variants still have a better scalability than other MOEAs (the differences are statistically significant with $p < 0.001$ for all problem sizes). However, NSGA-II and MOEA/D variants perform better than MO-GOMEA variants when solving the LOTZ problem due to the mutation operator as discussed in [16]. Section 5 above shows that if MO-GOMEA is coupled with a mutation operator, it can also solve LOTZ easily and does so more efficiently than NSGA-II. For these 3 scalable benchmark problems, the termination criterion of small populations shows no or little influence on MO-GOMEA while it improves the performance of NSGA-II and MOEA/D (the differences are statistically significant with $p < 0.05$).

It has been reported in [16] that NSGA-II with a population of size 4 can solve all Zeromax-Onemax and LOTZ problem instances more efficiently compared to MO-GOMEA. NSGA-II with the IMS, however,

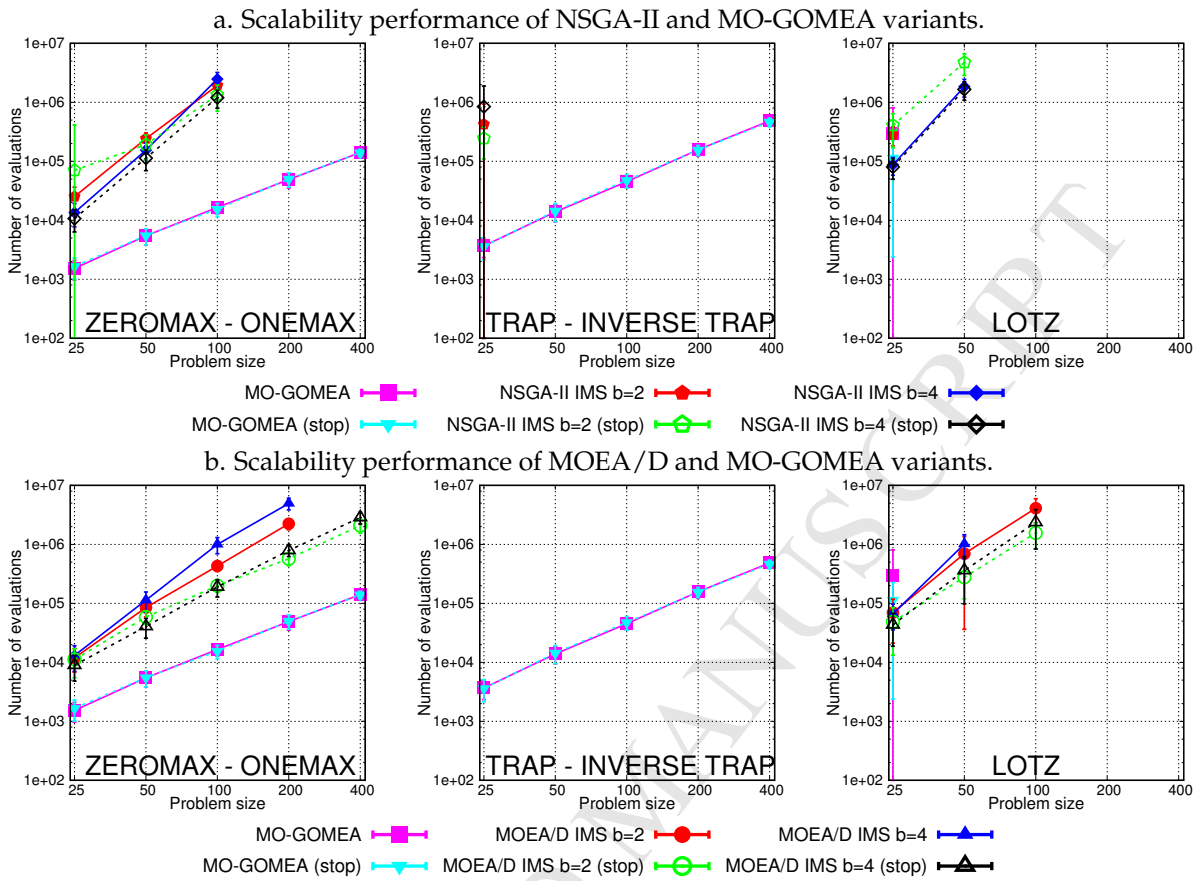


Figure 11: Scalability performance of NSGA-II, MOEA/D, and MO-GOMEA when terminating small populations on scalable benchmark problems. Horizontal axis: Problem size. Vertical axis: The number of evaluations until the whole Pareto-optimal front is obtained: mean & standard deviation. **(stop)** indicates that the termination criterion of small populations is in used.

cannot solve Zeromax-Onemax nor LOTZ as efficiently as the original NSGA-II. In fact, for these two problems, NSGA-II does not need to employ any solution recombination but only a mutation operator. NSGA-II simply needs to run many generations and wait for the right bits to be flipped at the right time to obtain a Pareto-optimal solution. However, it should be noted that, in practice, a population size of 4 hardly suffices for any population-based EA to solve any real-world problem. Therefore, putting MOEAs into the IMS would still be a more convenient option than trying many population sizes in a trial-and-error manner.

6.3.2. *MAXCUT Results*

Table 3 shows values of the 4 performance indicators which are evaluated based on the result approximation sets obtained at the end of 100 runs of each algorithm solving 4 MAXCUT problem instances. For all problem instances, the result approximation sets obtained by all algorithms have similar sizes (all FO values are similar to each other). For the largest case $l = 100$, MO-GOMEA variants obtain the best MS scores and the differences with other algorithms are statistically significant ($p < 0.001$) while, for the smaller cases cases $l = 12, 25,$ and 50, the differences are not statistically significant ($p > 0.05$). MO-GOMEA variants also perform significantly better than other MOEAs in terms of the GD indicator ($p < 0.001$ for the cases $l = 12, 25,$ and 50; for the case $l = 100$, GD scores of MOEA/D appear to be better but the differences with those of MO-GOMEA are not found to be significant with $p > 0.37$). Also, MO-GOMEA variants signifi-

Table 3: MAXCUT problems. Means and standard deviations (in brackets) of the 4 performance indicators evaluated on the result approximation sets of 100 runs of NSGA-II, MOEA/D, and MO-GOMEA variants. The best mean value of each indicator for each problem instance is presented in bold.

Instance	Algorithm	FO	MS	GD	IGD
12	NSGA-II IMS b=2	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	NSGA-II IMS b=2 (stop)	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	NSGA-II IMS b=4	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	NSGA-II IMS b=4 (stop)	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MOEA/D IMS b=2	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MOEA/D IMS b=2 (stop)	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MOEA/D IMS b=4	**6** (0.0)	1.414048 (0.000722)	0.001334 (0.005307)	0.001334 (0.005307)
	MOEA/D IMS b=4 (stop)	**6** (0.0)	1.414081 (0.000650)	0.000927 (0.004543)	0.000927 (0.004543)
	MO-GOMEA	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
	MO-GOMEA (stop)	**6** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
25	NSGA-II IMS b=2	**17** (0.3)	**1.414214** (0.000000)	0.000128 (0.000547)	0.000456 (0.001340)
	NSGA-II IMS b=2 (stop)	**17** (0.3)	**1.414214** (0.000000)	0.000195 (0.000662)	0.000466 (0.001229)
	NSGA-II IMS b=4	**17** (0.6)	**1.414214** (0.000000)	0.000917 (0.001947)	0.002325 (0.003800)
	NSGA-II IMS b=4 (stop)	**17** (0.5)	1.414172 (0.000415)	0.000400 (0.000997)	0.001128 (0.002490)
	MOEA/D IMS b=2	**17** (0.3)	**1.414214** (0.000000)	0.000204 (0.000789)	0.000575 (0.001308)
	MOEA/D IMS b=2 (stop)	**17** (0.4)	**1.414214** (0.000000)	0.000259 (0.000989)	0.000787 (0.002252)
	MOEA/D IMS b=4	**17** (0.6)	**1.414214** (0.000000)	0.000807 (0.001774)	0.002104 (0.003233)
	MOEA/D IMS b=4 (stop)	**17** (0.6)	**1.414214** (0.000000)	0.000557 (0.001368)	0.002043 (0.003199)
	MO-GOMEA	**17** (0.1)	**1.414214** (0.000000)	**0.000000** (0.000000)	0.000030 (0.000297)
	MO-GOMEA (stop)	**17** (0.0)	**1.414214** (0.000000)	**0.000000** (0.000000)	**0.000000** (0.000000)
50	NSGA-II IMS b=2	39 (2.7)	1.363267 (0.080120)	0.002409 (0.002323)	0.006813 (0.007236)
	NSGA-II IMS b=2 (stop)	39 (3.1)	1.356159 (0.089255)	0.002767 (0.002490)	0.007530 (0.008615)
	NSGA-II IMS b=4	38 (2.8)	1.362368 (0.088706)	0.002721 (0.002327)	0.007946 (0.008073)
	NSGA-II IMS b=4 (stop)	39 (3.2)	1.367207 (0.093441)	0.002543 (0.002567)	0.007167 (0.008717)
	MOEA/D IMS b=2	41 (1.8)	1.407598 (0.029967)	0.001589 (0.001914)	0.002490 (0.002811)
	MOEA/D IMS b=2 (stop)	41 (2.1)	1.401556 (0.044288)	0.001600 (0.001947)	0.002830 (0.004438)
	MOEA/D IMS b=4	39 (2.6)	1.382927 (0.061802)	0.002676 (0.002623)	0.005866 (0.005758)
	MOEA/D IMS b=4 (stop)	40 (2.7)	1.387771 (0.060406)	0.002866 (0.002886)	0.005373 (0.005745)
	MO-GOMEA	**42** (0.8)	**1.414214** (0.000000)	0.000764 (0.000749)	0.001001 (0.000535)
	MO-GOMEA (stop)	41 (0.7)	**1.414214** (0.000000)	**0.000322** (0.000376)	**0.000570** (0.000453)
100	NSGA-II IMS b=2	103 (7.1)	1.204932 (0.073950)	0.005985 (0.003470)	0.012660 (0.004213)
	NSGA-II IMS b=2 (stop)	108 (7.5)	1.221898 (0.063980)	0.004804 (0.003420)	0.010709 (0.004182)
	NSGA-II IMS b=4	102 (9.0)	1.224059 (0.081618)	0.009761 (0.006458)	0.015229 (0.007066)
	NSGA-II IMS b=4 (stop)	105 (8.0)	1.245506 (0.077076)	0.008136 (0.005708)	0.012690 (0.006081)
	MOEA/D IMS b=2	116 (7.2)	1.261731 (0.060062)	**0.003089** (0.002198)	0.007083 (0.002967)
	MOEA/D IMS b=2 (stop)	116 (5.7)	1.267189 (0.058216)	0.003154 (0.001912)	0.007002 (0.002425)
	MOEA/D IMS b=4	110 (6.7)	1.248822 (0.071784)	0.004698 (0.003229)	0.009660 (0.003890)
	MOEA/D IMS b=4 (stop)	112 (6.3)	1.242933 (0.075860)	0.005610 (0.004995)	0.010572 (0.005630)
	MO-GOMEA	**122** (3.8)	1.394845 (0.051454)	0.003229 (0.002474)	**0.004044** (0.002234)
	MO-GOMEA (stop)	119 (4.1)	**1.409658** (0.047542)	0.003275 (0.001792)	0.004069 (0.001775)

cantly outperform other algorithms in terms of IGD values ($p < 0.001$). The termination criterion of small populations have little or no influence on the performance of MO-GOMEA in solving these MAXCUT problem instances (e.g., for the case $l = 100$, the difference in IGD values is insignificant with $p = 0.22$).

Figure 12 compares the average IGD convergence performance of MO-GOMEA variants with NSGA-II and MOEA/D when solving MAXCUT. For the smallest instance $l = 12$, all algorithms perform similarly, but as the problem size increases, MO-GOMEA variants clearly outperforms other algorithms (the performance gaps are statistically significant with $p < 0.001$). As the MAXCUT problem size becomes larger, it is more important for the solution variation in MOEAs to respect and exploit linkage information between problem variables in order to efficiently solve the problem. The capability of learning and exploiting linkage structures gives MO-GOMEA an advantage over other MOEAs with traditional variation operators (e.g., crossover and mutation). It can be observed in Figure 12 that the termination criterion of inefficient populations has no significant influence on the performance of all algorithms ($p > 0.22$), except for the case $l = 100$, where the performance of NSGA-II variants is statistically significantly improved ($p < 0.005$). For NSGA-II and MOEA/D, the IMS with base $b = 2$ also obtains significantly better results than the IMS with

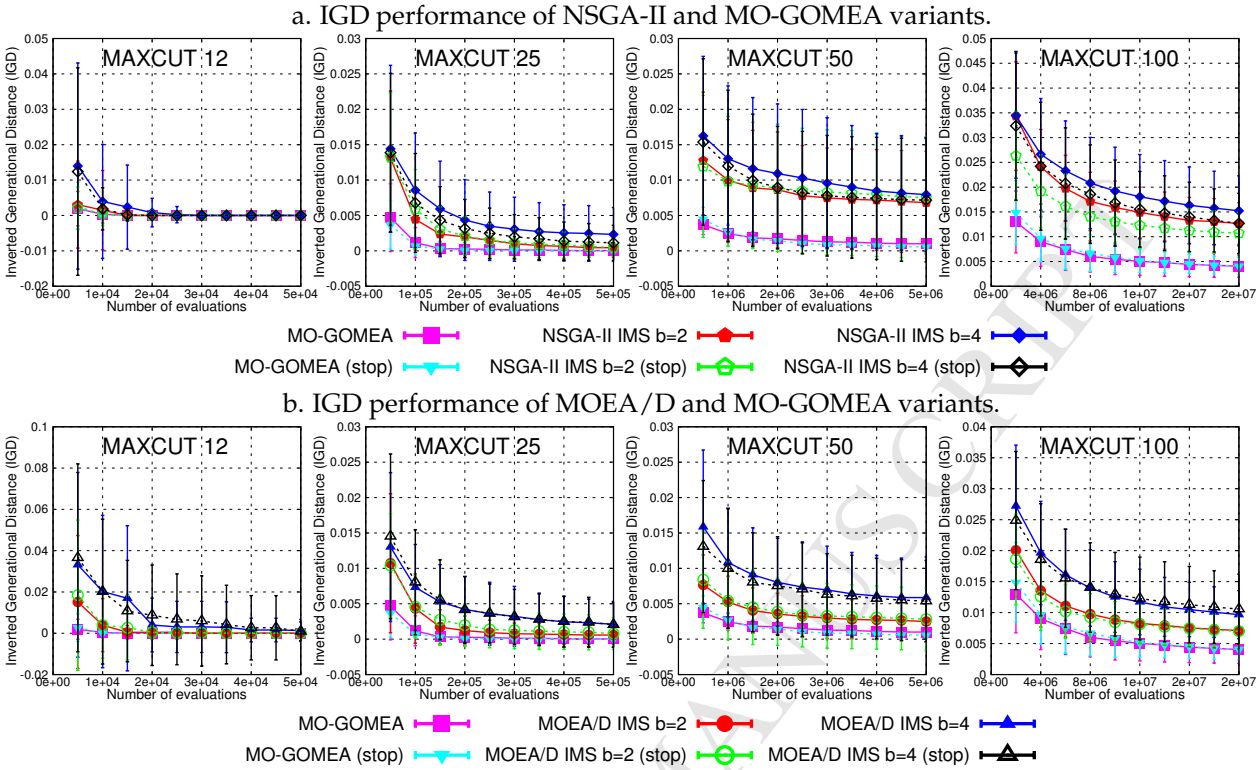


Figure 12: Average IGD convergence performance of NSGA-II, MOEA/D, and MO-GOMEA when terminating small populations on MAXCUT. Horizontal axis: number of evaluations (both objectives per evaluation). Vertical axis: IGD values: mean & standard deviation. **(stop)** indicates that the termination criterion of small populations is in used.

base $b = 4$ ($p < 0.05$).

6.3.3. *Knapsack Results*

Table 4 shows values of the 4 performance indicators which are evaluated based on the result approximation sets obtained at the end of 100 runs of each algorithm solving 4 knapsack problem instances. On the one hand, for most cases, MOEA/D variants obtain the best FO values, i.e., the numbers of solutions in approximation sets obtained by MOEA/D are significantly more than those of other algorithms ($p < 0.001$). On the other hand, for all cases, MO-GOMEA variants achieve the best MS values, i.e., the range of the solutions in approximation sets obtained by MO-GOMEA are significantly larger than those of other algorithms ($p < 0.001$). For all cases, both NSGA-II and MOEA/D score significantly better than MO-GOMEA in terms of the proximity metric GD indicator ($p < 0.001$), i.e., the solutions obtained by NSGA-II and MOEA/D are closer to the reference Pareto-optimal fronts than those obtained by MO-GOMEA. However, MO-GOMEA variants significantly outperform NSGA-II and MOEA/D variants in terms of the IGD indicator, which takes both proximity and diversity of approximation sets in account ($p < 0.001$). Based on the results in Table 4, it can be inferred that the approximation sets obtained by MO-GOMEA are more well-spread along the reference Pareto-optimal front than those found by NSGA-II and MOEA/D. These results suggest that the cluster-based operation and the exploitation of linkage structures (as in MO-GOMEA) are important for approaching all parts of the Pareto-optimal fronts, especially the solutions in the extreme regions of the fronts. While Tchebycheff decomposition-based operation in MOEA/D does address the diversity issue (by employing multiple uniformly-distributed weight vectors), the results here indicate that traditional variation operators (i.e., crossover and mutation) are not sufficient to ensure a good combination of both proximity and diversity.

Table 4: Knapsack problems. Means and standard deviations (in brackets) of the 4 performance indicators evaluated on the result approximation sets of 100 runs of NSGA-II, MOEA/D, and MO-GOMEA variants. The best mean value of each indicator for each problem instance is presented in bold.

Instance	Algorithm	FO	MS	GD	IGD
100	NSGA-II IMS b=2	113 (2.0)	1.297308 (0.041245)	0.000179 (0.000245)	0.003102 (0.000949)
	NSGA-II IMS b=2 (stop)	**114** (2.0)	1.297704 (0.035250)	**0.000118** (0.000103)	0.002915 (0.001040)
	NSGA-II IMS b=4	110 (2.3)	1.281999 (0.037681)	0.000256 (0.000230)	0.003817 (0.001396)
	NSGA-II IMS b=4 (stop)	113 (3.2)	1.291592 (0.042446)	0.000185 (0.000216)	0.003258 (0.001450)
	MOEA/D IMS b=2	110 (3.1)	1.341979 (0.047886)	0.000431 (0.000331)	**0.002441** (0.000955)
	MOEA/D IMS b=2 (stop)	109 (3.3)	1.333943 (0.051325)	0.000454 (0.000402)	0.002489 (0.000967)
	MOEA/D IMS b=4	111 (2.5)	1.282477 (0.041061)	0.000207 (0.000158)	0.003695 (0.001620)
	MOEA/D IMS b=4 (stop)	112 (2.5)	1.289111 (0.042569)	0.000249 (0.000200)	0.003525 (0.001581)
	MO-GOMEA	105 (3.7)	1.408707 (0.020036)	0.001456 (0.000408)	0.002504 (0.000598)
	MO-GOMEA (stop)	96 (5.2)	**1.412407** (0.008297)	0.002888 (0.000782)	0.004362 (0.000843)
250	NSGA-II IMS b=2	258 (13.0)	0.989515 (0.041676)	0.002928 (0.000513)	0.016098 (0.003058)
	NSGA-II IMS b=2 (stop)	273 (14.3)	1.007240 (0.039103)	0.002373 (0.000383)	0.014458 (0.002711)
	NSGA-II IMS b=4	216 (12.0)	0.864663 (0.043933)	0.004096 (0.000700)	0.028521 (0.004987)
	NSGA-II IMS b=4 (stop)	260 (22.7)	0.925066 (0.052333)	0.002506 (0.000673)	0.020705 (0.005440)
	MOEA/D IMS b=2	267 (15.5)	1.026295 (0.061838)	0.003071 (0.000789)	0.015363 (0.004512)
	MOEA/D IMS b=2 (stop)	269 (21.7)	1.032427 (0.049303)	0.003077 (0.001402)	0.014725 (0.004056)
	MOEA/D IMS b=4	271 (18.7)	0.892351 (0.060624)	0.002220 (0.000489)	0.025063 (0.005205)
	MOEA/D IMS b=4 (stop)	**283** (20.4)	0.915091 (0.059387)	**0.002131** (0.000716)	0.021963 (0.005450)
	MO-GOMEA	259 (14.6)	1.269258 (0.035736)	0.004828 (0.000646)	**0.005905** (0.000656)
	MO-GOMEA (stop)	240 (17.7)	**1.301111** (0.033942)	0.006939 (0.001414)	0.007485 (0.001239)
500	NSGA-II IMS b=2	439 (18.0)	0.911970 (0.025552)	0.007142 (0.000332)	0.028119 (0.002545)
	NSGA-II IMS b=2 (stop)	488 (21.4)	0.946015 (0.026307)	0.006806 (0.000345)	0.024329 (0.002385)
	NSGA-II IMS b=4	302 (16.4)	0.717528 (0.032352)	0.008264 (0.000864)	0.057076 (0.005141)
	NSGA-II IMS b=4 (stop)	398 (33.6)	0.791392 (0.039698)	0.006617 (0.000894)	0.043484 (0.006181)
	MOEA/D IMS b=2	473 (32.0)	0.949042 (0.050224)	0.007114 (0.000637)	0.027744 (0.004071)
	MOEA/D IMS b=2 (stop)	475 (67.3)	0.941330 (0.036743)	0.007948 (0.002447)	0.027839 (0.003509)
	MOEA/D IMS b=4	451 (22.1)	0.796041 (0.070996)	0.005618 (0.000612)	0.045179 (0.006981)
	MOEA/D IMS b=4 (stop)	**491** (45.3)	0.817352 (0.073071)	**0.005458** (0.000639)	0.041626 (0.007741)
	MO-GOMEA	369 (19.6)	1.214116 (0.029429)	0.008235 (0.000864)	**0.009419** (0.000913)
	MO-GOMEA (stop)	344 (21.9)	**1.221255** (0.027712)	0.010589 (0.001290)	0.011449 (0.001268)
750	NSGA-II IMS b=2	592 (23.1)	0.880609 (0.021631)	0.003902 (0.000301)	0.027685 (0.002219)
	NSGA-II IMS b=2 (stop)	687 (29.6)	0.921931 (0.020341)	0.003249 (0.000331)	0.023278 (0.001896)
	NSGA-II IMS b=4	381 (25.8)	0.645991 (0.035627)	0.006180 (0.000752)	0.059870 (0.005553)
	NSGA-II IMS b=4 (stop)	528 (61.9)	0.726519 (0.040664)	0.003827 (0.000770)	0.046552 (0.006034)
	MOEA/D IMS b=2	722 (50.7)	0.974968 (0.039470)	0.003228 (0.001565)	0.022816 (0.004078)
	MOEA/D IMS b=2 (stop)	**759** (98.3)	0.975061 (0.040800)	0.004074 (0.001704)	0.021901 (0.003829)
	MOEA/D IMS b=4	645 (42.2)	0.778944 (0.095123)	0.002934 (0.000520)	0.044435 (0.006764)
	MOEA/D IMS b=4 (stop)	712 (63.2)	0.828461 (0.083994)	**0.002529** (0.000519)	0.037823 (0.007605)
	MO-GOMEA	523 (24.0)	**1.309775** (0.020059)	0.007297 (0.000632)	**0.008014** (0.000697)
	MO-GOMEA (stop)	519 (27.2)	1.308935 (0.020659)	0.007927 (0.000848)	0.008546 (0.000894)

Figure 13 shows the IGD convergence performance of NSGA-II, MOEA/D, and MO-GOMEA with and without terminating inefficient populations on 4 knapsack problems over time (in terms of the number of evaluations). For the cases $l = 250, 500$, and 750, the termination criterion improves the performance of NSGA-II, where the differences are found to be statistically significant ($p < 0.001$). This confirms the fact that when small populations are inefficient for NSGA-II, they should be terminated as soon as possible so that fitness evaluations would not be wasted on them. For MOEA/D with the IMS $b = 4$, the termination criterion does improve the IGD convergence ($p < 0.001$ for the cases $l = 250, 500, 750$) while, for MOEA/D with the IMS $b = 2$, the termination criterion shows no significant influence on the performance of MOEA/D ($p > 0.05$ for all cases). Table 4 and Figure 13 show again that base 2 is also a better IMS setting for NSGA-II and MOEA/D than base 4. It has been suggested in [33] that smaller base values are more suitable for the employed EA if it suffers the effects of genetic drift. Diversity preservation is an important task in multi-objective optimization, and the IMS $b = 2$ introduces larger population sizes with more diverse candidate solutions at a faster rate than the IMS $b = 4$ variant. NSGA-II with the IMS $b = 2$ coupled with the termination criterion of small populations can get rid of small and inefficient populations

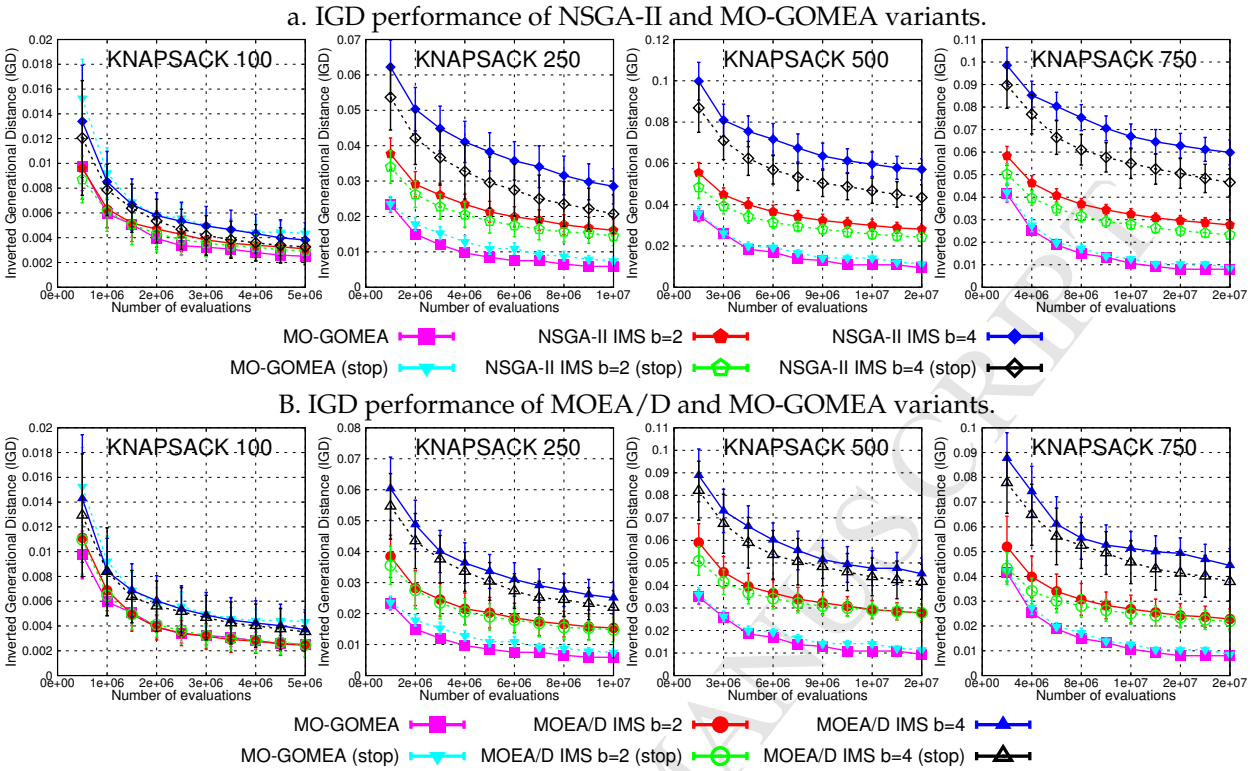


Figure 13: Average IGD convergence performance of NSGA-II, MOEA/D, and MO-GOMEA when terminating small populations on knapsack. Horizontal axis: number of evaluations (both objectives per evaluation). Vertical axis: IGD values: mean & standard deviation. **(stop)** indicates that the termination criterion of small populations is in used.

and move to sufficiently larger populations more quickly.

The termination criterion, however, shows little influence on the performance of MO-GOMEA in solving the problem instances considered here. For the knapsack problem instances, MO-GOMEA without the termination criterion actually performs slightly better than the variant with the termination criterion, where the differences are found to be statistically significant ($p < 0.001$ for all cases). This phenomenon suggests that MO-GOMEA can actually operate effectively with small populations and that indeed the sharing and active use of the elitist archive in all populations make terminating smaller populations unnecessary with respect to the problem instances concerned in this work. What we observe here conforms with previous research on the scalability of GOMEA in single-objective optimization, in which GOMEA generally has minimally-required population sizes that are much smaller than other population-based EAs [9, 28]. Nevertheless, for problem instances that require MO-GOMEA to have population sizes much larger than the sizes of the initial populations, the termination criterion could be used to terminate inefficient populations as in the case of NSGA-II. The results obtained in this section suggest that the termination criterion of inefficient populations, that we develop in this work, can be employed by the IMS when coupled with MOEAs to remove the requirements of parameter settings.

7. Discussion

On almost all benchmark problems, MO-GOMEA is found to be a promising MOEA with excellent scalability. Problem instances of large sizes normally have wide Pareto-optimal fronts with multiple regions. Population clustering ensures MO-GOMEA allocates an equal amount of search effort to every region and the whole Pareto-optimal front can thus be evenly approached. Especially the cluster-based

operating mechanism of MO-GOMEA is convenient for dedicated adaptations if different regions of the Pareto-optimal front have different characteristics and thus require different strategies to exploit problem structure effectively and efficiently. In the multi-objective knapsack benchmark (Figure 13), by clustering the working population, it is straightforward to assign the multi-objective repair mechanism to the middle-region clusters and the suitable single-objective repair mechanism to the corresponding extreme-region cluster. Clustering helps MO-GOMEA score on the diversity part of the IGD performance indicator.

As each cluster of MO-GOMEA approaches a specific region of the Pareto-optimal front, linkage learning captures problem-variable dependencies that are relevant to that region. Following the structure of the LT dedicatedly learned from a cluster, the GOM operator creates new candidate solutions by juxtaposing currently existing building blocks in a way that is specifically suitable to that cluster. The genetic local search nature of GOM also ensures that an offspring is better or at least as good as its parent solution. Linkage learning and GOM together ensure that the building blocks relevant to each cluster are synthesized, detected, and propagated to ensure effective convergence toward the Pareto-optimal front, helping MO-GOMEA score on the proximity part of the IGD performance indicator.

The combined effect of clustering the population and exploiting linkage information results in the better performance for MO-GOMEA over NSGA-II and MOEA/D. The widening performance gap between MO-GOMEA and NSGA-II or between MO-GOMEA and MOEA/D as the problem size increases furthermore confirms that linkage learning and GOM lead to excellent scalability (see Figures 11, 12 and 13) if the problem instance at hand exhibit certain linkage structures which can be exploit. MO-GOMEA with the IMS is also a practical MOEA in the sense that it does not require practitioners to perform any parameter tuning themselves.

In this article, we show how the IMS are employed to eliminate the requirements of setting the population size parameter (for NSGA-II, MOEA/D, and MO-GOMEA) and the number-of-clusters parameter (for MO-GOMEA). Note that while the size of the initial population P_0 still needs to be set, the performance of MOEAs with the IMS is robust against the setting of this parameter as well as other control parameter variations in general. That is, practitioners can start the algorithms with some small values for the control parameters, and given enough running time, the proper parameter values (e.g., the population size or the number of clusters) will be reached by the IMS, yielding better and better solutions over time. The IMS does incur certain computational overhead compared to using the optimal population sizes for the problem instances at hand, which are, however, impossible to known beforehand in real-world optimization. Compared to using IMS, the setting of control parameters of MOEAs with some fixed values might yield poor results if the chosen parameter values are not suitable for the problem instance at hand, and new optimization runs must then be performed again with different parameter settings in a trial-and-error manner. Therefore, IMS can be used to design anytime algorithms for real-world optimization, i.e., practitioners only need to start the algorithms without worrying about determining proper parameter settings beforehand and the algorithms can be kept running until acceptable solutions are obtained.

8. Conclusions

We have given an overview of the Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm (MO-GOMEA). We described the key features that make MO-GOMEA a scalable solver, namely elitist archiving, population clustering, linkage learning and gene-pool optimal mixing. We then made MO-GOMEA a practical solver by placing MO-GOMEA in the Interleaved Multi-start Scheme (IMS) that eliminates the required setting of the population size parameter, which is notoriously difficult for any population-based EA, and the number-of-clusters parameter. The resulting MO-GOMEA with the IMS was shown to retain the scalability of the original MO-GOMEA and to have excellent performance on different benchmark problems. The scalability and practicality of MO-GOMEA suggest that MO-GOMEA is a promising solver for tackling complicated (real-world) multi-objective combinatorial optimization problems.

Acknowledgments

The work in this article was within the COCAPLEN project (number 647.000.007) funded by The Netherlands Organisation for Scientific Research (NWO).

References

[1] N. H. Luong, H. La Poutré, P. A. N. Bosman, Exploiting Linkage Information and Problem-Specific Knowledge in Evolutionary Distribution Network Expansion Planning, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15, ACM Press, 2015, pp. 1231–1238. doi:10.1145/2739480.2754682.

[2] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation 1 (1) (2011) 32–49. doi:10.1016/j.swevo.2011.03.001.

[3] P. A. N. Bosman, D. Thierens, The balance between proximity and diversity in multiobjective evolutionary algorithms, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 174–188. doi:10.1109/TEVC.2003.810761.

[4] C. A. Coello Coello, G. Pulido, E. Montes, Current and Future Research Trends in Evolutionary Multiobjective Optimization, in: Information Processing with Evolutionary Algorithms, Advanced Information and Knowledge Processing, Springer London, 2005, pp. 213–231. doi:10.1007/1-84628-117-2_15.

[5] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, C. A. C. Coello, A Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part I, IEEE Transactions on Evolutionary Computation 18 (1) (2014) 4–19. doi:10.1109/TEVC.2013.2290086.

[6] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[7] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization, in: Evolutionary Methods for Design, Optimisation, and Control, CIMNE, Barcelona, Spain, 2002, pp. 95–100.

[8] Qingfu Zhang, Hui Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731. doi:10.1109/TEVC.2007.892759.

[9] D. Thierens, P. A. N. Bosman, Optimal mixing evolutionary algorithms, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11, ACM Press, 2011, pp. 617–624. doi:10.1145/2001576.2001661.

[10] N. H. Luong, M. O. W. Grond, H. La Poutré, P. A. N. Bosman, Scalable and Practical Multi-Objective Distribution Network Expansion Planning, in: 2015 IEEE Power & Energy Society General Meeting Proceedings, 2015, pp. 1–5.

[11] M. Pelikan, K. Sastry, D. E. Goldberg, Multiobjective hBOA, clustering, and scalability, in: Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05, ACM Press, 2005, pp. 663–670. doi:10.1145/1068009.1068122.

[12] M. Pelikan, M. W. Hauschild, F. G. Lobo, Estimation of Distribution Algorithms, in: Springer Handbook of Computational Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 899–928. doi:10.1007/978-3-662-43505-2{_}45.

[13] P. A. N. Bosman, The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10, ACM Press, 2010, pp. 351–358. doi:10.1145/1830483.1830549.

[14] M. Laumanns, J. Ocenasek, Bayesian Optimization Algorithms for Multi-objective Optimization, in: Parallel Problem Solving from Nature - PPSN VII, Vol. 2439 of Lecture Notes in Computer Science, Springer, 2002, pp. 298–307.

[15] I. Gronau, S. Moran, Optimal implementations of UPGMA and other common clustering algorithms, Information Processing Letters 104 (6) (2007) 205–210. doi:10.1016/j.ipl.2007.07.002.

[16] N. H. Luong, H. La Poutré, P. A. N. Bosman, Multi-objective gene-pool optimal mixing evolutionary algorithms, in: Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14, ACM Press, 2014, pp. 357–364. doi:10.1145/2576768.2598261.

[17] M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining convergence and diversity in evolutionary multiobjective optimization., Evolutionary Computation 10 (3) (2002) 263–82. doi:10.1162/106365602760234108.

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

[19] O. Giel, P. K. Lehre, On the Effect of Populations in Evolutionary Multi-Objective Optimisation, Evolutionary Computation 18 (3) (2010) 335–356. doi:10.1162/EVCO{_}a{_}00013.

[20] N. H. Luong, M. O. W. Grond, P. A. N. Bosman, H. La Poutré, Medium-voltage distribution network expansion planning with gene-pool optimal mixing evolutionary algorithms, in: Artificial Evolution - 11th International Conference, Evolution Artificielle, EA 2013, Bordeaux, France, October 21-23, 2013. Revised Selected Papers, 2013, pp. 93–105. doi:10.1007/978-3-319-11683-9_8.

[21] G. R. Harik, F. G. Lobo, A Parameter-Less Genetic Algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO '99), Morgan Kaufmann, 1999, pp. 258–265.

[22] M. Pelikan, A. Hartmann, T.-K. Lin, Parameter-less hierarchical Bayesian optimization algorithm, in: Parameter Setting in Evolutionary Algorithms. Studies in Computational Intelligence., Vol. 54, Springer Berlin Heidelberg, 2007, pp. 225–239.

[23] K. D. Tran, Elitist non-dominated sorting GA-II (NSGA-II) as a parameter-less multi-objective genetic algorithm, in: Proceedings. IEEE SoutheastCon, 2005., 2005, pp. 359–367. doi:10.1109/SECON.2005.1423273.

[24] A. Eiben, S. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm and Evolutionary Computation 1 (1) (2011) 19 – 31. doi:http://dx.doi.org/10.1016/j.swevo.2011.02.001.

[25] G. Karafotias, M. Hoogendoorn, Á. E. Eiben, Parameter control in evolutionary algorithms: Trends and challenges, IEEE Trans. Evolutionary Computation 19 (2) (2015) 167–187. doi:10.1109/TEVC.2014.2308294.

[26] J. Knowles, D. Corne, Properties of an adaptive archiving algorithm for storing nondominated vectors, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 100–116. doi:10.1109/TEVC.2003.810755.

[27] N. H. Luong, P. A. N. Bosman, Elitist Archiving for Multi-Objective Evolutionary Algorithms: To Adapt or Not to Adapt, in: C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, M. Pavone (Eds.), Parallel Problem Solving from Nature - PPSN XII, Vol. 7492 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 72–81. doi:10.1007/978-3-642-32964-7.

[28] P. A. N. Bosman, D. Thierens, More concise and robust linkage learning by filtering and combining linkage hierarchies, in: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13, ACM Press, 2013, pp. 359–366. doi:10.1145/2463372.2463420.

[29] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271. doi:10.1109/4235.797969.

[30] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., 2001.

[31] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings, Part II, Springer International Publishing, 2015, pp. 110–125. doi:10.1007/978-3-319-15892-1_8.

[32] J. C. Pereira, F. G. Lobo, A Java Implementation of Parameter-less Evolutionary Algorithms., CoRR abs/1506.0.

[33] M. Pelikan, F. G. Lobo, Parameter-less Genetic Algorithm: A Worst-case Time and Space Complexity Analysis. IlliGAL Report No. 99014., Tech. rep., University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL. (1999).