Mutation Operators based on Variable Grouping for Multi-objective Large-scale Optimization

Heiner Zille*, Hisao Ishibuchi†, Sanaz Mostaghim* and Yusuke Nojima†

*Institute for Intelligent Cooperating Systems
Faculty of Computer Science, Otto von Guericke University Magdeburg, Germany
Email: {heiner.zille, sanaz.mostaghim}@ovgu.de

†Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University, Sakai, Osaka 599-8531, Japan
Email: {hisaoi, nojima}@cs.osakafu-u.ac.jp

Abstract—**In this paper, we study the influence of using variable grouping inside mutation operators for large-scale multi-objective optimization. We introduce three new mutation operators based on the well-known Polynomial Mutation. The variable grouping in these operators is performed using two different grouping mechanisms, including Differential Grouping from the literature. In our experiments, two popular algorithms (SMPSO and NSGA-II) are used with the proposed operators on the WFG1-9 test problems. We examine the performance of the proposed mutation operators and take a look at the impact of the different grouping mechanisms on the performance. Using the Hypervolume and IGD indicators, we show that mutation using variable grouping can significantly improve the results on all tested problems in terms of both convergence and diversity of solutions. Furthermore, the performance of SMPSO and NSGA-II with the proposed operators is compared with a large-scale multi-objective algorithm (CCGDE3). The results show that the proposed operators can significantly outperform CCGDE3.**

I. Introduction

In evolutionary computation, the creation of new solutions in the search space is usually achieved by genetic operators, such as crossover and mutation. Mutation operators are designed to replicate the mutation processes in nature, usually by applying small changes to a given individual. Operators like the Polynomial Mutation [1] are widely used in the literature and work well on small-scale multi-objective optimization problems. However, when the dimensionality of the search space is increased, the performance of classical evolutionary computation approaches often deteriorates. Finding optimal solutions in problems with a large number of variables is an ongoing challenge both in single- and multi-objective optimization. Concepts like Cooperative Coevolution [2] and other approaches [3] have been proposed in the literature, which usually involve a special mechanism for the division of the variables into multiple *groups*.

This work examines the effects of variable grouping when used inside mutation operators. By using grouping mechanisms similar to those used in problem decomposition methods, we aim to make a more intelligent choice on which variables are changed by the mutation step during optimization. In addition to the well-known Polynomial Mutation [1], we propose three new mutation operators in this work. First, we extend the Polynomial Mutation to the *Grouped Polynomial Mutation* by making the selection on which variables are mutated based on a predefined grouping scheme. Second, we propose another version called the *Linked Polynomial Mutation*, which binds together the amount of change of the mutated variables per individual. The third operator proposed in this work utilizes both of these two concepts. The proposed operators are tested using two different grouping methods: One is a simple grouping strategy named *Ordered Grouping* that is based on absolute variable values, and the other is *Differential Grouping* [4], a single-objective method developed to detect variable interaction. All of the operators are used within two well-known algorithms (NSGA-II [5] and SMPSO [6]) and furthermore compared with a large-scale optimizer called CCGDE3 [7] on the WFG1-9 problems [8]. The goal of this work is to show that using grouping mechanisms in mutation operators can significantly improve the performance of existing algorithms in large-scale multi-objective optimization, and therefore be beneficial for the design of future large-scale algorithms.

This article is structured as follows. In Section II, we briefly explain the concept of multi-objective optimization, followed by a review of related studies on multi-objective large-scale approaches, grouping mechanisms and mutation operators. In Section III, we explain how grouping methods can be utilized in mutation, and propose three new mutation operators using this concept. Section IV briefly explains the two different grouping mechanisms that are used in this work. After a performance evaluation of the proposed mutation operators in Section V, we conclude the paper in Section VI.

II. Background and Related Work

Problems in nature and science often contain multiple conflicting objectives. These problems are called multi-objective problems (MOPs) and can mathematically be formulated as:

$$Z: \quad min \quad \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x}))^T \\ s.t. \quad \vec{x} \in \Omega \subseteq \mathbb{R}^n \tag{1}$$

This kind of MOP maps the decision Space $\Omega = \{\vec{x} \in \mathbb{R}^n | \vec{g}(\vec{x}) \leq 0\}$ of dimension n to the objective space of dimension m. For such problems, a single optimal solution can no longer be determined, and modern problem solving methods often concentrate on finding an approximation of a Pareto-optimal solution set. A variety of metaheuristic algorithms have been developed to find a non-dominated solution set for approximating the true Pareto-front.

Large-scale optimization usually deals with optimizing MOPs that contain a large number of variables (also called many-variable problems). The scope of this work is to enable optimization methods to optimize problems that contain a large number of decision variables, as these kinds of large-scale problems are often difficult to solve for traditional optimization techniques.

Even though a variety of large-scale optimizers have been developed in recent years, most of them concentrate on single-objective optimization. An overview of existing large-scale global optimizers for single-objective problems can be found in [9]. One of the most popular concepts is *Cooperative Coevolution* (CC), which was first introduced by Potter and De Jong [2]. CC aims to optimize several independent populations of subsets of the n decision variables. New solution candidates have to be formed by combining the variable values from different subcomponents. However, genetic operators are only applied within each subcomponent. The concept of CC has since been used in a variety of large-scale single-objective algorithms [10]–[14].

For multi-objective optimization, the CCGDE3 algorithm was proposed [7], combining CC with the GDE3 optimizer [15]. The CCGDE3 algorithm was tested for the ZDT1-3 and ZDT6 [16] problems with up to 5000 decision variables, and performed well especially for the high-dimensional instances. An earlier approach by Iorio and Li [17] combined the concept of CC with the NSGA-II algorithm, and focused on the ZDT problems for their analysis. However, only small-dimensional instances of 10 and 30 variables were tested.

In [3], variable grouping was used in the so-called *Weighted Optimization Framework* to tackle large-scale multi-objective problems without using CC. A set of weights was applied to the groups of decision variables. The original decision variables and the weight-variables were optimized in turns using an arbitrary metaheuristic. The authors reported a superior performance compared to the SMPSO and NSGA-II algorithms for the 1000-variable WFG1-9 test problems.

In [18], an optimization problem based on a real world application was introduced within a competition at the IEEE Congress on Evolutionary Computation (CEC) 2015. This problem involved up to 4864 decision variables, and its multi-objective version has been used in [19] and [20]. However, the first approach did not treat the problem in a black-box manner and used derivative information from the objective functions. Both approaches performed superior to the baseline provided by the authors of [18], but were not tested any further on established benchmark problems.

Concepts like CC usually require a scheme for variable grouping, to divide the n decision variables into different subcomponents. A simple random grouping mechanism was for instance used in [10]. Consideration for non-separable problems was done in [11], where the interaction of variables is taken into account by a learning mechanism for finding the optimal division of variables. The *Differential Grouping* that is used in the present work was developed in [4] to find improved divisions of the variables in single-objective CC algorithms. Other concepts and extensions to this approach for variable grouping have also been proposed [21], [22]. Section IV later in this work will go into further detail on *Differential Grouping*.

The basic mutation operator used in this work is the Polynomial Mutation. It was proposed in [1] and is designed to alter a variable value based on a polynomial distribution with the original value in its center. Polynomial Mutation has been used in a variety of optimization algorithms, e.g. [5], [6], [17]. In the literature there are two common versions of Polynomial Mutation that differ from each other in the distribution of the possible changes. These are sometimes referred to as *highly disruptive* and *non-highly disruptive* Polynomial Mutation. The original non-highly disruptive mutation has the disadvantage to become useless when the variable value gets closer to its domain border [23]. A new highly disruptive version was therefore introduced in [24], where the complete domain of each variable was included in the distribution of the operator. In [23], a hybrid of both versions was introduced. A detailed description of the operator is given in Section III-A later in this work.

In [25], Polynomial Mutation was used in a comparison study of five different mutation schemes. Instead of making the selection of which variables to mutate independently for each variable based on a probability, certain rules were used to decide which variables would be changed at a given time and in which order. Their results on some single-objective problems showed an improved performance when the new mutation schemes were used.

III. Variable Grouping in Mutation Operators

This section describes four different mutation operators based on the widely used Polynomial Mutation. After describing the original version, we propose three new versions called *Linked Polynomial*, *Grouped Polynomial* and *Grouped Linked Polynomial* mutation. We explain how the mutation can be equipped with different variable grouping mechanisms, and how the amount of change for each variable can be linked within these groups.

A. Polynomial Mutation

Polynomial Mutation is designed to alter a variable's value based on a certain distribution around the current value. As mentioned before, there are two versions in the literature, and the highly disruptive version is used in this work. The pseudocode of this operator is shown in Figure 1.

Polynomial Mutation has two parameters: the mutation probability p and the distribution index η. The probability p is used to determine whether a variable is mutated or not. This is decided for each variable x_i $(i = 1, ..., n)$ of a solution $\vec{x}$ independently. Thus, the expected number of mutated variables in a solution is $n \cdot p$. Often a value of $p = \frac{1}{n}$ is used in the literature [5], [6], [17], which results in the mutation of only a single variable on average in each solution. The distribution index η of the Polynomial Mutation is used to determine the distribution from which the new value is chosen. The larger η becomes, the higher the probability that a value very close to x_i is drawn from the distribution. A typical value for η according to the literature is 20.0 [5], [17].

After determining if a mutation occurs to a variable (Line 3 in Figure 1), another random value u is drawn from a uniform distribution. This value is crucial, as it determines the amount of change that will be made to x_i. In Polynomial Mutation, a distinction is made between the left side and the right side of the distribution centered at x_i, depending on the value of u. The updated value for x_i is chosen according to the equations in Lines 5-12 in Figure 1. If a value should exceed one of the domain borders, it will be set to the respective border value by the repair mechanism in Line 13.

B. Linked Polynomial Mutation

In this subsection we introduce the *Linked Polynomial Mutation* operator. It follows a similar idea as the Polynomial Mutation, but keeps the amount of change connected among all mutated variables. In problem decomposition strategies, it can be desirable to consider the separability of a problem. In non-separable problems, some or all of the decision variables interact with each other, and a change in one of them without or with the wrong change in the other variable might reduce solution quality. We therefore design the linked version of the Polynomial Mutation operator, that establishes a link between the variables that are subject to change.

This is done by keeping the relative amount of change in each mutated variable fixed and equal per solution. Given the pseudocode in Figure 1, we simply move the choice of u (the amount of change) in Line 4 further up, so that only one fixed value for u is drawn prior to Line 1. The value is drawn only one time and used for computing all changes of each mutated variable of a given solution. By doing so, all variables that actually undergo mutation according to the probability p are mutated by the same amount (relative to their domain).

The motivation behind this operator is the following: If by coincidence mutation is performed on variables that interact with each other, this mechanism will keep the amount of change the same, and this might lead to less disturbance in solution quality. Of course, this might not always be true and highly depends on the kind of interaction between the variables. The effectiveness of this will be examined later in the experiments in Section V.

```
Input: Solution $\vec{x}$, Probability $p$, Distribution Index $\eta$
Output: Mutated Solution $\vec{y}$
 1: for $i = 1$ to the number of variables in $\vec{x}$ do
 2:     $r \leftarrow$ random(0,1)
 3:     if $r < p$ then
 4:         $u \leftarrow$ random(0,1)
 5:         if $u \leq 0.5$ then
 6:             $\delta_1 = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$
 7:             $\delta_q = (2u + (1-2u)(1-\delta_1)^{\eta+1})^{\frac{1}{\eta+1}} - 1$
 8:         else
 9:             $\delta_2 = \frac{x_{i,max} - x_i}{x_{i,max} - x_{i,min}}$
10:             $\delta_q = 1 - (2(1-u) + 2(u-0.5)(1-\delta_2)^{\eta+1})^{\frac{1}{\eta+1}}$
11:         end if
12:         $y_i = x_i + \delta_q(x_{i,max} - x_{i,min})$
13:         repair($y_i$)
14:     else
15:         $y_i = x_i$
16:     end if
17: end for
18: return $\vec{y}$
```

Fig. 1. Pseudocode of the Polynomial Mutation operator.

C. Grouped Polynomial Mutation

In this part we propose a version of Polynomial Mutation that incorporates variable grouping. The variables are separated into distinct groups and mutation is only applied to a group as a whole entity, i.e. to each variable in a group at the same time. Similar decomposition strategies as in coevolutionary methods might be used, and Section IV later in this work will present two different grouping mechanisms.

In the *Grouped Polynomial Mutation*, the decision of which variables will be changed is transferred from a random mechanism to an “intelligent mechanism” that makes this choice beforehand. This is based on the assumption that a grouping mechanism makes a suitable choice of which variables interact and should be changed at the same time. Especially in non-separable problems, this is supposed to contribute to the search process.

In contrast to Polynomial Mutation, the mutation probability parameter p is no longer needed. Which and how many of the n variables are subject to change is defined by the number of groups. Thus, the *Grouped Polynomial Mutation* needs the additional parameter k, which defines the number of groups. The operator works as follows:

First, the variables of the given solution $\vec{x}$ are split into k distinct (but not necessarily evenly sized) groups by using an arbitrary grouping mechanism. In the next step, one of these k produced groups is chosen uniformly at random. All variables in this chosen group are then subject to the Polynomial Mutation procedure as in Lines 4-13 in Figure 1. That is, for each variable in the group, a **separate** value for u is chosen and the mutation is carried out according to this value. No additional probability value is used. The choice

```
Input: Solution $\vec{x}$, Grouping Mechanism $G$,
       Distribution Index $\eta$
Output: Mutated Solution $\vec{y}$
 1: $\{g_1, ..., g_k\} \leftarrow$ Apply $G$ to $\vec{x}$, producing $k$ groups
 2: $j \leftarrow$ Pick a group index at random from $\{1, ..., k\}$
 3: $u \leftarrow$ random(0,1)
 4: for all variables $x_i \in$ group $g_j$ do
 5:     if $u \leq 0.5$ then
 6:         $\delta_1 = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$
 7:         $\delta_q = (2u + (1-2u)(1-\delta_1)^{\eta+1})^{\frac{1}{\eta+1}} - 1$
 8:     else
 9:         $\delta_2 = \frac{x_{i,max} - x_i}{x_{i,max} - x_{i,min}}$
10:         $\delta_q = 1 - (2(1-u) + 2(u-0.5)(1-\delta_2)^{\eta+1})^{\frac{1}{\eta+1}}$
11:     end if
12:     $y_i = x_i + \delta_q(x_{i,max} - x_{i,min})$
13:     repair($y_i$)
14: end for
15: for all variables $x_i \notin$ group $g_j$ do
16:     $y_i = x_i$
17: end for
18: return $\vec{y}$
```

Fig. 2. Pseudocode of the *Grouped and Linked Polynomial Mutation* operator.

of which variables are changed is solely made by the random choice of the mutated group. All variables in the other groups remain unchanged. Assuming evenly sized groups, the amount of variables that are changed is $\frac{n}{k}$.

D. Grouped and Linked Polynomial Mutation

Finally we propose the *Grouped and Linked Polynomial Mutation*, which combines the concepts used in the previous two subsections. The pseudocode for this combined mutation operator is shown in Figure 2.

A grouping mechanism G and a distribution index η are used as inputs for the operator. As in the *Grouped Polynomial Mutation*, the variables that are to be mutated are chosen based on variable grouping (Line 1 in Figure 2) and picking one group at random (Line 2). Additionally, the value for u is now drawn beforehand (Line 3) like in the *Linked Polynomial Mutation*, so that the amount of change is fixed and equal for all variables in the changed group. Then, all variables in the chosen group are subject to change using the usual Polynomial Mutation procedure and given u.

As explained, variables that strongly interact with each other should usually be altered together, and we presume that it might also be beneficial to alter them by the same amount. Compared to the purely linked version, we can now assume that interacting variables might be altered at the same time (as they are gathered in the same group) and are furthermore altered by the same amount.

IV. Grouping Mechanisms

This section introduces two grouping mechanisms used in this work. As mentioned earlier, it might be desirable to gather variables in the same group that strongly interact with each other in non-separable optimization problems. Similar methods to those in cooperative coevolutionary approaches might be used. We chose to examine two grouping methods in this work. The first one called *Ordered Grouping* is rather simple and does not use any information about the objective functions. The second method is the *Differential Grouping* algorithm developed by Omidvar et al. [4] in 2014. It is used in this work as a representative of more intelligent grouping mechanisms that are based on problem analysis.

A. Ordered Grouping

The *Ordered Grouping* mechanism gathers the decision variables of the selected solution by their absolute values. All variables are sorted based on their absolute values and the $\frac{n}{k}$ variables with the smallest values are assigned to the first group, the next $\frac{n}{k}$ to the second group and so forth.

B. Differential Grouping

Differential Grouping (DG) is a single-objective grouping mechanism that was developed in 2014 [4] and aims to detect variable interaction prior to the optimization of the problem. In short, DG compares the amount of change in the (single) objective function in reaction to a change in a variable x_i when another variable x_j is changed. Interacting variables are gathered into the same group, and one additional group is created that contains all non-separable variables. The amount of interaction is determined using a threshold value ϵ, which is the only parameter in DG. The number of groups as well as their sizes are set automatically by the DG algorithm. A major drawback of the DG algorithm is its computational costs. Assuming the problem contains $k = \frac{n}{l}$ evenly sized groups with l variables each, the number of function evaluations consumed is in $O(\frac{n^2}{l})$. This means for a fully separable problem using $n = 1000$ decision variables, DG needs about $n^2 = 1,000,000$ function evaluation to perform the grouping. Given that our experiments in Section V only use $100,000$ function evaluation for the whole optimization process, this is little practicable. In this work, we use DG for comparison, although it has been developed for single-objective optimization. To make it applicable to MOP, we use the original version and regard only one objective (the first one) of the MOP in the grouping algorithm. We point out that this choice might not be optimal, and the question how to apply DG to MOP in the best way might be a topic for future research. A multi-objective grouping mechanism that takes into account all objective functions might also be used in the future.

V. Evaluation

In this section we evaluate the performance of the proposed mutation operators on large-scale instances of the well-known WFG1-9 test problems [8]. All experiments use two objectives and $n = 1000$ decision variables, which are split into 250 position-related and 750 distance-related variables. Based on the proposed mutation operators and grouping mechanisms, the experiments include seven different configurations of mutation, each of which is tested in the two well-known algorithms NSGA-II [5] and SMPSO [6]. In addition, we compare the performance with the CCGDE3 algorithm[1], which was designed for large-scale optimization in [7]. The used mutation operators are:

1) **Classical**: Polynomial Mutation is used with a probability of $\frac{1}{n}$.
2) **High Probability**: For comparison, Polynomial Mutation is used with a high mutation probability of $\frac{1}{k}$, where k is the number of groups used in the other operators.
3) **Ordered Grouped Polynomial**: *Grouped Polynomial Mutation* as in Section III-C. All variables are divided into k groups using the *Ordered Grouping* in Section IV-A.
4) **Differential Grouped Polynomial**: *Grouped Polynomial Mutation* with *Differential Grouping*.
5) **Linked Polynomial**: *Linked Polynomial Mutation* as in Section III-B with a mutation probability of $\frac{1}{k}$.
6) **Ordered Grouped + Linked**: *Grouped and Linked Polynomial Mutation* as in Section III-D using the *Ordered Grouping* mechanism.
7) **Differential Grouped + Linked**: *Grouped and Linked Polynomial Mutation* with *Differential Grouping*.

Each of these operators is used in two different algorithms, which results in a total of seven SMPSO variants, seven NSGA-II variants, plus the CCGDE3 algorithm. For each experiment we perform 51 independent runs and report the median and IQR values of the relative hypervolume [26] and IGD [27] indicators. The relative hypervolume is the hypervolume obtained by a solution set in relation to the hypervolume obtained by a sample of the Pareto-front of the problem. The used reference point for the hypervolume indicator is obtained by using the nadir point of our Pareto-front sample and multiply it by 2.0 in each dimension. This is done to make sure most of the obtained solutions can contribute to the HV, even when the sets are not close to the optimal front. Statistical significance is tested using a Mann-Whitney U Test and significance is assumed for a value of $p < 0.01$.

The size of the populations is set to 100 and we use $100,000$ function evaluations as a stopping criterion in all experiments. The number of groups is set to $k = 4$, since preliminary experiments showed that this value can provide a reasonable performance. A detailed analysis of different values for k can not be included here due to page limitations. The distribution index used in all operators is set to 20.0. For the operators that use Differential Grouping, the grouping algorithm was run prior to the optimization ($\epsilon = 10^{-5}$). The function evaluations used to obtain the groups are shown in Table III.

A. Results

The results of the SMPSO and NSGA-II experiments are shown in Tables I and II respectively. Best results for the

[1]https://www.cs.cinvestav.mx/~EVOCINV/software/CCLSMO/CCLSMO.html

TABLE I
SMPSO RESULTS (1000 VARIABLES). L = LINKED, G = GROUPED.

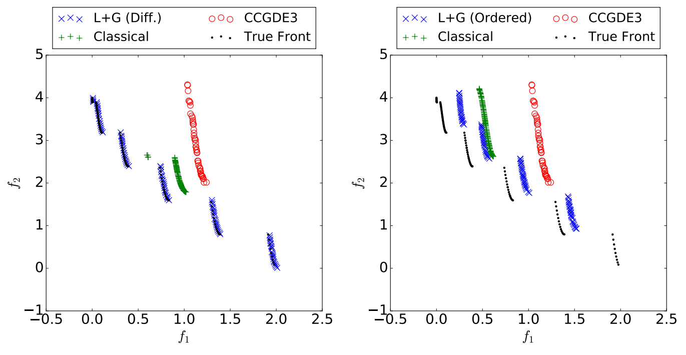
		Relative HV	IGD
WFG1	CCGDE3	*0.367846* (0.008945) *	*0.030674* (0.000507) *
	Classical	0.591757 (0.010107) *	0.016116 (0.000261) *
	High Probability	0.589273 (0.008977) *	0.016170 (0.000256) *
	Grouped (Ordered)	0.593749 (0.007699) *	0.016081 (0.000135) *
	Grouped (Differential)	0.589468 (0.010483) *	0.016186 (0.000297) *
	Linked	0.604692 (0.009476) *	0.015964 (0.000086) *
	L + G (Ordered)	**0.625262** (0.006802)	**0.015505** (0.000128)
	L + G (Differential)	0.592295 (0.007372) *	0.016115 (0.000213) *
WFG2	CCGDE3	*0.619216* (0.004488) *	*0.035341* (0.000453) *
	Classical	0.737396 (0.026523) *	0.029851 (0.003092) *
	High Probability	0.747556 (0.025985) *	0.027509 (0.003128) *
	Grouped (Ordered)	0.761513 (0.009156) *	0.022420 (0.000901) *
	Grouped (Differential)	0.770842 (0.005847) *	0.021737 (0.000494) *
	Linked	0.821493 (0.039492) *	0.014936 (0.001235) *
	L + G (Ordered)	0.970062 (0.008452) *	0.003065 (0.000804) *
	L + G (Differential)	**0.997120** (0.001199)	**0.000539** (0.000162)
WFG3	CCGDE3	*0.543488* (0.006390) *	*0.010261* (0.000178) *
	Classical	0.593086 (0.030444) *	0.009166 (0.002193) *
	High Probability	0.588649 (0.044221) *	0.009354 (0.002886) *
	Grouped (Ordered)	0.662223 (0.015546) *	0.006601 (0.000562) *
	Grouped (Differential)	0.584304 (0.019029) *	0.009713 (0.001991) *
	Linked	0.727292 (0.001564) *	0.004838 (0.000081) *
	L + G (Ordered)	**0.839046** (0.004231)	**0.003110** (0.000084)
	L + G (Differential)	0.728010 (0.000669) *	0.004805 (0.000035) *
WFG4	CCGDE3	*0.508533* (0.006340) *	*0.011513* (0.000823) *
	Classical	0.844238 (0.004784) *	0.003153 (0.000228) *
	High Probability	0.844314 (0.004439) *	0.003101 (0.000207) *
	Grouped (Ordered)	0.876147 (0.007694) *	0.002117 (0.000315) *
	Grouped (Differential)	0.843023 (0.002552) *	0.003172 (0.000140) *
	Linked	0.844032 (0.004020) *	0.003121 (0.000139) *
	L + G (Ordered)	**0.954334** (0.009066)	**0.000794** (0.000055)
	L + G (Differential)	0.849472 (0.008305) *	0.003191 (0.000209) *
WFG5	CCGDE3	*0.540655* (0.008380) *	*0.011037* (0.000504) *
	Classical	0.804578 (0.004674) *	0.006546 (0.000239) *
	High Probability	0.806261 (0.005083) *	0.006439 (0.000403) *
	Grouped (Ordered)	0.815496 (0.009366) *	0.005553 (0.000660) *
	Grouped (Differential)	0.804038 (0.005120) *	0.006569 (0.000271) *
	Linked	0.818252 (0.005702) *	0.006424 (0.000336) *
	L + G (Ordered)	**0.903456** (0.009612)	**0.001757** (0.000308)
	L + G (Differential)	0.833800 (0.006405) *	0.006644 (0.000358) *
WFG6	CCGDE3	*0.401075* (0.006442) *	*0.029259* (0.000705) *
	Classical	0.968987 (0.003734) *	0.005422 (0.000553) *
	High Probability	0.968228 (0.003379) *	0.005483 (0.000467) *
	Grouped (Ordered)	0.996617 (0.000853) *	0.000249 (0.000081) *
	Grouped (Differential)	**0.996977** (0.000313)	**0.000223** (0.000003)
	Linked	0.994513 (0.002721) *	0.000503 (0.000403) *
	L + G (Ordered)	0.996790 (0.000122) *	0.000228 (0.000005) *
	L + G (Differential)	0.996829 (0.000258)	0.000226 (0.000008)
WFG7	CCGDE3	*0.524629* (0.007656) *	*0.005185* (0.000180) *
	Classical	0.698961 (0.016047) *	0.003365 (0.000197) *
	High Probability	0.717487 (0.018648) *	0.003189 (0.000146) *
	Grouped (Ordered)	0.726447 (0.018814) *	0.002997 (0.000204) *
	Grouped (Differential)	0.709394 (0.020161) *	0.003258 (0.000178) *
	Linked	0.758392 (0.009962) *	0.003428 (0.000190) *
	L + G (Ordered)	**0.947363** (0.014633)	**0.000480** (0.000147)
	L + G (Differential)	0.774369 (0.006558) *	0.004091 (0.000682) *
WFG8	CCGDE3	*0.464333* (0.005882) *	*0.017667* (0.000780) *
	Classical	0.515900 (0.000816) *	0.015897 (0.000083) *
	High Probability	0.515196 (0.000601) *	0.015960 (0.000062) *
	Grouped (Ordered)	0.515288 (0.001891) *	0.015934 (0.000196) *
	Grouped (Differential)	0.515049 (0.000507) *	0.015964 (0.000040) *
	Linked	0.783168 (0.028670) *	0.004620 (0.000590) *
	L + G (Ordered)	0.859717 (0.020352) *	**0.003040** (0.000349)
	L + G (Differential)	**0.893367** (0.010179)	0.004071 (0.000718) *
WFG9	CCGDE3	*0.475881* (0.008748) *	*0.009960* (0.000320) *
	Classical	0.897946 (0.007659) *	0.001871 (0.000199) *
	High Probability	0.896914 (0.007444) *	0.001875 (0.000210) *
	Grouped (Ordered)	0.955709 (0.014401) *	0.000320 (0.000178) *
	Grouped (Differential)	0.894752 (0.009126) *	0.001938 (0.000178) *
	Linked	0.897048 (0.005813) *	0.001936 (0.000168) *
	L + G (Ordered)	**0.961435** (0.011490)	**0.000241** (0.000098)
	L + G (Differential)	0.896248 (0.007373) *	0.001981 (0.000221) *

TABLE II
NSGA-II RESULTS (1000 VARIABLES). L = LINKED, G = GROUPED.

		Relative HV	IGD
WFG1	CCGDE3	0.367846 (0.008945) *	0.030674 (0.000507) *
	Classical	0.315101 (0.002685) *	0.035935 (0.000161) *
	High Probability	*0.299124* (0.002227) *	*0.036327* (0.000173) *
	Grouped (Ordered)	0.306141 (0.003979) *	0.035924 (0.000318) *
	Grouped (Differential)	0.311402 (0.002919) *	0.035793 (0.000208) *
	Linked	**0.582779** (0.011338)	**0.016350** (0.000514)
	L + G (Ordered)	0.522169 (0.034854) *	0.020015 (0.002242) *
	L + G (Differential)	0.367080 (0.008066) *	0.031288 (0.000607) *
WFG2	CCGDE3	*0.619216* (0.004488) *	*0.035341* (0.000453) *
	Classical	0.667958 (0.048709) *	0.032840 (0.007741) *
	High Probability	0.663945 (0.059934) *	0.032449 (0.007978) *
	Grouped (Ordered)	0.680277 (0.067914) *	0.032052 (0.009224) *
	Grouped (Differential)	0.658211 (0.060273) *	0.032529 (0.007460) *
	Linked	0.732674 (0.004360) *	0.023569 (0.000360) *
	L + G (Ordered)	**0.873795** (0.006747)	**0.010968** (0.000461)
	L + G (Differential)	0.828344 (0.004611) *	0.015758 (0.000492) *
WFG3	CCGDE3	*0.543488* (0.006390) *	*0.010261* (0.000178) *
	Classical	0.611701 (0.007014) *	0.008346 (0.000286) *
	High Probability	0.632486 (0.006078) *	0.007351 (0.000197) *
	Grouped (Ordered)	0.656521 (0.006352) *	0.006749 (0.000219) *
	Grouped (Differential)	0.608953 (0.006612) *	0.007983 (0.000187) *
	Linked	0.713287 (0.002506) *	0.005204 (0.000104) *
	L + G (Ordered)	**0.796826** (0.004235)	**0.003569** (0.000080)
	L + G (Differential)	0.727341 (0.000042) *	0.004829 (0.000001) *
WFG4	CCGDE3	*0.508533* (0.006340) *	*0.011513* (0.000823) *
	Classical	0.622804 (0.013154) *	0.004845 (0.000293) *
	High Probability	0.654599 (0.008638) *	0.004319 (0.000200) *
	Grouped (Ordered)	0.672081 (0.009119) *	0.003965 (0.000214) *
	Grouped (Differential)	0.598239 (0.005898) *	0.005773 (0.000183) *
	Linked	0.666287 (0.050934) *	0.004399 (0.000234) *
	L + G (Ordered)	**0.847588** (0.007393)	**0.003086** (0.000259)
	L + G (Differential)	0.797889 (0.054696) *	0.003299 (0.001210) *
WFG5	CCGDE3	*0.540655* (0.008380) *	*0.011037* (0.000504) *
	Classical	0.590757 (0.010342) *	0.008853 (0.000463) *
	High Probability	0.657639 (0.007488) *	0.006783 (0.000240) *
	Grouped (Ordered)	0.687127 (0.009560) *	0.005702 (0.000202) *
	Grouped (Differential)	0.583821 (0.006670) *	0.009511 (0.000207) *
	Linked	0.833456 (0.006655) *	0.003552 (0.000145) *
	L + G (Ordered)	**0.870639** (0.005913)	**0.002239** (0.000155)
	L + G (Differential)	0.814990 (0.004686) *	0.004536 (0.000174) *
WFG6	CCGDE3	*0.401075* (0.006442) *	*0.029259* (0.000705) *
	Classical	0.575982 (0.013639) *	0.015226 (0.001135) *
	High Probability	0.557367 (0.010745) *	0.016532 (0.000704) *
	Grouped (Ordered)	0.581868 (0.011794) *	0.015111 (0.000663) *
	Grouped (Differential)	0.575970 (0.009301) *	0.015239 (0.000502) *
	Linked	0.566894 (0.015396) *	0.015281 (0.001262) *
	L + G (Ordered)	**0.807439** (0.013516)	**0.006365** (0.000257)
	L + G (Differential)	0.794394 (0.014941) *	0.008135 (0.000499) *
WFG7	CCGDE3	*0.524629* (0.007656) *	*0.005185* (0.000180) *
	Classical	0.647512 (0.008260) *	0.003732 (0.000157) *
	High Probability	0.696697 (0.009147) *	0.003328 (0.000065) *
	Grouped (Ordered)	0.702850 (0.007463) *	0.003070 (0.000075) *
	Grouped (Differential)	0.670489 (0.009197) *	0.004065 (0.000079) *
	Linked	0.668523 (0.009633) *	0.003611 (0.000226) *
	L + G (Ordered)	**0.859782** (0.005282)	**0.002158** (0.000305)
	L + G (Differential)	0.647380 (0.007890) *	0.003985 (0.000264) *
WFG8	CCGDE3	*0.464333* (0.005882) *	*0.017667* (0.000780) *
	Classical	0.550664 (0.007526) *	0.012517 (0.000550) *
	High Probability	0.593013 (0.008607) *	0.010872 (0.000402) *
	Grouped (Ordered)	0.605128 (0.008846) *	0.010129 (0.000407) *
	Grouped (Differential)	0.559252 (0.009561) *	0.012270 (0.000602) *
	Linked	0.704746 (0.018509) *	0.007971 (0.000465) *
	L + G (Ordered)	0.829897 (0.015840) *	0.004912 (0.000264)
	L + G (Differential)	**0.859430** (0.011909)	**0.004795** (0.000284)
WFG9	CCGDE3	*0.475881* (0.008748) *	*0.009960* (0.000320) *
	Classical	0.620250 (0.022098) *	0.005063 (0.000478) *
	High Probability	0.625662 (0.012819) *	0.004958 (0.000206) *
	Grouped (Ordered)	0.634514 (0.013140) *	0.004792 (0.000261) *
	Grouped (Differential)	0.505976 (0.006625) *	0.008087 (0.000170) *
	Linked	0.729095 (0.093076) *	0.003221 (0.000941) *
	L + G (Ordered)	**0.883259** (0.062866)	**0.001409** (0.000933)
	L + G (Differential)	0.819419 (0.058882) *	0.002754 (0.000289) *

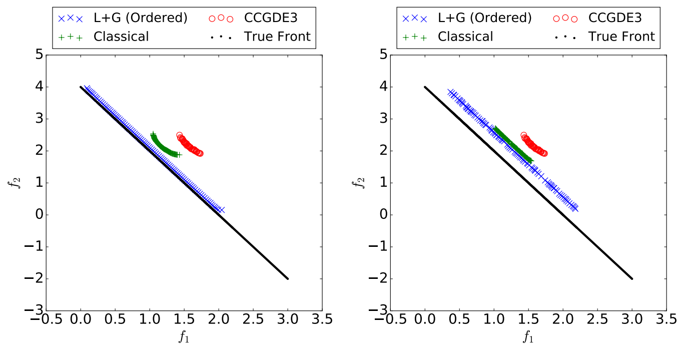
TABLE III
OBTAINED GROUPS AND USED FUNCTION EVALUATIONS BY THE DIFFERENTIAL GROUPING ALGORITHM ($\epsilon = 10^{-5}$).

	# Groups	# Eval.	Comment
WFG1	37	639,894	—
WFG2	2	565,250	Exactly split distance and position variables
WFG3	1	1,001,000	One group of "separable" variables
WFG4	1	1,001,000	One group of "separable" variables
WFG5	1	1,001,000	One group of "separable" variables
WFG6	2	565,250	Exactly split distance and position variables
WFG7	1	1,001,000	One group of "separable" variables
WFG8	2	949,702	—
WFG9	1	1,001,000	One group of "separable" variables



(a) SMPSO (b) NSGA-II

Fig. 3. Selected algorithms' solution sets for the WFG2 problem (Runs with median hypervolume)

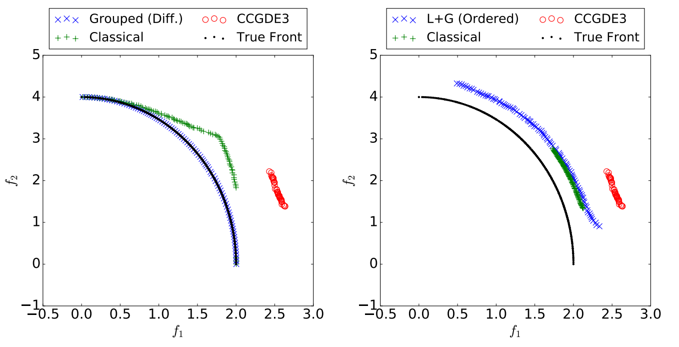


(a) SMPSO (b) NSGA-II

Fig. 4. Selected algorithms' solution sets for the WFG3 problem (Runs with median hypervolume)

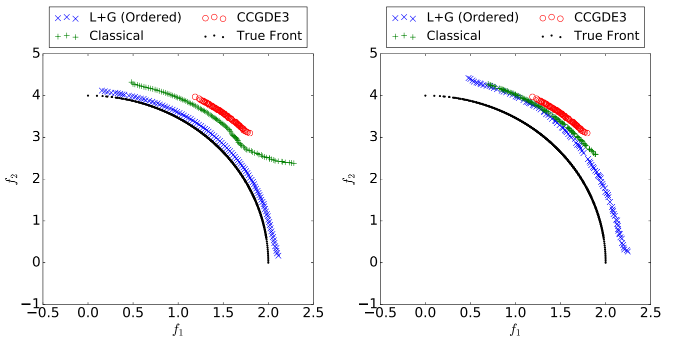
respective problem and indicator are shown in bold. The statistical significance (shown by an asterisk) of each entry is calculated against the respective best performance for each problem. The worst performance is shown in italic font. In addition to these tables, we show the runs that obtained the median hypervolume values of the SMPSO for the WFG2, 3, 6 and 7 problems in Figures 3-6.

For each of the nine problems, the best performance in terms of hypervolume is obtained by one of the algorithms with the linked and grouped operators. The only exceptions are NSGA-



(a) SMPSO (b) NSGA-II

Fig. 5. Selected algorithms' solution sets for the WFG6 problem (Runs with median hypervolume)



(a) SMPSO (b) NSGA-II

Fig. 6. Selected algorithms' solution sets for the WFG7 problem (Runs with median hypervolume)

II for the WFG1 problem, where the linked mutation obtains a higher value than the respective grouped operators, and SMPSO for the WFG6, where the grouped operator performs best. The results of the SMPSO experiments in Table I are in general superior to those of the NSGA-II algorithm in Table II. For both algorithms, we see a statistically significant improvement in performance when the linked and grouped mutation operators are used compared to the classical versions. This holds in terms of hypervolume as well as spread of the solutions (indicated by the IGD values). The classical mutation operator and its high probability version are outperformed significantly for all problems and both algorithms.

1) Linked Operator: We now take a closer look and compare the performance of the *Linked Polynomial Mutation* with the high probability version. Both operators use the same probability of $\frac{1}{k}$ which results in an equal expected amount of change. In Table I for the SMPSO, we observe a superior performance of the linked operator for all problems except WFG4, where both version perform equally well. The same is true for the NSGA-II results in Table II, where the linked operator is superior for the WFG1-6, 8 and 9 problems. By this, we can see that even when mutating random variables without using variable grouping, there is chance that interacting variables are changed at the same time by the operator, and that a link between these changes leads to a better result of the optimization.

2) Grouped Operators: Next we take a look at the operators that involve variable grouping. Overall, the best hypervolume values for all of the problems in Table I are obtained by the linked and grouped operators. The linked operator using the ordered grouping performs best and obtains the best hypervolume and IGD values for the WFG1, 3, 4, 5, 7 and 9 problems. For the WFG2, 6 and 8 problems, the Differential Grouping performs better in terms of hypervolume and IGD values, and for WFG6, the non-linked version performs best with just an insignificant difference to the linked version. Moreover, the grouped versions of the linked operator perform significantly better than the linked operator without variable grouping. With the exception of WFG1, this is also true for the NSGA-II results.

3) Differential Grouping: Next we take a closer look on the performance of Differential Grouping (DG). Out of all nine problems, the WFG2, 6 and 8 were the only ones, where the DG-enhanced operators were able to outperform all other methods. The hypervolume (and most IGD) values of the linked DG operator were significantly better than the linked Ordered Grouping ones for WFG2, 6 and 8 when SMPSO was used (Table I) and WFG8 when NSGA-II was used (Table II).

To analyze this behaviour, Table III lists the number of obtained groups of the DG algorithm for each problem. The WFG2 and 6 are the only ones that are split into two groups, where the first one contains the first 250 decision variables while the second group contains the last 750 variables. As mentioned in Section V, the 1000 decision variables in our experiments are split into 250 position-related and 750 distance-related variables. Thus the DG mechanism recognizes the different properties of the variable types for the WFG2 and WFG6 problems and split these variables apart. The obtained groups for the WFG8 problem look different. Here, the DG algorithm creates two groups as well, with group one containing the variables number 1 and 251 - 276 and group two the rest of the variables. Hence, the DG put 1 position variable into the same group with 26 distance-related variables. The performance for the WFG8 however is supporting this choice.

The performance of the DG algorithm compared to the simple Ordered Grouping mechanism, especially for the WFG2, 6 and 8 problems, emphasizes that the usage of a suitable grouping mechanism is beneficial for the search. We also observe (Table III) that the DG algorithm considers all 1000 variables of the WFG3-5, 7 and 8 problems as separable. By default DG gathers separable variables in the same group, which results in just one single group for WFG3-5, 7 and 8. In contrast, the variables of the WFG1 were split into 37 groups. Some of these observations are in conflict with the statements about separability in [8]. We need to emphasize here, that DG is sensitive to its parameter ϵ and was developed for single-objective problems. The obtained groups in this work are only based on the first objective function, and a perfect performance can therefore not be expected. Seeing the good performance for the WFG2, 6 and 8, however, the need for suitable multi-objective grouping mechanisms in future algorithms is evident.

4) CCGDE3: At last, we take a look at the CCGDE3's performance. The CCGDE3 was designed specifically to optimize large-scale multi-objective problems. However, the good results that were reported for the ZDT problems in [7], which were also obtained in our preliminary computational experiments, cannot be obtained for the WFG benchmarks in this work. In Table I, we observe that the worst values for hypervolume are obtained by the CCGDE3 algorithm for all test problems. The same is true when the CCGDE3 is compared to NSGA-II in Table II. The only problem where it can achieve a higher hypervolume value than the original NSGA-II algorithm is the WFG1. In conclusion, the CCGDE3 optimizer cannot match the high performance in both hypervolume and IGD values of any of the grouped and/or linked operator versions of the NSGA-II or SMPSO algorithms, and is in most instances even outperformed by their respective classical operator versions.

VI. Conclusion

We examined the effect of variable grouping in mutation operators for large-scale multi-objective optimization. Three new mutation operators based on the well-known Polynomial Mutation were proposed. These operators make use of variable grouping mechanisms as well as linking the amount of change of the variables together. The results of our experiments with the WFG1-9 test problems using $n = 1000$ variables indicate that using linked and grouped mutation operators can significantly improve the performance of existing metaheuristics for large-scale optimization. Future experiments might involve the examination of the proposed operators for small-scale problems as well as instances larger than the ones tested here. The use of more sophisticated multi-objective grouping schemes might also be a subject for future work.

Acknowledgment

This work was partly funded by the German Academic Exchange Service (DAAD).

References

[1] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, no. 1, pp. 30–45, 1996.

[2] M. A. Potter and K. A. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature PPSN III*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1994, vol. 866, pp. 249–257.

[3] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "Weighted optimization framework for large-scale multi-objective optimization," in *Companion of Genetic and Evolutionary Computation Conference - GECCO*, 2016.

[4] M. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[6] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *IEEE Symposium on Computational Intelligence in Multi-criteria Decision-making*. IEEE, 2009, pp. 66–73.

[7] L. M. Antonio and C. A. Coello Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 2758–2765.

[8] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[9] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Information Sciences*, vol. 295, pp. 407–428, 2015.

[10] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

[11] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving from Nature, PPSN XI*, ser. Lecture Notes in Computer Science, R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, Eds. Springer Berlin Heidelberg, 2010, vol. 6239, pp. 300–309.

[12] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2009, pp. 1546–1553.

[13] ——, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, 2012.

[14] Z. Yang, J. Zhang, K. Tang, X. Yao, and A. C. Sanderson, "An adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2009, pp. 102–109.

[15] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 1. IEEE, 2005, pp. 443–450.

[16] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[17] A. W. Iorio and X. Li, "A cooperative coevolutionary multiobjective algorithm using non-dominated sorting," in *Genetic and Evolutionary Computation Conference - GECCO*. Springer, 2004, pp. 537–548.

[18] S. K. Goh, K. C. Tan, A. Al-Mamun, and H. A. Abbass, "Evolutionary big optimization (BigOpt) of signals," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 3332–3339.

[19] Y. Zhang, J. Liu, M. Zhou, and Z. Jiang, "A multi-objective memetic algorithm based on decomposition for big optimization problems," *Memetic Computing*, vol. 8, no. 1, pp. 45–61, 2016.

[20] S. Elsayed and R. Sarker, "An adaptive configuration of differential evolution algorithms for big data," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 695–702.

[21] M. N. Omidvar, Y. Mei, and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1305–1312.

[22] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Genetic and Evolutionary Computation Conference - GECCO*. ACM Press, 2015, pp. 313–320.

[23] M. Hamdan, "On the disruption-level of polynomial mutation for evolutionary multi-objective optimisation algorithms," *Computing and Informatics*, vol. 29, no. 5, pp. 783–800, 2010.

[24] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008.

[25] K. Deb and D. Deb, "Analysing mutation schemes for real-parameter genetic algorithms," *International Journal of Artificial Intelligence and Soft Computing*, vol. 4, no. 1, pp. 1–28, 2014.

[26] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms a comparative case study," *Parallel Problem Solving from Nature PPSN V*, 1998.

[27] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multi objective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003.