A Framework for Large-scale Multi-objective Optimization based on Problem Transformation

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 Engineering
Southern University of Science and Technology, Shenzhen, China
Email: hisao@sustc.edu.cn
‡Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University, Sakai, Osaka 599-8531, Japan
Email: nojima@cs.osakafu-u.ac.jp

***Abstract*—In this work we propose a new method for solving multi-objective optimization problems with a large number of decision variables. The proposed method called Weighted Optimization Framework is intended to serve as a generic method that can be used with any population-based metaheuristic algorithm. After explaining some general issues of large-scale optimization, we introduce a problem transformation scheme that is used to reduce the dimensionality of the search space and search for improved solutions in the reduced subspace. This involves so-called weights that are applied to alter the decision variables and are also subject to optimization. Our method relies on grouping mechanisms and employs a population-based algorithm as an optimizer for both original variables and weight variables. Different grouping mechanisms and transformation functions within the framework are explained and their advantages and disadvantages are examined. Our experiments use test problems with 2 - 3 objectives 40 - 5000 variables. Using our approach on three well-known algorithms and comparing its performance with other large-scale optimizers, we show that our method can significantly outperform most existing methods in terms of solution quality as well as convergence rate on almost all tested problems for many-variable instances.**

***Index Terms*—Multi-objective optimization, large-scale optimization, variable Grouping, many-variable optimization, weighting, metaheuristic Framework**

I. Introduction

In the area of optimization, a growing interest in so-called *large-scale optimization* (LSO) can be observed [1]. This area of research deals with solving single- or multi-objective problems that include a large number of decision variables. The performance of classic metaheuristic algorithms often deteriorates when the dimensionality of the decision space increases. When large numbers of variables are involved, algorithms are faced with very high-dimensional search spaces that are difficult to explore with limited resources.

Among large-scale problems, this work focuses on multi-objective optimization problems that involve a large number of decision variables (many-variable problems). While single-objective large-scale optimization has already been extensively studied for many years [1], less work has been done to deal with multi-objective many-variable problems. Recently in 2015, a competition with single and multiple objectives based on a real-world EEG time series [2] data was held on the IEEE Congress on Evolutionary Computation (CEC) that involved up to 4864 decision variables. These kinds of applications emphasize the need for better algorithms to deal with large numbers of decision variables.

In this work, we propose a new optimization framework called *Weighted Optimization Framework* (WOF) which is based on a problem transformation, optimizing multiple weight vectors, and variable grouping. In our previous work [3] we briefly introduced the basic idea of WOF in a two-page short paper. The present article introduces and explains the WOF in detail by providing analysis based on theory and experimental data. The experiments performed in this article use a larger set of benchmark functions and algorithms. Furthermore, we go into detail about different grouping mechanisms and transformation functions that can be used within the framework. The performance for different numbers of variables as well as further sensitivity analysis is provided. Finally, the results are compared with other large-scale optimization techniques.

The WOF is a new algorithm framework that enables population-based metaheuristics to solve many-variable problems more effective and efficiently. In this work, we show that the WOF can strongly enhance the performance of population-based metaheuristic algorithms to optimize many-variable multi-objective optimization problems. We provide an in-depth analysis of its functionality and give insight into its capabilities and limitations from a theoretical perspective. Additionally we show that it can significantly outperform existing classical and large-scale algorithms on the tested multi-objective WFG [4], ZDT [5], DTLZ [6] and CEC2009 competition [7] problems with large numbers of variables.

This paper is organized as follows. In Section II we give a short introduction about multi-objective optimization. Section III deals with occurring difficulties in LSO and gives a short overview of existing approaches for single- and multi-objective many-variable optimization. In Section IV we propose the concept of the problem transformation, followed by explaining the details of the proposed optimization framework. Section V deals with different grouping mechanisms and transformation functions that are used in the framework. In our experiments

in Section VI we use the WOF framework on three well-known algorithms (NSGA-II [8], SMPSO [9] and GDE3 [10]) and compare their performances with each other and their original algorithm versions on a variety of well-known benchmark functions. Additionally, the multi-objective large-scale algorithms CCGDE3 [11], which is based on coevolution, and MOEA/DVA [12] are used for comparison. Finally, in Section VII, we conclude this paper.

II. Multi-objective Optimization

Many problems in the real world can be formulated as mathematical problems with certain goals (objectives) which have to be optimized. Additionally, some of these problems have constraints. If there are more than just one objective to optimize, such a problem is called a multi-objective problem (MOP). This can be formulated as follows:

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

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 M of dimension m. It is important to solve such a problem with or without constraints as accurate as possible. Since there are more than one objective, a single optimal solution can no longer be determined. Instead, modern problem solving methods concentrate on finding a Pareto-optimal solution set. However, because of the complexity of multi-objective problems in real-world applications and limited computational resources, the true Pareto-front can hardly be analytically or exactly computed. Instead, heuristic algorithms are developed and used to find a non-dominated solution set as good as possible to approximate the true Pareto-front.

III. Large-scale Optimization

Large-scale optimization deals with optimizing problems that contain large numbers of variables (many-variable problems). Recently, a wide variety of single-objective many-variable optimizers have been proposed in the literature (see for instance the review by Mahdavi et al. [1]). Among others a competitive single-objective PSO algorithm [13] was developed in 2015 which showed good results for large-scale benchmark problems with up to 5000 decision variables. The majority of large-scale optimizers tend to use a concept called Cooperative Coevolution (CC). Since the pioneering work of Potter and De Jong [14], [15], the idea of CC has been used in various large-scale optimization algorithms. CC aims to optimize several independent populations, of which each holds a subset of the n decision variables. New solution candidates have to be formed by a combination of the variable values from different subcomponents, but genetic operators are used within one subcomponent only.

Several authors have used this concept for single-objective problems [16]–[21] and some work on multiple objectives also relied on coevolution [11], [22]. Regarding the distribution of the variables into subcomponents or groups, consideration for non-separable problems was studied in [17], where the interaction of variables was taken into account by a learning mechanism for finding the optimal distribution of variables to the subcomponents. In [23] a mechanism called Differential Grouping (DG) was introduced to find improved distributions of the variables in single-objective CC algorithms. Other concepts and extensions to this approach for variable grouping have been proposed [24] - [25]. In Section V, we go into detail on DG and other grouping mechanisms.

Our present work has been inspired by Yang et al. [16], where they applied a CC method to single-objective problems using two special features. One is a repeated grouping and regrouping of the variables into subcomponents in every iteration of the algorithm. The other is a weighting scheme to optimize a weight for each subcomponent, i.e. apply the same weight value (in terms of multiplication) to every variable in the same subcomponent. These weights were evolved for the best, worst and a random member of the population. The frequent (re-)grouping in [16] was done randomly. The aspect that was a source of inspiration and that was adapted for the present work was their mechanism of optimizing a number of variables (in each subcomponent) at the same time by altering one weight value only. The same principle of using CC with weights has also been used in [18], [19]. A mechanism for choosing appropriate lower and upper bounds of weights was proposed in [18]. More recently, [20] stated that using the weighting approach in CC is less effective than improving the frequent (re-)grouping of variables. The above-mentioned CC-based studies used benchmark functions of up to 1000 decision variables.

In contrast to the above-mentioned and numerous other studies on large-scale single-objectives problems, less work has been conducted so far regarding multi-objective many-variable problems. In the area of multi-objective CC, good performance was reported in [11] for the ZDT [5] problems where CC was combined with a differential evolution algorithm called GDE3 [10]. They performed experiments with decision variables ranging from 200 up to 5000 and received good performance especially for the higher dimensional problems. Nevertheless, the performance of the proposed method on more complicated benchmarks like the WFG toolkit or the DTLZ benchmarks has not been analyzed. Furthermore, although outperforming traditional methods by far, good approximations of the ZDT Pareto-front still needed a large number of function evaluations ranging between $150,000$ to over $2,000,000$ evaluations. An earlier approach by Iorio and Li [22] combined the concept of CC with the NSGA-II algorithm, but also focused on the ZDT problems for their analysis and tested rather small-dimensional instances of 10 and 30 variables.

For the real-world application in [2], good results compared to classical methods are shown in [26]–[28]. However, [28] did only use the single-objective version of the problem, and both [28] and [26] did not treat the optimization problem in a black-box manner, as they used information obtained by analytical derivation of the objective functions. None of these three approaches used decomposition or CC approaches, and the performance on popular benchmarks as the ones frequently used by the community and in this work have not been reported.

Two more very recent approaches to multi-objective large-

scale optimization are introduced in [12] and [29]. In [12], a large-scale optimizer called MOEA/DVA is presented with the main focus on a new grouping mechanism, which aims to identify whether a decision variable contributes to convergence (i.e. distance to the Pareto-front), diversity, or both. Additionally, interactions between the variables are identified. Combining this information, a number of convergence-related, diversity-related as well as mixed groups are formed. The following optimization procedure of the MOEA/DVA focusses on the convergence/distance-variables first, before concentrating on the diversity. The authors of [12] reported a good performance for 200-variable instances of the UF1-6, UF10, ZDT4 and 3-objective DTLZ1 and DTLZ3 benchmark functions. The grouping mechanism in this approach consumes a large amount of function evaluations prior to the start of the optimization, especially for large numbers of variables (refer to Subsection VI-E).

The most recent approach found to solve many-variable MOPs is called LMEA [29]. This method relies on a clustering approach to form groups of variables. It divides the variables into distance- and diversity-related groups. The experiments focused on many-variable instances and used selected 5- and 10-objective DTLZ, WFG and UF benchmark functions. LMEA showed good performance for 100, 500 and 1000 variable instances. It performed superior to the previously mentioned MOEA/DVA for the DTLZ1 and 2 (5-obj.), DTLZ5-7 and WFG3 problems, and worse on the UF9 and 10 as well as the DTLZ4 (10-obj.) problems.

IV. Proposed Method

The aim of our proposed method is to reduce the dimensionality of optimization problems, so that existing algorithms are able to deal with them in a much more efficient manner. In contrast to the CC-based studies mentioned above, our proposed method is not using coevolution, but extending the concept of weighting the variables to multiple objectives instead. Similar to CC, our method distributes the decision variables to groups, but in contrast to CC it does not aim to optimize them independently. Our aim is to provide a multi-objective framework that makes use of the concept of optimizing smaller subproblems in a different way than current coevolutionary methods. Note that we do not intend to propose a new or improved grouping mechanism. The focus lies on proposing a framework that makes use of an arbitrary grouping mechanism and that is generic so that any grouping mechanism from the literature can be used. In this section, we first explain the concept of problem transformation and then continue with the proposed WOF algorithm that makes use of this transformation.

A. Problem Transformation

Let Z be a multi-objective optimization problem with n decision variables and m objectives as denoted earlier in Equation 1. We are interested in the set of Pareto-optimal solutions $P := \{\vec{x}^*\}$ (Pareto-set) and the corresponding objective values $\{\vec{f}(\vec{x}^*)\}$ (Pareto-front). For any solution vector $\vec{x}$, we can write $\vec{f}(\vec{x})$ as $f(\psi(\vec{w}_I, \vec{x}))$, with $\vec{w}_I = (1, ..., 1)$ a vector of ones and ψ the transformation function $\psi(\vec{w}_I, \vec{x}) := (w_{I1}x_1, ..., w_{In}x_n)$. To change the given solution we can now change the values of $\vec{w}_I$. In this way, instead of optimizing the vector $\vec{x}$, for any fixed real values of $\vec{x}$ we can optimize a vector $\vec{w}$ to approximate an optimal solution. It can easily be seen that by doing so, an optimal solution to the optimization problem can be found (See Equation 2). For the optimal weights we obtain $\vec{f}(\psi(\vec{w}^*, \vec{x})) = \vec{f}(\vec{x}^*)$ for a fixed $\vec{x}$.

$$\begin{aligned} &\forall x_i : \exists w_i : w_i x_i = x_i^* \\ &\qquad\Rightarrow \\ &\forall \vec{x} \in \mathbb{R}^n : \exists \vec{w} : \psi(\vec{w}, \vec{x}) = \vec{x}^* \end{aligned} \tag{2}$$

By taking a fixed solution $\vec{x}'$, we can reformulate the original optimization problem Z to a new problem $Z_{\vec{x}'}$ which optimizes the vector $\vec{w}$ using $\vec{x}'$ as an input parameter:

$$\begin{aligned} Z_{\vec{x}'} : min \quad & \vec{f}_{\vec{x}'}(\vec{w}) = (f_{1,\vec{x}'}(\vec{w}), ..., f_{m,\vec{x}'}(\vec{w}))^T \\ s.t. \quad & \vec{w} \in \Phi \subseteq \mathbb{R}^n \\ & f_{o,\vec{x}'}(\vec{w}) = f_o(\psi(\vec{w}, \vec{x}')) \quad \forall o = 1, ..., m \\ & \psi(\vec{w}, \vec{x}') := (w_1 x'_1, ..., w_n x'_n) \end{aligned} \tag{3}$$

This new optimization problem has still the same number of decision variables as the original one. To reduce the dimensionality of the problem, we divide the original variables into γ groups $(g_1, ..., g_\gamma)$, each of size l $(\gamma \cdot l = n)$. Furthermore, instead of assigning one w_i for each x_i, only one weight w_j is used for all variables within the same group g_j with l variables. Therefore we reformulate the above-mentioned transformation function as:

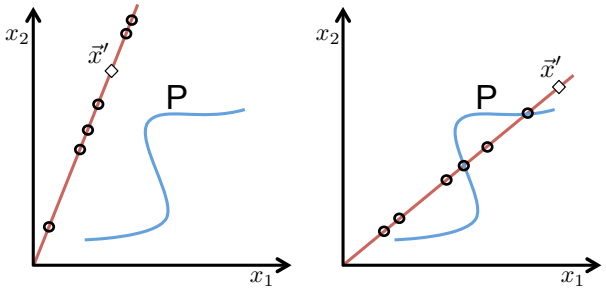
$$\psi(\vec{w}, \vec{x}) = (\underbrace{w_1 x_1, ..., w_1 x_l}_{w_1}, ..., \underbrace{w_\gamma x_{n-l+1}, ..., w_\gamma x_n}_{w_\gamma}) \tag{4}$$

In this way, the vector $\vec{w}$ has only a reduced size of $\gamma < n$ (same as the number of groups), and therefore our new optimization problem $Z_{\vec{x}'}$ has only γ decision variables. For an arbitrary but fixed solution $\vec{x}'$, we can now try to compute an approximation of the Pareto-set by optimizing this new optimization problem instead.

Up to this point there is a drawback to this approach, since by grouping the original variables together, their values cannot be changed independently of each other any more. This limits the reachable solutions in the original search space to a great extent. The searchable subspace of Ω, which can be reached by optimizing $Z_{\vec{x}'}$, is defined mainly by three choices:

1) The choice of $\vec{x}'$
2) The choice of the grouping scheme, i.e. the decision of which variables are put together in the same group
3) The choice of the transformation function $\psi(.)$

In the remainder of this work we will explain some possible approaches to each of these three choices in further detail. First, an appropriate choice of $\vec{x}'$ is crucial for the success of the optimization since it defines a "direction" of the future search. This shall be visualized by a small example. Consider a simple optimization problem with just two decision variables x_1 and x_2. In Figure 1 we show the search space of this problem, as well as a hypothetical (Pareto-)optimal set denoted



(a) "Suboptimal" choice of $\vec{x}'$ (b) "Optimal" choice of $\vec{x}'$

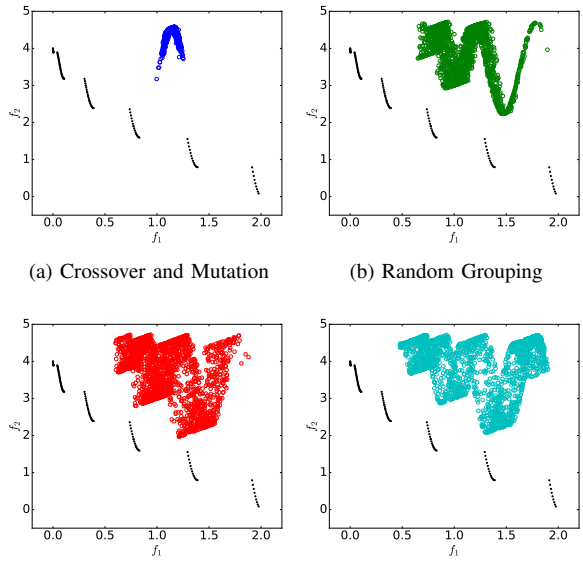
Fig. 1. Examples of variable grouping and a choice of $\vec{x}'$ in the decision space. Solutions on the red straight line can be obtained after x_1 and x_2 are grouped together. P denotes the Pareto-optimal set in the decision space. Note that even the suboptimal choice on the left might be beneficial to the search, as it allows to advance closer to certain parts of the Pareto-Set.

as P. When optimizing the original problem Z, x_1 and x_2 are changed independently of each other and all of the solutions in the search space can be found. Therefore a good approximation of P can be obtained. The effect of grouping x_1 and x_2 together and forming a new problem $Z_{\vec{x}'}$ will result in the optimization of just one single variable w_1. This limits the reachable solutions to a 1-dimensional subspace (denoted as a red line from $\vec{x}'$ to the origin in Figure 1). Comparing the scenes of the Figures 1a and 1b, we see that some optimal solutions can or cannot be reached depending on the choice of $\vec{x}'$. Carrying this knowledge to higher dimensional problems, every group of variables forms a subset of configurations that are reachable by the new optimization process. However, note that even the "suboptimal" choice of $\vec{x}'$ in Figure 1a might be beneficial to the overall search, as it allows the optimization process to advance closer to certain parts of the Pareto-Set.

In a similar way, this accessible subspace is influenced by the grouping scheme and transformation function. The grouping of the variables has, similar to the location of $\vec{x}'$, an influence on the subspace of $\mathbb{R}^n$ in which the search will take place. The transformation function on the other hand mainly describes how the movement takes place inside this space, i.e. in which way the locations of the newly created solutions change. Additionally, it can affect the shape of the lower-dimensional space (i.e. a line like in Figure 1). The transformation of the problem, i.e. the definition of the searchable subspace that is accessible from a fixed solution, could also be seen as a kind of "neighbourhood" of the original solution $\vec{x}'$, in the sense that only certain directions of change in a solution are possible starting from $\vec{x}'$. Consequently, different transformation functions (other than in Equation 4) produce different "neighbourhoods". However, in contrast to neighbourhoods in local search methods, the solutions created in these "neighbourhoods" in WOF might not necessarily have a locality around the original solution. In fact, the opposite might be the case, as the steps made in the decision space might be very large depending on the weight values.

Although the explained problem transformation has disadvantages in preserving diversity and therefore in the ability to approximate the whole Pareto-optimal set, it must be mentioned that it can, on the other hand, have positive effects on the convergence speed, if a good choice of $\vec{x}'$ is made and appropriate grouping schemes and transformation functions are used. Different grouping schemes and transformation function will be explained in Section V. Here we want to point out that once good choices for these parameters are found, the new smaller search space can be searched much faster and more thoroughly.

To examine the grouping and transformation process and its implications in further detail, in Figures 2 - 4 we show the effect of this approach exemplary on the three test problems WFG2, UF2 (from the CEC2009 competition) and ZDT1 with $n = 1000$ decision variables. All plots show 2000 solutions created from a fixed randomly created solution by different methods. Figures 2 - 4 (a) show the solutions obtained by a standard SBX-crossover and polynomial mutation. For each of the 2000 solutions shown in (a), another solution was randomly created and used for crossover. The mutation and crossover parameters were the same as reported in Subsection VI-A. Using the same solution as in (a), instead of applying crossover and mutation, the proposed problem transformation was carried out with different grouping mechanisms in Figures 2 - 4 (b) - (d) (for details on these mechanisms see Subsection V-A). For each of these transformed problems, 2000 random solutions, i.e. weight vectors were created.



(a) Crossover and Mutation (b) Random Grouping

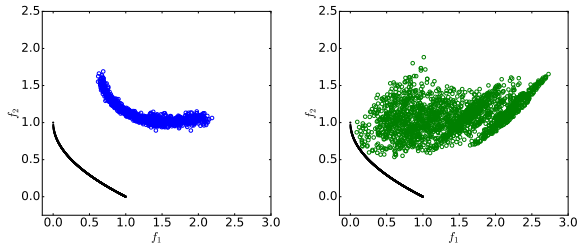
(c) Ordered Grouping (d) Differential Grouping

Fig. 2. Pareto-front (small black dots) and created solutions for the WFG2 problem by classic EA methods and different WOF groupings.

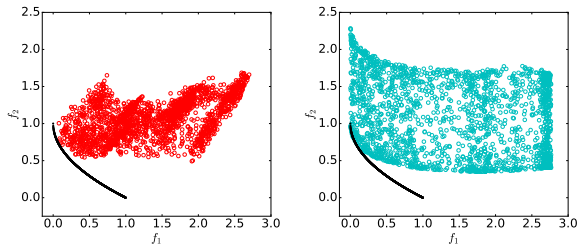
We can see in these figures that for all three problems the solutions obtained by the problem transformation differ to a great extent from the ones accessible by traditional evolutionary operators. In case of the WFG2 problem, the reachable solutions offer by far more diversity as well as closeness to the Pareto-front. If the transformed problem is optimized to obtain the best of the solutions seen in Figures 2 (b) - (d), an immense speed up of the search can be expected.

Almost the same situation occurs when transforming the UF2 problem in Figure 3. The effects of the grouping mechanisms differ slightly, however all these transformed problems

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2704782, IEEE Transactions on Evolutionary Computation

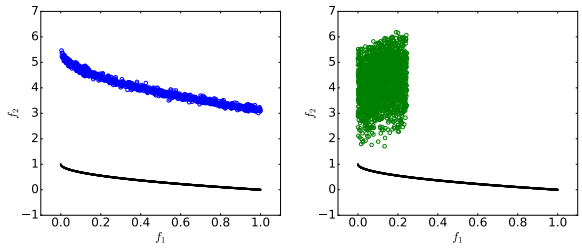


(a) Crossover and Mutation (b) Random Grouping

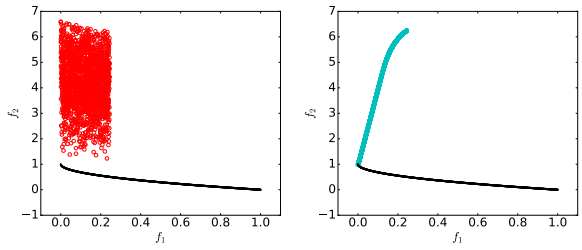


(c) Ordered Grouping (d) Differential Grouping

Fig. 3. Pareto-front (small black dots) and created solutions for the CEC2009 UF2 problem by classic EA methods and different WOF groupings.



(a) Crossover and Mutation (b) Random Grouping



(c) Ordered Grouping (d) Differential Grouping

Fig. 4. Pareto-front (small black dots) and created solutions for the ZDT1 problem by classic EA methods and different WOF groupings.

offer the possibility to reach a set as diverse as the one reached by classical operators. Moreover, the solutions obtainable lie very close to the Pareto-front.

The situation in Figure 4 when transforming the ZDT1 problem differs significantly from the ones in Figures 2 and 3. In Figure 4 we see the effect of loosing diversity that was described previously. While the classical crossover and mutation operators do offer a great extend of diversity, mostly because these solutions are based on crossover with other solutions, the transformed random solutions are limited to a part of the objective space as seen in Figures 4 (b) - (d). The area to which the search is limited is defined by the chosen $\vec{x}'$ in this case, which happened to be on the "left side" of the Pareto-front. Therefore, optimizing only the transformed problem might lead to fast convergence in Figure 4 (d), but might suffer from a very low diversity.

B. The Weighted Optimization Framework

In this section we describe our proposed approach in detail. The Weighted Optimization Framework (WOF) is designed as a generic meta-heuristic, which can be incorporated in an arbitrary population-based optimization mechanism. The choice of this mechanism depends on the preferences of the user or the properties of the problem. This optimization algorithm will be used for optimizing two different problems Z and $Z_{\vec{x}'}$ in turns. It is also possible to use two different optimization algorithms for the two problems. This might be advisable as the two problems differ largely in size. For instance, the smaller transformed problem could be thoroughly searched with a conventional method, while the original, large-scale problem can be searched with a CC-based method. The current work however concentrates on employing only one optimizer for both problems. The use of two different methods for the two optimization phases is left for future work.

As we have seen in the previous subsection, the transformed problem $Z_{\vec{x}'}$ has the drawback of limiting the search space and the advantage of searching a smaller space more thoroughly, which might result in a faster convergence but less diversity. On the other hand, the original problem Z can reach all possible solutions but might incorporate very slow convergence in a large-dimensional space. To utilize the synergy of these two formulations, WOF alternates two different phases of optimization: A *normal* optimization step and a *weighting* optimization step. The outline of our proposed method is shown in Figure 5 and more detailed steps are given as pseudocode in Algorithms 1, 2 and 3.

The WOF algorithm receives an optimization problem Z, a population-based optimization algorithm A, a grouping mechanism G (see Subsection V-A) as well as a transformation function ψ (see Subsection V-B) as its input. The starting point is an initial random population for the problem Z. In every iteration of the main loop (i.e. Lines $3-10$ in Algorithm 1), first the normal optimization step is carried out, where the original problem is optimized for a fixed number t_1 of function evaluations (Line 4 in Algorithm 1). The only special requirement here is that the algorithm needs to start with a given population instead of creating a new random one.

Next, in the weighting optimization step (Lines 6 to 8 in Algorithm 1) the transformation and optimization of weights are done q times to preserve diversity in the population (details below in Subsection IV-C). For this, q different solutions $\vec{x}'_k$ ($k = 1, 2, ..., q$) are drawn from the current population (Line 5). For every one of them, the weighting optimization is carried out as shown in Algorithm 2. The problem is first transformed as described above, resulting in different transformed problems $Z^k_{\vec{x}'}$. At this point, a transformation function has to be applied, and as mentioned earlier, different functions can be used. It is also possible to choose different transformation functions for each of the q transformed problems. The optimization algorithm is then used to optimize a population W_k of randomly created solutions (weights) for each of these $Z^k_{\vec{x}'}$. The

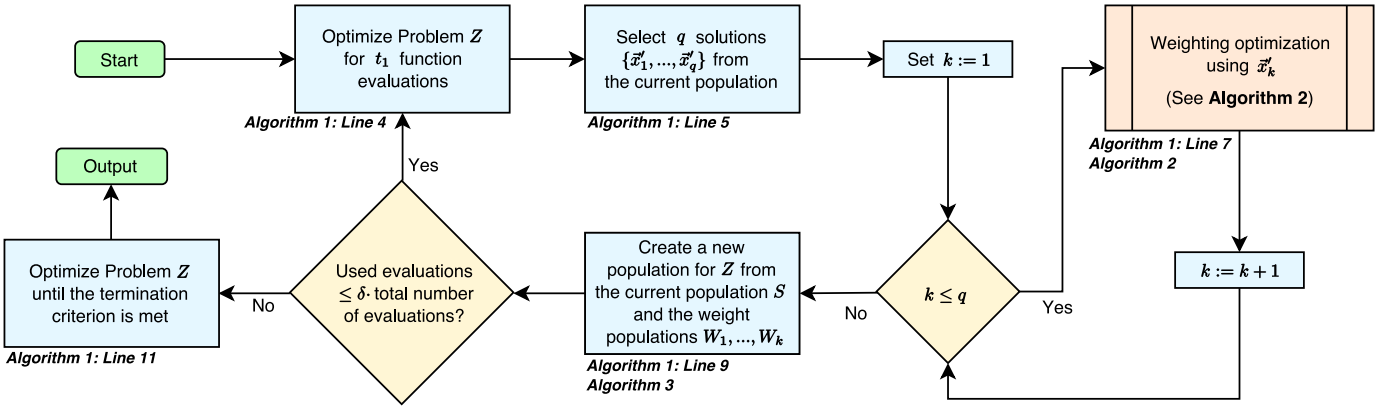


Fig. 5. Outline of the Weighted Optimization Framework.

Algorithm 1 WOF(Z,A,G,ψ)

Input: Problem Z, Optimization Algorithm A, Grouping Mechanism G, Transformation Function ψ
Output: Solution population S

1: Initialization
2: $S \leftarrow$ Random initial population for Z
3: **repeat**
4: $S \leftarrow A(Z, S, t_1)$ // Optimize Z with Algorithm A for t_1 evaluations, using S as a starting population.
5: $\{\vec{x}'_1, .., \vec{x}'_q\} \leftarrow$ Selection of q solutions based on Crowding Distance from the first non-dominated front of S
6: **for** $k = 1$ **to** q **do**
7: $W_k \leftarrow$ WeightingOptimization($\vec{x}'_k, Z, A, G, \psi$) // **Algorithm 2**
8: **end for**
9: $S \leftarrow$ updatePopulation($W_1, .., W_q, S$) // **Algorithm 3**
10: **until** $\delta \cdot total\#Evaluations$ used
11: $S \leftarrow A(Z, S)$ // Optimize Z with Algorithm A, using S as starting population, until all evaluations are used.
12: **return** FirstNonDominatedFront(S)

Algorithm 2 WeightingOptimization($\vec{x}'_k, Z, A, G, \psi$)

Input: Solution $\vec{x}'_k$, Problem Z, Optimization Algorithm A, Grouping Mechanism G, Transformation Function ψ
Output: Population of weights W_k

1: Initialization
2: Divide n variables into γ groups // See **Subsection V-A**
3: $Z^k_{\vec{x}'} \leftarrow$ Build a transformed problem with γ decision variables (weights) from Z, $\vec{x}'_k$, G and ψ
4: $W_k \leftarrow$ Random population of weights for $Z^k_{\vec{x}'}$
5: $W_k \leftarrow A(Z^k_{\vec{x}'}, W_k, t_2)$ // Optimize $Z^k_{\vec{x}'}$ with Algorithm A for t_2 evaluations, using W_k as a starting population.
6: **return** W_k

Algorithm 3 UpdatePopulation($W_1, ..., W_q, S$)

Input: Weight populations $W_1, ..., W_q$, Solution population S
Output: Solution population S

1: **for** $k = 1$ **to** q **do**
2: $w_k \leftarrow$ Select one individual from W_k
3: $S'_k \leftarrow$ Apply w_k to population S
4: **end for**
5: $S \leftarrow$ Perform non-dominated sorting on $S \cup \{S'_k\}_{k=1,..,q}$
6: **return** S

result of this low-dimensional optimization step is a population W_k of weights that optimizes the objective function values based on the values of the originally chosen solution $\vec{x}'_k$. This whole process of problem transformation and optimization is carried out q times, giving us a set of q weight populations $\{W_1, ..., W_q\}$.

These obtained weights are then assigned to the original solution population (Line 9 in Algorithm 1) in the following way, as also shown in detail in Algorithm 3: In order to save computational resources, from every W_k, we select one member (with the largest Crowding Distance from the first non-dominated front) and apply its values to every solution in S. By doing so, we obtain q sets of new solutions S'_k ($k = 1, ..., q$). To update the original population, we combine the obtained new solution sets with S and perform a non-dominated sorting mechanism as used in NSGA-II on the set $S \cup \{S'_k\}_{k=1,...,q}$. Duplicate solutions found in this union set are removed prior to the non-dominated sorting, as they would deteriorate diversity. The non-dominated sorting procedure in this step also ensures that worse solutions found in the current iteration (for example due to a suboptimal choice of one of the $\vec{x}'_k$) will not be further regarded in the upcoming steps. After that, the main loop starts again and we return to a normal optimization step to alter the variable values independently of each other.

In addition to applying one individual of each W_k to the population S, it is also possible to add all members of each W_k to the population update process. These solutions have already been evaluated in the optimization of $Z^k_{\vec{x}'}$ and can therefore be included in the process. Depending on the population size used when optimizing $Z^k_{\vec{x}'}$, this might improve the results slightly in favor of a slightly longer computation time. However, previous experiments showed, that by doing so the improvement in performance is, in most cases, negligible.

The alternation of the main loop of WOF (Lines 3 - 10 in Algorithm 1) is only repeated until the first termination condition is met. More precisely, the optimization of variables and weights will only take place during the first

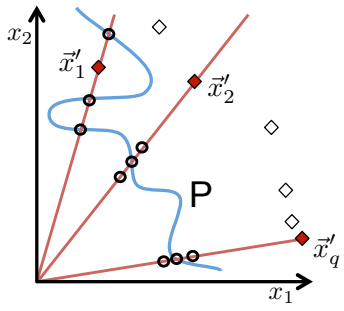


Fig. 6. $\boldsymbol{q}$ optimization steps in WOF. Diamonds: Original population. Circles: Achieved solutions of the $\boldsymbol{q}$ transformed problems.

$\delta \cdot total\#Evaluations$, where $\delta \in [0, 1]$ defines how many of the total function evaluations are spent for this phase. After that, the algorithm will use the normal optimization without weights for the second half of the optimization process (Line 11 in Algorithm 1).

The reason for this alternation is that the transformation of the problem might result in a directed search where only a certain area of the Pareto-front can be reached. Early experiments showed that using the weight optimization throughout the whole search process results in an overall loss of diversity. In general, the efficiency of WOF is most probably higher in the initial phase of an optimization process than in the final phase. Towards the end of the search, the Pareto-front might already be approximated to a certain extent. When WOF is used in this situation, the grouped alternation might deteriorate optimal values of one variable in favor of another. In this phase of the search, classical (ungrouped) evolutionary operators can make finer adjustments to the decision variable values independently of each other and therefore slowly approach the optimal values for each single variable.

C. The Parameter q and Choice of $\vec{x}'$

Now we take a look at how to choose the $\vec{x}'$ for the weighted optimization. In the case of just one objective function the selection is rather straightforward, as we can determine the *best* and *worst* solutions. Unfortunately we do not have a total order on the solutions in multi-objective optimization, so choosing one $\vec{x}'$ might result in a great acceleration of the optimization process towards (a part of) the Pareto-front, but in exchange, we might loose the diversity of the population (see Figure 4 (d)). To deal with this problem, we perform multiple independent weighting optimizations using different $\vec{x}'_k$, $k = 1, ..., q$. It might be advisable to specify q as m or larger (where m is the number of objectives), and decide to make the selection of $\vec{x}'$ based on a diversity indicator. In this way we can choose multiple solutions that lie "far" from each other in the objective space, to hopefully cover different areas of the decision space with the weighting optimization as shown in Figure 6. In this work, we select the best q solutions by applying the Crowding Distance metric to the first non-dominated front of the current population of the original problem. It is also possible to use other selection criteria in this step. In our case, we have obtained better results using Crowding Distance compared to a random selection (refer to Tables 17 - 19 and Figures 42 - 45 in the supplementary file). For problems with many objectives it may be promising to use reference-line based approaches.

V. Grouping and Transformation Functions

In this section we will describe four different mechanisms for grouping variables and three transformation functions that will be used later in the experiments.

A. Grouping Mechanisms

Grouping strategies usually aim to put those variables into the same group that interact strongly with each other (in non-separable optimization problems). Here we briefly explain different grouping methods that are used in this work to examine the influence of a good grouping strategy. While the first three methods are rather simple and do not use any information about the objective functions, the DG incorporates an intelligent mechanism based on problem analysis. We use this method for comparison, although it has been developed for single-objective optimization.

a) Random Grouping: Random grouping forms a fixed number γ of equal-sized groups and assigns each variable randomly to each of these groups.

b) Linear Grouping: Linear grouping assigns all n variables to a fixed number γ of groups in natural order. This means, the first $\frac{n}{\gamma}$ variables of the optimization problem are assigned to the first group and so on. This corresponds to the situation seen in Equation 4.

c) Ordered Grouping: The ordered grouping mechanism ranks the decision variables of the selected solution by their absolute values. This means that all variables are sorted based on their current absolute values and the $\frac{n}{\gamma}$ variables with the smallest values are assigned to the first group, the next $\frac{n}{\gamma}$ to the second group and so forth.

d) Differential Grouping: DG was developed in 2014 by Omidvar et al. [23] and used in single-objective optimization with CC. It aims to detect variable interaction prior to the optimization of the problem. The number of groups as well as their sizes are set automatically by the DG algorithm. In short, DG compares the amount of change in the objective function in reaction to a change in a variable x_i before and after another variable x_h is changed. DG answers the following question: *When changing the value of x_i, does the amount of change in $f(\vec{x})$ remains the same regardless of the value of another variable x_h?* If this is true, the variables x_i and x_h seem not to interact with each other, so they can be separated into different groups. Else, they seem to interact, so they are assigned to the same group. Two drawbacks must be mentioned when using this approach here: (1) The DG algorithm can become computationally expensive as analyzed in [23]. Assuming the problem contains a number $\gamma = \frac{n}{l}$ 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 $n = 2000$ decision variables, DG requires $n^2 = 4,000,000$ function evaluations to perform the grouping. Given that our experiments in Section VI only use $100,000$ function evaluations for the whole optimization process, this

is little practicable. (2) As mentioned, the DG algorithm was developed for single-objective problems. It was not designed to take multiple objective functions into account and, as a result, is not directly applicable to multi-objective problems. To make it applicable to multi-objective optimization, in this work only one of the objective functions (the first one) is considered in the DG algorithm. The implications and possibly different outcomes when using other than the first objective function might be studied in a future work.

The four grouping mechanisms differ in the required amount of computational effort in each generation. The linear grouping does not change during the process and therefore does not require recomputation in each generation. The same holds for DG, which is precomputed before the optimization starts and not changed during the search. On the other hand, the random grouping and the ordered grouping mechanisms are updated every time a problem transformation is carried out. In the former, a new random assignment to the γ groups is computed, in the latter the variables of each chosen x'_k are ordered in each problem transformation step.

B. Transformation Functions

In the following we will describe three different transformation functions $\psi(\vec{w}, \vec{x}')$ and examine their advantages and disadvantages. We assume the variables were devided into γ groups $g_1, ..., g_\gamma$ prior to the transformation. Let $g(x_i) = g_j$ be the respective group x_i is assigned to (i.e. the set of variables in the same group as x_i) and w_j the respective weight used to transform the values of g_j.

a) ψ_1 - Product Transformation: In Subsection IV-A we used a basic function $\psi(\vec{w}, \vec{x}') := (w_1 x'_1, ..., w_\gamma x'_n)$. Broken down to a single element, this equation results in the simple expression $x_{i,new} = w_j \cdot x_{i,old}$ for each variable. Therefore the first transformation function is as follows:

$$\begin{aligned} &x_{i,new} = w_j \cdot x_{i,old} \\ &w_j \in [0, 2] \end{aligned} \tag{5}$$

A disadvantage of ψ_1 is that we do not expect it to deal well with variable domains that involve both positive and negative values (since a multiplication with positive weights can only obtain positive or negative new values respectively). Additionally, the progress that can be made by altering one weight variable w_j is determined by the absolute value of x_i. The genetic operators used to alter the w_j are naturally limited in the amount of change per iteration, while at the same time we also need to apply some sort of domain to the w_j during the optimization process. That means, if the variable x_i has a domain of $x_i \in [0, 10]$, and the variable w_j has a domain of $w_j \in [0, 2]$, an absolute value of $x_i = 0.1$ will give the optimization algorithm only a maximum chance to explore the original search space in the interval $[0, 0.2]$, while a value of $x_i = 8$ will result in a search that covers at least the whole domain of x_i. Since we do not know the optimal variable values beforehand, the search should not be dependent on absolute values. Therefore, we propose the following transformation function:

b) ψ_2 - p-Value Transformation: This method is intended to uncouple the relationship between absolute variable values and the reachable domain space of the original variable.

$$\begin{aligned} &x_{i,new} = x_{i,old} + p \cdot (x_{i,max} - x_{i,min}) \cdot (w_j - 1.0) \\ &w_j \in [0, 2] \\ &p \in [0, 1] \end{aligned} \tag{6}$$

Using this function, the value of x_i is always altered within a certain range (defined by p) around its original value, where $x_{i,min}$ and $x_{i,max}$ are the lower and upper bounds of the variable x_i. The values of the w_j are bound to the interval $[0, 2]$ to ensure that variable values during optimization are always positive. As can be seen in the first line of the equation, these values are translated into $[-1, 1]$, which describes a change of values in one or the other direction around the original value. The actual maximum and minimum amounts of change are set by the parameter p, so that the domain of the w_j does not have to be altered. For instance, setting $p = 0.2$, the value of x_i is altered by 20% of the width of the variable's domain, centered around the original value $x_{i,old}$.

c) ψ_3 - Interval-Intersection Transformation: This function follows an approach suggested in [18] for single-objective optimization. We include it in this work especially for the comparison on the ZDT functions. This is because the previous two functions need a kind of *repair-mechanism* in case the result of the transformation function exceeds a variables domain. In this case, both ψ_1 and ψ_2 set the value of the variable to the respective boundary. However, this might be of advantage for problems like the ZDT benchmarks, since their optimal values are extremal values and can be obtained by such repairing. To provide a fair comparison with other algorithms on the ZDT problems in Subsection VI-D, we use the following transformation function ψ_3:

$$\begin{aligned} &x_{i,new} = w_j \cdot x_{i,old} \\ &w_j \in [\min_{x_h \in g(x_i)} (x_{h,min}/x_h), \max_{x_h \in g(x_i)} (x_{h,max}/x_h)] \end{aligned} \tag{7}$$

where $x_{h,min}$ and $x_{h,max}$ are the lower and upper bounds of the variable x_h. The application of the weight value follows the same multiplication equation as in ψ_1, only the domains of the weights for each group are changed. For each variable in a group, the maximum (minimum) weight value that can be used without exceeding the domain borders is calculated. The upper- and lower-bounds of the variables w_i are set based on the highest (lowest) of these possible values over all variables in the respective group. This interval $[w_{j,min}, w_{j,max}]$ can be seen as the intersection interval of all possible minimum and maximum values defined by each variable in the group.

VI. Evaluation

The benchmark problems used in this work are WFG1-9 [4], DTLZ1-7 [6], ZDT1-4 and 6 [5], and the ten unconstrained problems of the CEC2009 competition on multi-objective optimization (denoted as UF1-10) [7]. With the exception of the CCGDE3 algorithm, we rely on the jMetal Framework for Java, version 4.5 [30] for the implementation of the algorithms. The source codes of the mentioned problems are included

in the framework. For the comparison with the CCGDE3 algorithm, we obtained the original code from the website of the authors [1]. The C++ implementations of the UF problems were taken from the original competitions resources. The original C++ implementation of the WFG group was used, and the DTLZ C++ implementation were taken from the C++ adaptation of jMetal.

The algorithms used in this evaluation are the following:

- SMPSO, which represents a state-of-the-art swarm optimizer.
- NSGA-II, which is a widely known multi-objective evolutionary algorithm.
- GDE3, which uses differential evolution.
- WOF-SMPSO, WOF-NSGA-II, WOF-GDE3, which are WOF-enhanced versions of the previous ones, i.e., the respective algorithm (SMPSO, NSGA-II, GDE3) is used as an optimizer within our proposed framework to optimize the weights and variables.
- CCGDE3 is used as another large-scale optimizer. Published in 2013, it is a recent algorithm designed specifically for multi-objective many-variable problems.

The performance indicator used in this paper is the hypervolume indicator [31]. It can be used to measure both convergence and diversity of an obtained solution set. When reporting hypervolume values in this work, relative values are used, i.e. the obtained hypervolume divided by the hypervolume of the true Pareto-front. The reference point for calculating the hypervolume in all instances is obtained from a sample of the true Pareto-front of each problem by multiplying the nadir-point of the sample by 2.0 in each dimension. For each algorithm and each tested problem, we perform 51 independent runs. All experiments are conducted with 2 and 3 objectives wherever possible, and use decision variables from 40 up to 5000 depending on the respective experiment. For testing the statistical significance of the differences between indicator values (which are not likely to follow a normal distribution) we use the Mann-Whitney U statistical test. We indicate a difference as significant for a value of $p < 0.01$.

A. General Parameter Settings

Many of the experiments in the following sections will use identical parameter settings. In the following sections, only settings that differ from those given here will be explained.

The maximum number of function evaluations is set to $100,000$ in all experiments. In all WOF algorithms, the number t_1 of evaluations for each optimization of the original problem Z is set to 1000, and the number of evaluations for the transformed problem t_2 is 500. The parameter q, the number of repetitions of the weight optimization (therefore also the number of chosen solutions $\vec{x}'$) is set to 3, and the number of groups (γ) is set to 4 in the WOF algorithms and the CCGDE3. The ordered grouping mechanism is used in all experiments and the transformation function is ψ_2 with $p = 0.2$. The parameter δ, which defines the evaluations spent for the first phase (Lines 3 to 10 in Algorithm 1) is set to 0.5. A dynamic version where the termination of the first phase is linked to the progress of the search might be possible too, however, this task is left for future work. The sensitivity of WOF to different values of δ is reported later in Subsection VI-G.

SMPSO and NSGA-II use polynomial mutation with a probability of $1/n$ (with n being the number of decision variables) and a distribution index of 20.0. The NSGA-II and its WOF-enhanced version use an SBX crossover with a crossover probability of 0.9 and a distribution index of 20.0. In GDE3, CCGDE3 and WOF-GDE3, the parameters F and CR are both set to 0.5.

The size of the populations as well as the archives is set to 100 for all algorithms. In WOF, the optimization of the weights takes place with a population size of 10 due to the reduced size of the search space. The CCGDE3 uses an internal population size of 40, but creates a larger population with up to 160 solutions when the algorithm returns. We wanted to keep the parameters of the CCGDE3 the same as in the original paper, so the same setting is used here, with the exception that only the best 100 solutions are returned to ensure fairness to the other methods. The number of generations per single group in CCGDE3 also follows the original publication and is set to 1.

B. Performance on Various Benchmarks

In this part we will compare the performance of all algorithms on the mentioned benchmarks. All test problem instances use $n = 1000$ decision variables in this experiment. For the WFG problems, they are split into 250 position-related and 750 distance-related variables.

The median relative hypervolume values and interquartile ranges (IQR) can be seen in Table I. Best performance is indicated by a bold font. An asterisk is used to indicate whether the result is significantly worse than the respective best algorithm's performance for each problem. For readability, values equal to 0.0 are displayed as dashes.

We see that the WOF algorithms perform on average very well. For almost all of the tested problems, at least one version of the WOF algorithms performs significantly better than all the non-WOF methods. In the following, some interesting observations will be pointed out for the different problem families.

1) WFG: In the WFG problems none of the classic algorithms nor CCGDE3 can obtain a best hypervolume value. Moreover, all of the results are significantly worse than the respective best algorithm. In Table I, the best results are obtained by WOF-SMPSO for almost all test problems except for WFG1 with 2 and 3 objectives and WFG5 with 2 objectives (for which no significant difference between WOF-SMPSO and WFG-GDE3 is obtained). Additionally we want to point out the good performance on both, separable and non-separable problems. Decomposition methods often face difficulties when the problems are non-separable, since the interaction between the variables asks for a suitable grouping of the variables. The problems WFG2, 3, 6, 8 and 9 are non-separable. Still we observe that WOF can achieve the same superior performance as on the separable problems, which

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

TABLE I
MEDIAN RELATIVE HYPERVOLUME VALUES OF VARIOUS BENCHMARK PROBLEMS FOR $n = 1000$ DECISION VARIABLES. BEST PERFORMANCE IS SHOWN IN BOLD FONT. AN ASTERISK INDICATES STATISTICAL SIGNIFICANCE COMPARED TO THE RESPECTIVE BEST METHOD.

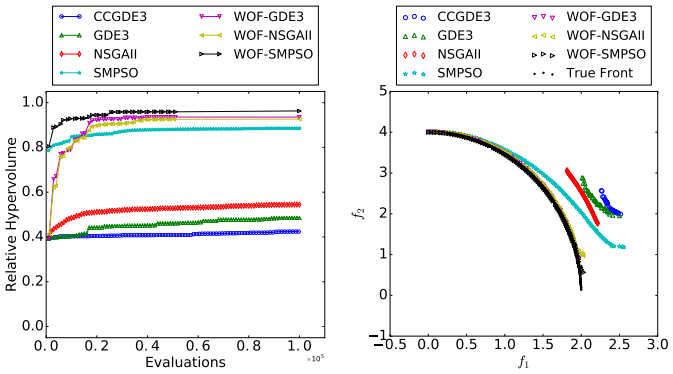
m		CCGDE3	GDE3	NSGAII	SMPSO	WOF-GDE3	WOF-NSGAII	WOF-SMPSO
2	DTLZ1	— (—) *	— (—) *	— (—) *	0.642949 (—)	— (—) *	— (—) *	0.642949 (—)
2	DTLZ2	— (—) *	— (—) *	— (—) *	**0.998909** (5.10E-5)	0.995145 (1.49E-3) *	0.997587 (4.00E-4) *	0.998694 (1.26E-4) *
2	DTLZ3	— (—) *	— (—) *	— (—) *	0.520072 (—)	— (—) *	— (—) *	0.520072 (—)
2	DTLZ4	— (—) *	— (—) *	— (—) *	**0.998900** (5.00E-5)	0.987029 (1.94E-2) *	0.996243 (7.80E-4) *	0.998700 (1.49E-4) *
2	DTLZ5	— (—) *	— (—) *	— (—) *	**0.999348** (5.83E-5)	0.995736 (2.90E-3) *	0.998003 (4.86E-4) *	0.999100 (1.36E-4) *
2	DTLZ6	— (—) *	— (—) *	— (—) *	— (—) *	0.999011 (1.45E-4) *	**0.999150** (6.17E-5)	0.999149 (3.75E-1)
2	DTLZ7	0.698002 (7.78E-3) *	0.380034 (1.40E-2) *	0.442085 (2.36E-2) *	0.070975 (5.03E-2) *	0.798233 (1.01E-5) *	0.798236 (1.51E-5) *	**0.798270** (6.21E-6)
2	WFG1	0.368023 (9.71E-3) *	0.575510 (8.54E-3) *	0.314626 (2.34E-3) *	0.590658 (7.06E-3) *	**0.652464** (1.66E-2)	0.646211 (4.40E-3) *	0.645470 (1.11E-2)
2	WFG2	0.619606 (5.85E-3) *	0.688903 (4.22E-3) *	0.667313 (9.52E-3) *	0.736841 (2.53E-2) *	0.862312 (3.16E-2) *	0.800874 (1.41E-2) *	**0.983370** (6.56E-3)
2	WFG3	0.543414 (6.03E-3) *	0.683368 (5.56E-3) *	0.612655 (7.43E-3) *	0.594265 (2.81E-2) *	0.807946 (8.23E-3) *	0.807269 (6.12E-3) *	**0.855820** (4.67E-3)
2	WFG4	0.507776 (6.72E-3) *	0.630822 (1.46E-2) *	0.621814 (9.71E-3) *	0.843217 (4.46E-3) *	0.870188 (2.07E-2) *	0.856543 (1.47E-2) *	**0.975442** (1.26E-2)
2	WFG5	0.541706 (7.00E-3) *	0.668053 (2.06E-2) *	0.587745 (1.11E-2) *	0.803472 (4.10E-3) *	0.940453 (7.84E-2)	0.878879 (9.23E-2) *	**0.946467** (2.30E-2)
2	WFG6	0.400922 (8.92E-3) *	0.495129 (1.14E-2) *	0.576546 (8.92E-3) *	0.969041 (4.28E-3) *	0.906795 (4.52E-2) *	0.834530 (5.97E-2) *	**0.998855** (2.55E-4)
2	WFG7	0.524232 (1.18E-2) *	0.696964 (1.52E-2) *	0.648534 (8.83E-3) *	0.697714 (1.82E-2) *	0.857582 (4.17E-3) *	0.862129 (5.15E-3) *	**0.962463** (1.35E-2)
2	WFG8	0.466575 (5.19E-3) *	0.667567 (6.59E-3) *	0.551133 (1.01E-2) *	0.515858 (7.58E-4) *	0.810636 (1.97E-2) *	0.815991 (2.46E-2) *	**0.885010** (1.91E-2)
2	WFG9	0.473511 (1.08E-2) *	0.511297 (9.30E-3) *	0.621036 (3.40E-2) *	0.894743 (9.31E-3) *	0.935509 (1.96E-2) *	0.931595 (2.27E-2) *	**0.962831** (1.47E-2)
2	UF1	0.768177 (2.06E-2) *	0.673944 (3.26E-2) *	0.709649 (6.09E-2) *	0.214618 (8.11E-3) *	0.834999 (6.02E-3) *	**0.901471** (5.07E-2)	0.784401 (6.12E-3) *
2	UF2	0.576268 (1.56E-2) *	0.673588 (9.42E-3) *	0.804848 (9.14E-3) *	0.870498 (1.07E-3) *	0.927186 (2.69E-3) *	0.911687 (1.94E-3) *	**0.932005** (1.41E-3)
2	UF3	0.517992 (4.94E-3) *	0.536167 (1.09E-2) *	0.675722 (7.55E-3) *	0.628483 (2.77E-3) *	0.982468 (1.48E-3) *	0.980885 (3.36E-3) *	**0.989193** (1.14E-3)
2	UF4	0.832407 (7.82E-3) *	0.904059 (4.44E-3) *	0.855195 (4.12E-3) *	0.890782 (1.98E-3) *	0.908135 (3.42E-2) *	0.906701 (2.29E-2) *	**0.915155** (1.53E-2)
2	UF5	— (—) *	— (—) *	— (—) *	— (—) *	— (—) *	**0.199880** (9.99E-2)	— (—) *
2	UF6	0.407692 (4.75E-2) *	0.076527 (4.21E-2) *	0.217237 (1.08E-1) *	— (—) *	0.503085 (2.10E-2) *	**0.701907** (6.42E-2)	0.256224 (3.93E-2) *
2	UF7	0.606616 (1.86E-2) *	0.709033 (1.38E-2) *	0.725638 (1.48E-1) *	0.190892 (1.17E-2) *	0.842589 (7.20E-3) *	**0.898994** (3.27E-2)	0.777471 (8.81E-3) *
2	ZDT1	0.854970 (8.69E-3) *	0.548507 (2.05E-2) *	0.568226 (4.09E-2) *	0.112146 (3.88E-2) *	0.998415 (1.55E-4) *	0.998462 (1.13E-4) *	**0.998609** (1.21E-4)
2	ZDT2	0.758925 (2.05E-2) *	0.109237 (4.02E-2) *	0.110764 (5.31E-2) *	— (—) *	0.998239 (1.50E-4) *	0.998270 (1.49E-4) *	**0.998476** (1.74E-4)
2	ZDT3	0.815271 (6.62E-2) *	0.512281 (1.82E-2) *	0.621983 (2.71E-2) *	0.217650 (3.62E-2) *	0.999277 (7.55E-5) *	0.999326 (5.44E-5) *	**0.999439** (5.62E-5)
2	ZDT4	— (—) *	— (—) *	— (—) *	**0.998789** (4.13E-3)	— (—) *	— (—) *	0.998615 (1.10E-4) *
2	ZDT6	— (—) *	— (—) *	— (—) *	— (—) *	0.998498 (1.69E-4) *	0.998639 (4.69E-5) *	**0.998766** (7.89E-5)
3	DTLZ1	— (—) *	— (—) *	— (—) *	**0.587705** (1.48E-1)	— (—) *	— (—) *	0.587705 (5.87E-1) *
3	DTLZ2	— (—) *	— (—) *	— (—) *	**0.981398** (2.04E-3)	0.781343 (6.49E-2) *	0.976170 (3.24E-3) *	0.979655 (3.22E-3) *
3	DTLZ3	— (—) *	— (—) *	— (—) *	**0.390379** (—)	— (—) *	— (—) *	— (3.90E-1) *
3	DTLZ4	— (—) *	— (—) *	— (—) *	**1.004301** (1.61E-3)	0.485969 (3.05E-1) *	0.976607 (1.14E-2) *	1.001749 (5.63E-3) *
3	DTLZ5	— (—) *	— (—) *	— (—) *	**0.998575** (1.93E-4)	0.004036 (6.93E-2) *	0.982175 (5.72E-3) *	0.998159 (4.52E-4) *
3	DTLZ6	— (—) *	— (—) *	— (—) *	— (—) *	0.999158 (2.53E-4) *	0.999316 (1.58E-4) *	**0.999504** (1.14E-4)
3	DTLZ7	0.279526 (2.50E-1) *	0.219555 (2.66E-2) *	0.477192 (2.50E-2) *	0.002625 (6.41E-3) *	**0.989964** (3.10E-3)	0.989889 (2.55E-3)	0.979407 (5.84E-3) *
3	WFG1	0.416198 (8.10E-3) *	0.570543 (3.66E-3) *	0.289553 (3.82E-3) *	0.579919 (4.74E-3) *	**0.604212** (1.12E-2)	0.590271 (5.22E-3) *	0.602670 (7.99E-3)
3	WFG2	0.552173 (5.18E-3) *	0.617007 (4.76E-3) *	0.655834 (5.87E-3) *	0.740452 (5.07E-3) *	0.868322 (1.10E-2) *	0.865291 (4.93E-2) *	**0.956430** (1.13E-2)
3	WFG3	0.243353 (1.84E-2) *	0.476209 (9.00E-3) *	0.502668 (1.84E-2) *	0.564899 (1.61E-2) *	0.765131 (3.90E-2) *	0.804832 (2.64E-2) *	**0.948554** (1.29E-2)
3	WFG4	0.357544 (7.68E-3) *	0.538518 (1.00E-2) *	0.509696 (1.05E-2) *	0.719935 (5.03E-3) *	0.806799 (2.02E-2) *	0.806814 (3.26E-2) *	**0.885714** (3.72E-2)
3	WFG5	0.390448 (7.58E-3) *	0.601179 (1.16E-2) *	0.460440 (8.12E-3) *	0.678253 (5.92E-3) *	0.796938 (4.47E-2) *	0.793484 (3.09E-2) *	**0.835976** (2.78E-2)
3	WFG6	0.316011 (1.03E-2) *	0.445367 (1.01E-2) *	0.451270 (1.29E-2) *	0.874272 (6.95E-3) *	0.845404 (1.03E-1) *	0.741775 (8.18E-2) *	**0.967580** (1.31E-2)
3	WFG7	0.387126 (7.60E-3) *	0.567569 (8.38E-3) *	0.552353 (6.13E-3) *	0.662307 (1.42E-2) *	0.796639 (1.45E-2) *	0.800744 (1.63E-2) *	**0.856828** (2.75E-2)
3	WFG8	0.327917 (6.15E-3) *	0.536088 (8.10E-3) *	0.453376 (7.62E-3) *	0.547306 (1.91E-2) *	0.739925 (2.35E-2) *	0.752790 (2.84E-2) *	**0.812080** (2.39E-2)
3	WFG9	0.287493 (5.26E-3) *	0.369398 (1.82E-2) *	0.492735 (1.32E-2) *	0.824473 (2.25E-2) *	0.869764 (3.33E-2) *	0.841607 (2.40E-2) *	**0.903643** (3.07E-2)
3	UF8	— (—) *	0.087400 (2.11E-2) *	0.296347 (5.62E-2) *	0.857926 (6.39E-4)	0.855352 (1.53E-3) *	0.857292 (8.50E-4) *	**0.858015** (4.85E-4)
3	UF9	0.002653 (8.61E-3) *	0.062455 (1.54E-2) *	0.353288 (5.55E-2) *	0.555565 (2.99E-3) *	**0.694577** (1.01E-2)	0.642928 (6.99E-2) *	0.641168 (7.83E-3) *
3	UF10	— (—) *	— (—) *	— (—) *	0.854390 (3.84E-3)	0.778501 (1.46E-1) *	0.690878 (1.10E-1) *	**0.855156** (2.40E-3)

lets us assume that separability doesn't affect the WOF's performance much. It must be noted that the used ordered grouping mechanism is likely to separate the position- and distance-related variables into different groups. This lets WOF optimize them separately and results in a good performance of the non-separable problems. However, further experiments using random grouping also showed a superior performance of the WOF algorithms on the separable and non-separable WFG problems.

2) ZDT: In Table I, the best performance on the ZDT problems is obtained by WOF-algorithms. Only in the ZDT4, where all but the PSO-based algorithms fail to obtain any hypervolume value, the original SMPSO performs a little better than the WOF-SMPSO. This difference however might not be of great importance, since both methods achieve a relative hypervolume of more than 99.8%.

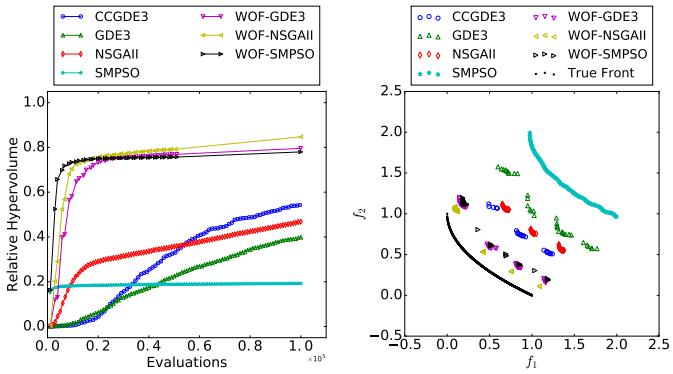
3) CEC 2009 Competition: The results on the CEC2009 competition basically follow the same pattern. The WOF-SMPSO and the WOF-NSGA-II perform best and in almost all instances significantly better than the non-WOF algorithms. The only two exceptions are the UF8 and UF10 problems where the original SMPSO is on a par with its WOF version. That is, SMPSO is not significantly outperformed by its WOF version for UF8 and UF10.

4) DTLZ: On the DTLZ problems, the WOF versions of GDE3 and NSGA-II perform significantly better than their respective original parts. Also, all WOF algorithms perform better than the CCGDE3 algorithm. For most instances of DTLZ1-5 in Table I, the best results are obtained by the SMPSO. On these problems, WOF does not further improve the performance of the SMPSO algorithm in the same way as it does of NSGA-II and GDE3, mainly due to a good performance of the original SMPSO in the first place. Although the results of the SMPSO are better than the WOF-SMPSO on these five problems, we also see that the results of WOF-SMPSO and SMPSO do not actually differ to a great extend, even though these differences are marked as statistically significant. The obtained relative hypervolumes range in the same order of magnitude, as both algorithms perform very similar to each other. This indicates that the



(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 7. Convergence and obtained solution sets on the WFG9 problem with $n = 2000$ variables.



(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 8. Convergence and obtained solution sets on the CEC2009 UF1 problem with $n = 2000$ variables.

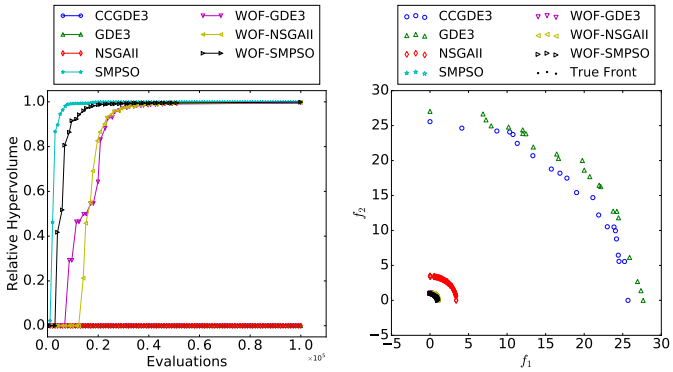
SMPSO already performs well (to a certain degree) on the large-scale DTLZ1-5 problems, thus the WOF version of it does not deliver the same amount of improvement as for the other algorithms. At the same time, we can observe that the performance of the NSGA-II and GDE3 methods are improved significantly, as they perform poor without WOF, but much better when WOF is used. For the DTLZ6 and DTLZ7 problems, we observe the same superior performance of all WOF algorithms as for the other problem families in Table I. We can conclude for the DTLZ problems that the WOF does improve the performance of previously bad-performing algorithms (NSGA-II, GDE3) in the same way as in the case of the UF and WFG benchmarks, and does on the other hand deliver comparable results to the original algorithm (SMPSO), if this one does already deal well with the large-scale problem.

A special consideration goes to the CCGDE3, since it is an algorithm that was specifically designed to work well on large-scale problems. In the experiments performed in this section, it could not perform of the same level with any of the WOF algorithms on any on the test problems. The comparison using the original settings of the CCGDE3 in Subsection VI-D will support this in further detail.

Concerning the diversity of solutions, Figures 7 - 12 and also Figures 14 - 41 of the supplement material illustrate that the obtained solutions are well-spread for several test problems. Nevertheless, the final populations obtained by our approach do not always cover the entire Pareto-fronts. On the other hand, we observe much better convergence performance compared to the other algorithms. This observation clearly shows the difficulty to find a balance between convergence and diversity for large-scale multi-objective problems. Our approach seems to be somewhat convergence-oriented. Diversity improvement is an important future research topic including the use of uniformly specified reference lines instead of the Crowding Distance for selecting solutions (see Subsection IV-C).

C. Scaling the Number of Variables

Table I gave some detailed results for $n = 1000$ variables. In addition to the analysis that we gave for each of the test problems until now, we examined how these results change



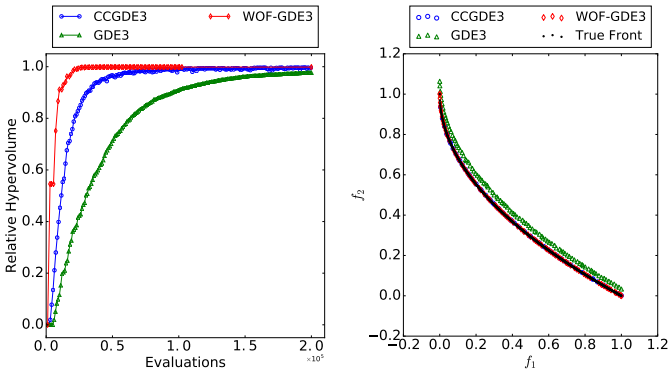
(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 9. Convergence and obtained solution sets on the DTLZ2 problem with $n = 1000$ variables.

with the dimension of the search space. For each setting of $n \in \{40, 80, 200, 600, 1000, 2000\}$ we perform the same experiment as in the previous section. Due to limited space, Tables 15 and 16 of the supplement material to this article give an overview over these experiments. In these tables, for every combination of problem and number of variables we list the algorithm that achieved the highest as well as second highest median hypervolume values. An asterisk is used to indicate whether the respective values are significantly better than the next-best performance.

As expected, for small numbers of variables, the original optimization methods SMPSO, NSGA-II and GDE3 work best on most problems, as seen in Tables 15 and 16 of the supplements in the experiments with $n = 40$ and $n = 80$. The exception is the WFG problem family, where the WOF versions already work better than the original algorithms for these relatively small problems instances, although the differences are not always statistically significant in these cases.

When the number of variables is increased, we observe a superior performance of the WOF-methods, especially the WOF-SMPSO. In the 2-objective experiments (Table 15 of the supplements), for $n = 600$, $n = 1000$ (which corresponds to the results in Table I) and $n = 2000$, the WOF algorithms are placed best for all WFG, ZDT and UF problems, with the



(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 10. Comparison with CCGDE3: Convergence and obtained solution sets on the ZDT1 problem with $n = 500$ variables.

exception of ZDT4.

In Figures 7, 8 and 9 we show the convergence rate and obtained solution sets on three problem instances. The runs shown are the ones that achieved the median hypervolume of each respective algorithm among all 51 runs. Figure 7 shows the WFG9 problem with 2000 variables. We observe not only a superior performance in terms of closeness to the Pareto-front as shown in the previous tables, but also a very fast convergence rate especially in the beginning of the optimization. The same can be seen in Figure 8 for the UF1 problem with 2000 variables. All three WOF algorithms obtain a much closer approximation of the Pareto-front. Additionally the WOF algorithms only need a very small share of the total function evaluation to reach this approximation. Finally in Figure 9 we observe the DTLZ2 performance for $n = 1000$ variables. As seen before, the WOF does not perform in the same superior way when used with the SMPSO algorithm on the DTLZ problems. However, the convergence as seen in Figure 9 (a) shows that all 3 WOF algorithms do still outperform all but the SMPSO algorithm in terms of solution quality as well as convergence rate.

In conclusion, we observe that for any large-scale optimization problem, the WOF methods can outperform the traditional GDE3 and NSGA-II methods as well as the large-scale method CCGDE3 significantly. The performance of the SMPSO lies on par with the WOF-SMPSO for some of the DTLZ problems, but the WOF method also increases its performance significantly on the other benchmark families. Overall, the results of the WOF do not only show a superior solution quality, but also a very fast convergence rate.

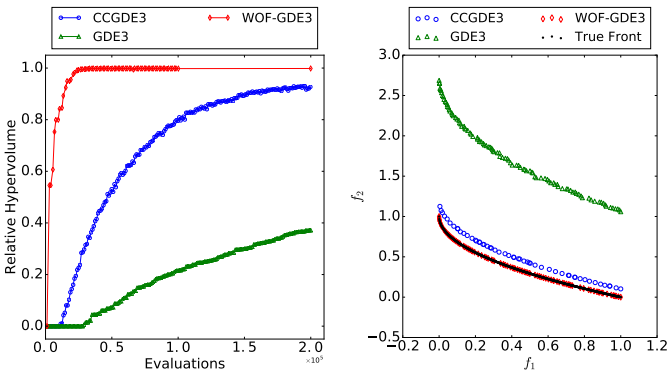
D. Comparison with CCGDE3

As described above, CC has been used frequently for large-scale optimization, especially for single-objective problems. A few works also used CC for multi-objective optimization and performed experiments using the ZDT [5] suite. One of the most recent advances is the CCGDE3 algorithm presented in [11], where the CC concept is applied to the GDE3 algorithm.

In the previous subsection, we observe in Table 15 of the supplements that for only one small-scale problem instance of the UF5 problem with $n = 80$ variables and 2 objectives

TABLE II
MEDIAN RELATIVE HYPERVOLUME VALUES AND IQRS OF THE ZDT PROBLEMS WITH DIFFERENT NUMBERS OF VARIABLES. SETTINGS USED ARE THE SAME AS IN [11]. BEST PERFORMANCE IS SHOWN IN BOLD FONT. AN ASTERISK INDICATES STATISTICAL SIGNIFICANCE COMPARED TO THE RESPECTIVE BEST METHOD.

	CCGDE3	GDE3	WOF-GDE3
n = 200			
ZDT1	0.996298 (3.91E-3) *	0.998319 (6.82E-5)	**0.998346** (2.13E-4)
ZDT2	0.983671 (9.13E-3) *	0.997598 (1.88E-4) *	**0.998198** (3.25E-4)
ZDT3	0.958335 (6.09E-2) *	0.999220 (3.31E-5)	**0.999239** (1.18E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	0.982064 (8.55E-3) *	0.858491 (6.72E-3) *	**0.998698** (2.58E-4)
n = 500			
ZDT1	0.995791 (3.54E-3) *	0.976890 (1.99E-3) *	**0.998392** (1.53E-4)
ZDT2	0.985697 (1.01E-2) *	0.960381 (2.26E-3) *	**0.998217** (2.03E-4)
ZDT3	0.929806 (6.71E-2) *	0.951782 (4.13E-3) *	**0.999252** (1.04E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	0.862895 (1.13E-2) *	— (—) *	**0.998573** (3.30E-4)
n = 1000			
ZDT1	0.984426 (4.46E-3) *	0.783685 (1.90E-2) *	**0.998382** (2.19E-4)
ZDT2	0.966051 (1.46E-2) *	0.549738 (3.50E-2) *	**0.998204** (3.98E-1)
ZDT3	0.922063 (6.41E-2) *	0.696589 (1.63E-2) *	**0.999267** (1.54E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	0.453370 (5.22E-2) *	— (—) *	**0.998545** (2.91E-4)
n = 2000			
ZDT1	0.932251 (6.91E-3) *	0.371685 (2.40E-2) *	**0.998410** (2.05E-4)
ZDT2	0.892224 (1.77E-2) *	— (—) *	**0.998161** (3.98E-1)
ZDT3	0.761271 (2.04E-1) *	0.384368 (1.15E-2) *	**0.999264** (8.57E-5)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	— (—) *	— (—) *	**0.998367** (3.86E-4)
n = 3000			
ZDT1	0.838997 (9.71E-3) *	0.193257 (2.31E-2) *	**0.998411** (1.58E-4)
ZDT2	0.774619 (3.05E-2) *	— (—) *	**0.998220** (3.98E-1)
ZDT3	0.575661 (1.58E-1) *	0.262717 (1.33E-2) *	**0.999242** (1.31E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	— (—) *	— (—) *	**0.998292** (6.89E-4)
n = 4000			
ZDT1	0.720947 (1.54E-2) *	0.104674 (1.66E-2) *	**0.998407** (2.20E-4)
ZDT2	0.633076 (2.28E-2)	— (—) *	**0.998246** (3.98E-1)
ZDT3	0.472919 (1.52E-1) *	0.199042 (9.78E-3) *	**0.999227** (1.16E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	— (—) *	— (—) *	**0.998138** (6.87E-4)
n = 5000			
ZDT1	0.614349 (2.17E-2) *	0.047964 (1.06E-2) *	**0.998428** (1.44E-4)
ZDT2	0.486880 (2.45E-2) *	— (—) *	**0.998159** (3.98E-1)
ZDT3	0.384726 (1.61E-1) *	0.160194 (8.56E-3) *	**0.999234** (1.18E-4)
ZDT4	— (—) *	— (—) *	**0.545522** (—)
ZDT6	— (—) *	— (—) *	**0.998120** (1.00E-3)

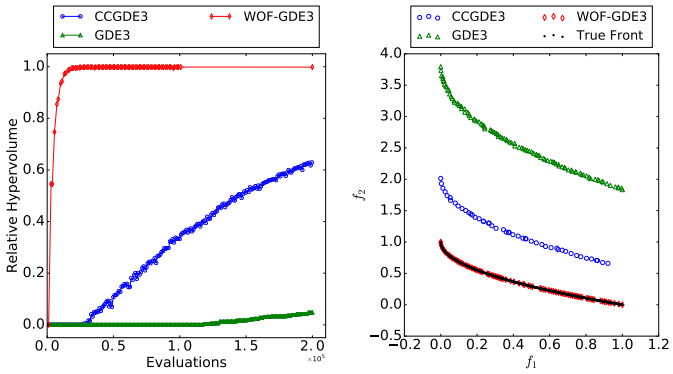


(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 11. Comparison with CCGDE3: Convergence and obtained solution sets on the ZDT1 problem with $n = 2000$ variables.

CCGDE3 could win against the other methods. However, except for this 80-variable UF5 problem, CCGDE3 is not performing best for any test problem in Table I and Tables 15 and 16 of the supplement material.

In this section we will further compare the performance of the WOF with CCGDE3. As experimental benchmark, the



(a) Convergence of rel. hypervolume (b) Solution sets and Pareto-front

Fig. 12. Comparison with CCGDE3: Convergence and obtained solution sets on the ZDT1 problem with $n = 5000$ variables.

ZDT problems 1 to 4 and 6 were used with the numbers of decision variables ranging from 200 to 5000. To compare the WOF with the CCGDE3 method, we perform all experiments using the same configurations and parameter settings as in the original work [11]. We compare the performance of the original GDE3 algorithm, CCGDE3, and WOF-GDE3 using the same test problems (ZDT1-3 and 6) that were used in [11], as well as the ZDT4 problem.

The parameter settings differ from Subsection VI-A in the following: The number of groups γ is set to 2 in CC and WOF. The grouping mechanism used for both is a random grouping of the variables. Furthermore, since the optimal solutions of the ZDT problems lie on the boundaries of the variables domains, for fair comparison we use the transformation function ψ_3 as described in Subsection V-B. In this way, the need for a repair mechanism is almost eliminated. When infeasible values are created using the obtained weights on other solutions in the population, the respective variable values are set to random values instead of the extremal ones.

The results of this experiment can be seen in Table II. Even for rather small problem instances ($n = 200$) the WOF-GDE3 can outperform the two other methods. Only for the ZDT1 and ZDT2 problems, the original GDE3 algorithm does not perform significantly worse.

With increasing the number of variables we see that the original GDE3 cannot find a good approximation of the Pareto-front for any problem. The CCGDE3 starts to outperform the original GDE3 when the number of variables grows larger. However, for all problem instances with $n \geq 500$, the WOF algorithm clearly obtains the highest median hypervolume in all test problems with statistical significance (with the one exception of ZDT2 with 4000 variables). This shows that for a large number of decision variables the WOF algorithm is clearly superior to the CC framework on the ZDT benchmarks. This adds to the results seen in Subsections VI-B and VI-C, where we saw that the superiority of WOF is already present for much smaller values of n when other problems than the ZDT are used.

Another aspect that is notable is once again the convergence rate of the WOF algorithm. The difference in performance for the ZDT1 problem is shown in Figures 10 - 12. We plotted

TABLE III
COMPARISON OF GROUPING SCHEMES AND TRANSFORMATION FUNCTIONS. SHOWN ARE THE NUMBERS OF TIMES THE RESPECTIVE COMBINATION ACHIEVES THE BEST / SECOND BEST / WORST MEDIAN REL. HYPERVOLUME OVER ALL TESTED PROBLEMS.

		Transformation function		
		ψ_1	ψ_2	ψ_3
Grouping	Random	1 / 1 / 8	3 / 1 / 2	1 / 1 / 7
	Ordered	4 / 4 / 3	3 / 6 / 0	2 / 4 / 4
	Linear	4 / 2 / 2	2 / 2 / 2	2 / 4 / 3
	Differential	4 / 1 / 3	7 / 0 / 1	2 / 5 / 1

for each algorithm the respective run that achieved the median hypervolume of all 51 runs for $n = 500$, $n = 2000$ and $n = 5000$ decision variables. Additionally, the convergence rate is plotted next to it. In these figures one can clearly see that the WOF algorithm was able to obtain an almost perfect approximation of the Pareto-front with a good diversity in all cases. In contrast, the original GDE3 and the CCGDE3 are not able to obtain any good approximation of the Pareto-optimal solutions for $n > 500$. In all cases, we also see that the convergence rate is extremely high for the WOF-GDE3. In all three plots, it reaches a relative hypervolume of over 99% already after less than 12% of the total function evaluations, no matter how many variables are used.

A special consideration goes to the ZDT4 problem. It is the only tested problem that allows negative variable values, which limits the reachable search space of the WOF algorithm. Since we used the ψ_3 transformation function, as explained before the weights can only map positive to positive and negative to negative variable values. As a result, a part of the search space might not be reachable in the weight optimization steps of WOF. Nevertheless, the performance of the WOF on the ZDT4 problem is significantly higher that that of the CCGDE3 for any tested number of variables. While the CCGDE3 does not obtain any hypervolume value with the used reference point, the WOF always obtains the same hypervolume value (in all runs, therefore the IQR is non-existent). A closer look in the results showed that in fact the WOF-GDE3 always finds exactly one (same) solution that is part of the true Pareto-front in every run. Therefore, its performance can be regarded as extremely reliable for ZDT4, while in contrast no diversity exists. This however, is not a direct weakness of the WOF-method, because we saw that the WOF-SMPSO performed well on the ZDT4 problem before (see Table I).

E. Comparison with MOEA/DVA

In this part, we compare the performance of our WOF method with MOEA/DVA [12][2]. This approach focusses on finding optimal groupings of the variables by dividing them into diversity-related and convergence-related variables. In comparison with our proposed method, some interesting results can be observed. The tables and figures supporting these can be found in the supplement material to this article. We compared the performance of the best found WOF-version from the previous experiments, the WOF-SMPSO,

[2]Codes are obtained from the original authors' webpage: http://web.xidian.edu.cn/home/fliu/lunwen.html

with MOEA/DVA on the UF1 - UF10 as well as the 2-objective WFG1 - 5 and WFG7 and the 3-objective WFG1-5 problems using 1000 decision variables in all instances.

The experiments in this subsection differ from the previous ones in only one aspect which is the number of function evaluations used. The MOEA/DVA uses the majority of its total functions evaluations in the preliminary variable grouping method, before the actual optimization starts. In our experiments, the grouping mechanism of MOEA/DVA for 1000-variable problems consumed around $9,000,000$ evaluations. Given that all previous experiments with WOF in this article only use 100,000 evaluations in total, the comparison might not be fair in this case. For this reason we decided to provide all algorithms a total of 10,000,000 evaluations for the experiments in this section. To compare, the original article of MOEA/DVA [12] used between 1,200,000 and 3,000,000 evaluations in their experiments with the 200-variable problems. The parameters of the MOEA/DVA were the same as found in the original MOEA/DVA paper with the exception of adjusting the population sizes to the same values as the ones of SMPSO and WOF-SMPSO to ensure a fair comparison using the hypervolume indicator. Furthermore, the distribution indexes of crossover and mutation are set to the same values as described in Subsection VI-A. No other changes were made to the original sourcecode provided by the authors of [12].

The obtained median hypervolume and IQR values are shown in Tables 13 and 14 in the supplement materials of this article, and the solution sets and convergence plots for the 2-objective problems can be found in Figures 1 to 13 in the supplement material. The results show that on the (2-objective) UF1-7 and the UF9 problems, the MOEA/DVA performs better than the WOF-SMPSO. On the other hand, the WOF-SMPSO significantly outperforms MOEA/DVA on all tested WFG problem instances with 2 and 3 objectives and the UF10 problem. Partly this difference results from a slightly worse performance of the WOF algorithm on the UF problems, however we also observed a large decrease in the performance of MOEA/DVA on the WFG problems in comparison to UF.

A second observation regards the convergence rate. In contrast to the MOEA/DVA, which uses 90% of the total evaluations for an initial grouping phase, the WOF-SMPSO does immediately start with the optimization process and can therefore deliver solution sets of reasonable quality after just a fraction of the total evaluations. In all used WFG2, 4, 5, and 7 as well as UF3 instances the WOF-SMPSO provides a hypervolume value of over 0.95 after just 10% (1,000,000) of the total function evaluations, and in the WFG1 and 3 it obtains a hypervolume very close to the final value after this amount. For all other problems we observe a more gradual convergence, however, the advantage remains that even with a very small amount of evaluations, we can already obtain results of certain quality using the WOF-SMPSO (as also seen in Subsection VI-B where we used only 100,000 evaluations). In contrast, while the MOEA/DVA outperforms WOF on most of the UF problems, these results can only be obtained after a large computational effort, since no solution sets are produced by MOEA/DVA during its grouping phase. As a conclusion, there are some problems where the MOEA/DVA outperforms WOF-SMPSO, but with the disadvantage of using a much larger computational budget, while on some other problems the WOF-SMPSO performs significantly better with a much smaller computation effort.

F. Influence of Grouping and Transformation Methods

In this subsection we will examine the effects of different grouping mechanisms and transformation functions. We implement the four grouping mechanisms and the three transformation functions explained in Section V. The WOF-SMPSO algorithm is used for comparison. For each combination of a grouping mechanism and a transformation function we perform the same experiment as in Subsection VI-B on the WFG1-4, DTLZ1-4 and ZDT problems with 2 and 3 objectives. As we are only interested in the efficiency of the mechanisms, the DG was precomputed for each problem using an ϵ-value of 10^{-10}, i.e. the consumed evaluations were not counted to those of the optimization process.

Table III lists the number of problems (out of the 21 test problems) for which the respective combination performed best, second best and worst. We observe that the algorithms using the p-value transformation (ψ_2) and the multiplication transformation ψ_1 perform on average better than the ψ_3 transformation function. Comparing the grouping mechanisms, we see that the random grouping seems to perform worse than the other mechanisms. The best combination was obtained when using the DG method together with the p-value (ψ_2) transformation.

We take a closer look at the performance of DG. As a single-objective grouping strategy, it is based on only the first objective in our work. Therefore, we did not expect its high performance for the multi-objective problems. If we look closer for instance at the ZDT problems, which are all separable, we observe that all their variables are in the same (single) group (since all separable variables by default are grouped together by the original DG algorithm). This results in the optimization of just one weight value for all variables. For all $i = 2, ..., n$, reducing the value of a variable x_i in a ZDT problem is improving solution quality in the second objective component. Therefore, we can expect a speed-up in convergence by using just one single weight.

The better performance of the linear grouping compared to the random one (in ψ_1 and ψ_3) can be explained by taking a look at the variables of the WFG problems. Since out of the 1000 variables the first quarter of them are position-related variables, a linear grouping with $\gamma = 4$ groups results in a splitting of position and distance variables. This might be a benefit compared to the random grouping and therefore leads to a better overall performance. Our observations further show that DG, although just regarding one objective, can approximately divide position and distance variables of some of the WFG problems into different groups.

Overall, no single combination is clearly superior in the majority of the 21 test problems used in this experiment. These observations show that the choice of a good grouping mechanism can strongly affect the performance of the WOF method. Finding a single mechanism that fits all problems

might be a difficult task. It can be expected however, that by developing an advanced multi-objective grouping mechanism, the performance can be further improved.

G. Sensitivity Analysis

In this part we take a closer look at the different parameters of the WOF and analyze the sensitivity to certain changes in these parameter values. The tested parameters are (1) q, the number of chosen solutions $\vec{x}'_k$, (2) γ, the number of groups used in the ordered grouping mechanism, (3) δ, the share of the total function evaluations used for the weight optimization, (4) p, the value used in the ψ_2 transformation function, (5) t_1, the amount of evaluations used for each optimization of the original problem (Line 4 in Algorithm 1) and (6) t_2, the amount of evaluations used for each optimization of the transformed problem (Line 5 in Algorithm 2). For each parameter we tested five different values including values higher and lower than those described previously in Subsection VI-A. All other parameters remain unchanged. The sensitivity experiments are done with the WOF-SMPSO and the WOF-NSGA-II algorithms using the UF, WFG, DTLZ and ZDT benchmark functions with two and three objectives, where applicable, and 21 independent runs are performed for each configuration. The detailed results can be found in Tables 1 to 12 of the supplement materials of this article.

For the parameters t_1 and t_2 we observe a low sensitivity in general. While the performance varies when ranging t_1 from 400 to 2000 and t_2 from 100 to 900, the differences in the hypervolume are in most cases not statistically significant for both tested algorithms. For the value of p in the transformation function, in most problem instances a significantly lower performance was observed for a very small value of 0.05 in both algorithms. For the WOF-SMPSO, performance remained stable for values between 0.2 and 0.5, while the NSGA-II achieved on average better results with higher values of p. The value of δ was altered between 0.1 and 1.0, and we observed an average increase in performance for higher values of δ, especially when the NSGA-II is used within our framework. On the other hand the results when using the SMPSO also show, that the optimal value for δ might depend on the used optimizer in WOF as well as the problem to be optimized. Future work on the framework should therefore include an automatic detection of the starting point for the traditional optimization phase. For the parameter γ, the number of groups, we observe that the WOF-SMPSO and WOF-NSGA-II perform significantly worse if the values of γ are too high ($\gamma = 50$) or too low ($\gamma = 2$). These observations however, were made for the described p-value transformation (ψ_2), which we do not claim to be a perfect grouping mechanism. A more suitable grouping mechanism (e.g. from [12] or [29]) could be used within WOF to obtain more reliable results, since the optimal number of groups might depend on each specific problem. Lastly, for the parameter q, the number of chosen solutions, differences in the performance were observed by varying q, but most of these were not significant with the exception of choosing extreme values ($q = 1$ or $q = 8$). This result was to be expected as a different number of q can be used to balance between convergence and diversity.

VII. Conclusion

In this work we explained and examined the idea proposed in [3] in detail. We introduce the use of a framework for multi-objective optimization that is able to improve the performance of multi-objective optimization algorithms for many-variable problems. The proposed Weighted Optimization Framework (WOF) is not making use of any coevolution as many other approaches in this area and is designed to employ another multi-objective optimization algorithm while transforming the problem during the optimization. We explained the underlying problem transformation mechanism and the algorithm in detail and examined its implications for the optimization process. Different grouping and transformation functions were introduced. Through experimental evaluation we showed significant improvements of the performance on various benchmark problems compared to classical optimization methods as well as existing large-scale approaches. The best performance was achieved when the SMPSO algorithm was used in the proposed framework. Overall, the WOF showed significant improvements in solution quality and convergence rate for almost all tested benchmark problems.

Future work might involve the implementation of better strategies for choosing the used solutions $\vec{x}'_k$ for problem transformations. An important issue in future research might also be the development of more advanced multi-objective grouping mechanisms, since current intelligent grouping methods are mostly single-objective oriented. Furthermore, since the q different transformed problems are optimized independently, this research can easily be extended to a parallel version in the future.

Acknowledgment

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

References

[1] 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.

[2] 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.

[3] H. Zille, H. Ishibuchi, Y. Nojima, and S. Mostaghim, "Weighted optimization framework for large-scale multi-objective optimization," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference (GECCO) Companion*. ACM, 2016, pp. 83–84.

[4] 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.

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

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 1, 2002, pp. 825–830.

[7] Q. Zhang, A. Zhou, and S. Zhao, "Multiobjective optimization test instances for the CEC 2009 special session and competition," *University of Essex*, 2008.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, 2016.

[13] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015.

[14] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature PPSN III*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Springer Berlin Heidelberg, 1994, vol. 866, pp. 249–257.

[15] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

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

[17] 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.

[18] 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.

[19] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *IEEE Congress on Evolutionary Computation (CEC)*, 2007, pp. 3523–3530.

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

[21] 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.

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

[23] 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.

[24] 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.

[25] 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*. New York, USA: ACM Press, 2015, pp. 313–320.

[26] 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.

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

[28] Y. Zhang, M. Zhou, Z. Jiang, and J. Liu, "A multi-agent genetic algorithm for big optimization problems," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 703–707.

[29] X. Zhang, Y. Tian, Y. Jin, and R. Cheng, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2016.

[30] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[31] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms — a comparative case study," *Parallel Problem Solving from Nature PPSN V*, pp. 292–301, 1998.

Heiner Zille is currently working as a research assistant and pursuing his PhD degree at the Institute for Intelligent Cooperating Systems, Otto von Guericke University Magdeburg, Germany. He studied computer science and economics at the Karlsruhe Institute of Technology (KIT) in Germany and received his B.S. and M.S. degrees in 2011 and 2014 respectively. During his studies, he spent seven months at the Doshisha University in Kyoto and one year at the Osaka Prefecture University in Sakai, Japan. His research focusses on multi-objective optimization, in particular on problems with large numbers of input variables.

Hisao Ishibuchi (M'93-SM'10-F'14) received the B.S. and M.S. degrees in precision mechanics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from Osaka Prefecture University, Sakai, Japan, in 1992. Since 1987, he had been with Osaka Prefecture University for 30 years. Currently, he is a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science Technology, Shenzhen, China. His current research interests include fuzzy rule-based classifiers, evolutionary multiobjective optimization, many-objective optimization, and memetic algorithms. Dr. Ishibuchi was the IEEE Computational Intelligence Society (CIS) Vice-President for Technical Activities from 2010 to 2013. He is an IEEE CIS AdCom Member from 2014 to 2019, an IEEE CIS Distinguished Lecturer from 2015 to 2017, and an Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2014 to 2019. He is also an Associate Editor for the *IEEE Transactions on Evolutionary Computation*, the *IEEE Transactions on Cybernetics*, and the *IEEE ACCESS*.

Sanaz Mostaghim is a full professor of computer science at the Otto von Guericke University Magdeburg, Germany. She holds a PhD degree in electrical engineering and computer science from the University of Paderborn, Germany. She worked as a post-doctoral fellow at ETH Zurich in Switzerland and as a lecturer at Karlsruhe Institute of technology (KIT), Germany. Her research interests are in the area of evolutionary multi-objective optimization, swarm intelligence, and their applications in robotics, science and industry. She serves as an associate editor for the *IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on System, Man and Cybernetics: Systems* and *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Yusuke Nojima (M'00) received the B.S. and M.S. degrees in mechanical engineering from the Osaka Institute of Technology, Osaka, Japan, in 1999 and 2001, respectively, and the Ph.D. degree in system function science from Kobe University, Hyogo, Japan, in 2004. Since 2004, he has been with Osaka Prefecture University, Sakai, Japan, where he was a Research Associate and is currently an Associate Professor with the Department of Computer Science and Intelligent Systems. His current research interests include multiobjective genetic fuzzy systems, evolutionary multiobjective optimization, and parallel distributed data mining. Dr. Nojima is an Associate Editor of the *IEEE Computational Intelligence Magazine*.