An Ensemble of Scalarizing Functions and Weight Vectors for Evolutionary Multi-Objective Optimization

Diana Cristina Valencia-Rodríguez
Computer Science Department
CINVESTAV-IPN
Mexico City, Mexico
diana.valencia@cinvestav.mx

Carlos A. Coello Coello
Computer Science Department
CINVESTAV-IPN
Mexico City, Mexico
ccoello@cs.cinvestav.mx

***Abstract*—Ensembles have been used in the evolutionary computation literature to evolve several populations in an independent manner, using different search approaches. Moreover, each population's parents compete with their offspring and the other population's offspring to improve diversity. It has been shown that ensemble algorithms improve the performance of the techniques embedded within them, when considered independently. Furthermore, scalarizing functions have been successfully used in decomposition-based and some indicator-based Multi-objective Evolutionary Algorithms (MOEAs). However, it has been shown that the performance of scalarizing function tends to be tied to the geometrical shape of the Pareto front. In this work, we propose a new ensemble algorithm that adopts different scalarizing functions and weight vectors using Hungarian Differential Evolution as the baseline multi-objective optimizer. Our experimental study shows that our proposed approach outperforms the original HDE, and it is competitive with respect to modern MOEAs.**

***Index Terms*—ensemble, scalarizing functions, weight vectors, multi-objective optimization**

I. INTRODUCTION

The famous No Free Lunch Theorem (NFL) for search [1] has taught us that no single search algorithm can outperform all the others in all classes of problems. This has shifted the research focus within metaheuristics to either try to tailor our algorithms to specific problems (or classes of problems) or to design schemes that allow the combination of different (ideally complementary) mechanisms that can generalize the search scope of a particular approach. Ensemble algorithms belong to the second type of approaches, since they contain several populations that evolve simultaneously different types of techniques. Within each population, parents compete with their offspring and with the offspring of other populations with the aim of improving diversity [2].

On the other hand, scalarizing functions have been widely used to transform a multi-objective optimization problem into single-objective problems using weight vectors. Although scalarizing functions have been popularized by decomposition-based multi-objective evolutionary algorithms (MOEAs), other types of MOEAs also rely on them. One example is the so-called Hungarian Differential Evolution (HDE) algorithm, which was introduced in [3] and improved in [4].

The main idea of HDE is to transform a multi-objective problem into a Linear Assignment Problem, using a scalarizing function and a set of weight vectors. HDE was found to be a competitive MOEA which was able to deal with many-objective problems [3], [4].

Although some researchers have shred evidence about the benefits of adopting several scalarizing functions simultaneously within a MOEA (see for example [5], [6]), this sort of mechanism had not been explored so far for HDE and this is indeed the aim of this paper.

In this work, we analyze the influence of scalarizing functions and weight vectors in the behavior of HDE. With this information, we propose an ensemble of scalarizing functions and weight vectors with the obvious aim of providing a more powerful multi-objective optimizer. Our preliminary experimental results show that our proposed approach is able to improve the performance of the original HDE algorithm, and it is competitive with respect to state-of-the-art MOEAs.

The remainder of this paper is organized as follows. In Sections II and III, we present some basic background. In Section IV, we introduce the HDE algorithm and, in Section V, we analyze its performance with different scalarizing functions and weight vectors. After that, in Section VI, we present our proposed approach. Section VII shows the experimental validation of our proposed approach. Finally, we present our conclusions and some possible paths for future research in Section VIII.

II. MULTI-OBJECTIVE OPTIMIZATION

In a multi-objective optimization problem (MOP) we aim to simultaneously optimize two or more (often conflicting) objectives. Its formal definition is the following:

$$\text{minimize } \boldsymbol{f}(\boldsymbol{x}) := [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})] \quad (1)$$

$$\text{subject to: } g_i(\boldsymbol{x}) \leq 0 \quad i = 1, 2, \dots, p \quad (2)$$

$$h_i(\boldsymbol{x}) = 0 \quad i = 1, 2, \dots, q \quad (3)$$

where $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$ is the vector of decision variables, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ are the objective functions and

$g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., p$, $j = 1, ..., q$ are the constraint functions of the problem. We denote $\mathcal{X}$ as the decision space and $\mathcal{F}$ as the feasible region.

Rather than finding a single optimum solution, when solving MOPs we aim to obtain the best possible trade-offs among the objectives. For this sake, we adopt Pareto dominance, which is defined as follows. A vector $\boldsymbol{x} \in \mathcal{X}$ is said to *dominate* $\boldsymbol{y} \in \mathcal{X}$ (denoted as $\boldsymbol{x} \prec \boldsymbol{y}$), if $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ for all $i = 1, ..., m$ and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{y})$ in at least one j. On the other hand, if $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ for all $i = 1, ..., m$, then $\boldsymbol{x}$ is said to *weakly dominate* $\boldsymbol{y}$ (denoted as $x \preceq y$).

Furthermore, a vector $x \in \mathcal{X}$ is *Pareto optimal* if there does not exist another vector $y \in \mathcal{X}$, such that $y \prec x$. Therefore, in multi-objective optimization, we want to find the set of Pareto optimal solutions (called *Pareto Optimal Set*). The image (i.e., the corresponding objective function values) of the Pareto Optimal Set is called the *Pareto Optimal Front*.

III. Scalarizing Functions and Weight Vectors

A scalarizing function transforms a multi-objective optimization problem into a single-objective problem using a weight vector. In many cases and under certain assumptions, minimizing a scalarizing function can lead to a Pareto optimal solution.

Let $\boldsymbol{f'}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{z}$ where $\boldsymbol{z} := (z_1, ..., z_m)^T$ is the ideal point, *i.e.* $z_i := \min\{f_i(\boldsymbol{x}) | \boldsymbol{x} \in \mathcal{X}\}$. Also, let $\boldsymbol{w} := (w_1, ..., w_m)$ be a weight vector where $w_i \geq 0$ and $\sum_i w_i = 1$. In the following, we will introduce some scalarizing functions that we will use later on:

i) *Tchebycheff function (TCH)* [7]. The search ability of this function is equivalent to the use of Pareto-based methods. Moreover, it can produce at least weakly Pareto optimal solutions. Its definition is as follows:

$$u^{TCH}(\boldsymbol{f'}, \boldsymbol{w}) := \max_i\{w_i |f'_i|\}). \tag{4}$$

ii) *Augmented Tchebycheff (ATCH)* [7]. This function is the same as TCH, but it adds a term to avoid weakly Pareto optimal solutions. It is defined as:

$$u^{ATCH}(\boldsymbol{f'}, \boldsymbol{w}) := \max_i\{w_i |f'_i|\}) + \alpha \sum_i |f'_i|. \tag{5}$$

The behavior of ATCH depends on the value of α. For example, large values may produce unreachable non-dominated points. It is recommended that $\alpha \in [0.001, 0.01]$.

iii) *Achievement Scalarizing Function (ASF)* [7]. This function is defined in the following way:

$$u^{ASF}(\boldsymbol{f'}, \boldsymbol{w}) := \max \left\{ \frac{f'_i}{w_i} \right\}. \tag{6}$$

The ASF can produce weakly Pareto optimal solutions.

iv) *Augmented Achievement Scalarizing Function (AASF)* [7]. This function adds a term to the ASF to avoid weakly Pareto optimal solutions. It is defined as:

$$u^{ASF}(\boldsymbol{f'}, \boldsymbol{w}) := \max \left\{ \frac{f'_i}{w_i} \right\} + \alpha \sum_i \frac{f'_i}{w_i}. \tag{7}$$

A value of $\alpha \approx 10^{-4}$ is recommended.

v) *Penalty Boundary Intersection (PBI)* [7]. This function is given by

$$u^{PBI}(\boldsymbol{f'}, \boldsymbol{w}) := d_1 + \theta d_2 \tag{8}$$

where $d_1 := \left| \boldsymbol{f'} \cdot \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \right|$ and $d_2 := \left\| \boldsymbol{f'} - d_1 \frac{\boldsymbol{w}}{||\boldsymbol{w}||} \right\|$. The PBI function tries to balance convergence and diversity using θ. Hence, its behavior depends on the value of this parameter.

vi) *Artificially Generated Scalarizing Functions (AGSF1 and AGSF2)* [8]. These functions were generated using genetic programming and have been scarcely studied. In the study reported in [8], AGSF2 outperformed ASF in most of the test problems adopted. Thus, we decided to adopt AGSF2 in this work. AGSF2 is defined as follows:

$$u^{AGSF2}(\boldsymbol{f'}, \boldsymbol{w}) := \max_i \left\{ \left| w_i - \frac{f'_i}{w_i} - f'_i \right| \right\} \tag{9}$$

With the aid of scalarizing functions, we can produce a good approximation of the Pareto optimal set using evenly distributed weight vectors. One of the most widely used approaches to generate these vectors is the Simplex-Lattice-Design (SLD) [9]. The SLD generates weight vectors in a unit simplex with a uniform spacing of $\delta = 1/H$. The H parameter determines the number of divisions in each objective. Therefore, SLD generates $N = C^{H+m-1}_{m-1}$ weight vectors. One of the drawbacks of this method is that N considerably grows when the number of objectives (m) increases. Furthermore, SLD produces many vectors at the boundary of the domain.

A method that avoids some of the drawbacks of the SLD is Uniform Design with the Hammersley method (UDH) [3]. This method generates a set of weight vectors with low *discrepancy* (a numerical measure of scattering). UDH can produce any number of weight vectors without any special consideration.

IV. Hungarian Differential Evolution

Hungarian Differential Evolution (HDE) is a MOEA that adopts scalarizing functions [3]. Its core idea is to transform a multi-objective problem into a Linear Assignment Problem (LAP). In a LAP, there are n agents that must be assigned to m tasks. Assigning an agent to a task implies a cost. Thus, the goal is to perform all the tasks by minimizing the total assignment cost. The most common approach to solve a LAP is to use the Kuhn-Munkres algorithm (also known as the Hungarian algorithm) which has an algorithmic complexity $O(n^3)$ [3]. The formal definition of a LAP is the following:

Let $A = \{a_1, ..., a_n\}$ be a set of agents and $T = \{t_1, ..., t_m\}$ be a set of tasks, where $m = n$. Let $\Phi : A \rightarrow T$ be the set of all possible bijections between A and T, and $C : A \times T \rightarrow \mathbb{R}$ be the cost function. Then, a LAP can be stated as:

$$\min_{\phi \in \Phi} \sum_{a \in A} C(a, \phi(a)) \tag{10}$$

where the cost function can be represented as a square matrix C with $C_{i,j} = C(a_i, t_j)$.

Analogously, HDE has $2n$ individuals (parents and their offspring) and n weight vectors representing the Pareto front regions. The cost of assigning an individual to a weight vector denotes how well an individual represents a region. Therefore, the goal is to select the individuals that better approximate the Pareto front by minimizing the overall assignment cost.

HDE constructs the cost matrix C of the LAP by first normalizing the objective vectors and then computing each component $C_{i,j}$ using the following expression:

$$C_{i,j} := \begin{cases} u(\hat{\boldsymbol{f}}(\boldsymbol{x}_i), \boldsymbol{w_j}) & \text{for } j = 1, ..., n, \quad i = 1, ..., 2n \\ 0 & \text{for } j = n+1, ..., 2n, \quad i = 1, ..., 2n \end{cases} \tag{11}$$

where u is a scalarizing function and $\hat{\boldsymbol{f}}(\boldsymbol{x}_i)$ is the normalized objective vector of individual i.

In summary, HDE works in the following way. First, the algorithm initializes the population randomly and evaluates it. After that, it generates the weight vectors using the UDH method. During a predefined number of generations, the algorithm creates the offspring using DE/$rand$/1/bin and evaluates the resulting individuals. Then, it merges the parents with their offspring and normalizes their objective values. With the resulting information, the algorithm constructs the cost matrix using (11) given a scalarizing function and a weight vectors set. The algorithm solves the LAP using the Hungarian method, and the resulting assignment determines which individual proceeds to the next generation. Algorithm 1 summarizes the above process.

Algorithm 1 Hungarian Differential Evolution (HDE)

Require: Multi-objective problem, population size (n), maximum number of generations (g_{max}), parameters C_r and F for DE/$rand$/1/bin

Ensure: P_{gmax}

```
Generate initial population P_1 randomly
Evaluate each individual in P_1
W ← Generate n weight vectors using UDH
for g = 1 to g_max do
    P*_g ← Generate offspring from P_g using DE/rand/1/bin.
    Evaluate each individual in P*_g
    Q_g ← P_g ∪ P*_g.
    NQ_g ←Normalize objectives of each individual in Q_g
    C ← Construct a cost matrix using (11) with NQ_g, a scalarizing function and W.
    I ←Obtain the best assignment in C using the Hungarian method
    P_{g+1} ← {x_i | i ∈ I, x_i ∈ Q_g}
end for
```

V. Influence of the Scalarizing Functions in the Hungarian Differential Evolution

Initially, the HDE was implemented with the TCH scalarizing function [3], and later on, it was found that the use of the ASF function was able to improve its performance [4]. Hence, it is evident that the type of scalarizing function adopted in HDE has an impact on its performance. Furthermore, only the UDH method has been tested in HDE, and we do not know if other methods could improve its performance. In this section, we present an experimental study that analyzes the behavior of HDE using different scalarizing functions and weight vectors.

A. Experimental Setup

We tested the HDE algorithm using the scalarizing functions presented in Section III: TCH, ATCH, ASF, AASF, PBI, and AGSF2. We selected $\theta = 5$ for PBI, $\alpha = 10^{-4}$ for AASF and $\alpha = 0.005$ for ATCH. Besides the scalarizing functions, we tested the UDH and the SLD methods. For the case of the SLD vectors we adopted $H = 14$. We performed 30 independent runs of each scalarizing function with each weight vector set.

The parameters adopted for HDE in all cases were: $F = 1.0$, $C_r = 0.4$, $g_{max} = 300$ and popsize of 120. We tested the DTLZ1 - DTLZ7 problems from the Deb-Thiele-Laumanns-Zitzler (DTLZ) test suite [10], the WFG1 - WFG9 problems from Walking-Fish-Group (WFG) test suite [11], and the Minus-DTLZ and Minus-WFG test problems from [12]. All of these problems were adopted with three objectives. The number of variables in DTLZ is defined by $n = 3 + k - 1$ where $k = 5$ for DTLZ1, $k = 10$ for DTLZ2-DTLZ6 and $k = 20$ for DTLZ7. The same was applied to Minus-DTLZ. Regarding the WFG and Minus-WFG, we set the position parameters to $k = 4$ and the distance parameters to $l = 20$. In this case, we focus on which scalarizing function has the best convergence; therefore, we used the hypervolume indicator [13] for assessing the performance. This indicator measures the size of the objective space covered by the solution set. A larger value implies a better approximation.

B. Experimental Results and Discussion

The average and standard deviation of the 30 independent runs are shown in Tables I and II. The two best values of each problem are highlighted in gray, where the darker tone corresponds to the best value. In addition, the "*" symbol means that the result is statistically significant using Wilcoxon's rank-sum test with a significance level of 5%.

In 81.25% of the conventional test problems (WFG and DTLZ), the best performance scalarizing functions were the ones that adopted the SLD method. However, in the minus problems (WFG-Minus and DTLZ-Minus), the UDH method's functions had the best performance in 93.75% of the problems.

The function with the best performance in the conventional problems was AGSF2, which won in 56.25% of the problems. It was followed by the AASF which won in 25% of the problems. Regarding the minus problems, AGSF2 had the best performance with 62.5%, followed by AASF with 18.75%.

From these experimental results, we can conclude that the performance of HDE depends on the reference set adopted. Moreover, the suggested scalarizing functions are AGSF2 and AASF.

TABLE I

AVERAGE AND STANDARD DEVIATION OF HYPERVOLUME VALUES OF HDE USING DIFFERENT SCALARIZING FUNCTIONS AND WEIGHT VECTORS IN CONVENTIONAL PROBLEMS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" REPRESENTS THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

	SLD						UDH					
	AASF	AGSF2	ASF	ATCH	PBI	TCH	AASF	AGSF2	ASF	ATCH	PBI	TCH
dtlz1	1.331e+00 (4.44e-16)	1.326e+00 (1.27e-03)	1.331e+00 (4.44e-16)	1.331e+00 (4.44e-16)	1.331e+00 (4.44e-16)	1.331e+00 (3.59e-07)	1.331e+00 (1.58e-06)	1.327e+00 (1.31e-03)	1.331e+00 (1.98e-06)	1.331e+00 (5.47e-07)	1.331e+00 (3.24e-06)	1.331e+00 (7.48e-07)
dtlz2	9.647e-01 (1.54e-04)	9.659e-01 (1.59e-04)*	9.648e-01 (1.52e-04)	9.513e-01 (5.14e-04)	9.182e-01 (7.18e-03)	9.440e-01 (1.46e-03)	9.620e-01 (4.47e-04)	9.629e-01 (4.51e-04)	9.619e-01 (3.70e-04)	9.573e-01 (6.29e-04)	9.188e-01 (4.93e-03)	9.546e-01 (9.39e-04)
dtlz3	1.331e+00 (2.40e-06)	1.326e+00 (1.10e-02)	1.331e+00 (2.64e-05)	1.330e+00 (3.57e-03)	1.331e+00 (2.01e-03)	1.331e+00 (6.06e-06)	1.331e+00 (7.11e-05)	1.331e+00 (1.33e-03)	1.330e+00 (3.47e-03)	1.330e+00 (3.50e-03)	1.331e+00 (1.63e-05)	1.331e+00 (3.04e-06)
dtlz4	9.174e-01 (1.47e-03)	9.171e-01 (7.86e-03)*	9.170e-01 (3.65e-03)	9.025e-01 (1.33e-03)	8.691e-01 (5.69e-03)	8.956e-01 (1.92e-03)	9.117e-01 (8.20e-03)	9.111e-01 (1.26e-02)	9.130e-01 (4.94e-03)	9.059e-01 (1.22e-02)	8.651e-01 (6.66e-03)	9.015e-01 (1.75e-02)
dtlz5	1.097e+00 (4.94e-06)	1.097e+00 (8.55e-06)	1.097e+00 (6.08e-06)	1.098e+00 (4.01e-06)	1.097e+00 (4.74e-04)	1.098e+00 (3.35e-06)	1.094e+00 (3.46e-03)	1.090e+00 (6.88e-04)	1.093e+00 (1.55e-03)	1.099e+00 (5.66e-06)	1.096e+00 (5.43e-04)	1.100e+00 (9.91e-06)*
dtlz6	1.094e+00 (3.22e-06)	1.095e+00 (2.10e-05)	1.094e+00 (1.32e-05)	1.096e+00 (1.97e-06)	1.095e+00 (3.82e-04)	1.096e+00 (1.65e-05)	1.089e+00 (1.17e-04)	1.087e+00 (5.21e-04)	1.089e+00 (1.45e-04)	1.097e+00 (2.21e-06)*	1.096e+00 (4.07e-04)	1.097e+00 (9.37e-05)
dtlz7	6.342e-01 (1.37e-04)	6.346e-01 (2.16e-04)*	6.339e-01 (1.74e-04)	6.098e-01 (2.88e-03)	6.236e-01 (9.32e-04)	6.153e-01 (2.62e-04)	6.339e-01 (1.09e-03)	6.328e-01 (1.11e-03)	6.340e-01 (1.05e-03)	6.250e-01 (1.35e-03)	6.068e-01 (1.49e-03)	6.269e-01 (9.68e-04)
wfg1	1.217e+00 (9.38e-03)	1.218e+00 (1.13e-02)	1.218e+00 (1.24e-02)	1.192e+00 (9.09e-03)	1.148e+00 (7.83e-03)	1.187e+00 (1.08e-02)	1.183e+00 (9.21e-03)	1.185e+00 (8.15e-03)	1.182e+00 (8.58e-03)	1.186e+00 (8.92e-03)	1.123e+00 (7.81e-03)	1.189e+00 (1.27e-02)
wfg2	1.215e+00 (1.23e-02)	1.219e+00 (1.29e-02)	1.217e+00 (1.30e-02)	1.211e+00 (1.25e-02)	1.110e+00 (1.71e-02)	1.209e+00 (1.50e-02)	1.211e+00 (1.37e-02)	1.216e+00 (1.24e-02)	1.214e+00 (1.59e-02)	1.211e+00 (1.08e-02)	1.059e+00 (2.43e-02)	1.208e+00 (1.08e-02)
wfg3	9.486e-01 (6.71e-03)*	9.441e-01 (6.77e-03)	9.444e-01 (7.25e-03)	9.288e-01 (8.92e-03)	8.754e-01 (9.90e-03)	9.244e-01 (7.82e-03)	9.388e-01 (7.37e-03)	9.366e-01 (7.44e-03)	9.374e-01 (6.30e-03)	9.337e-01 (8.37e-03)	8.568e-01 (1.13e-02)	9.283e-01 (9.85e-03)
wfg4	9.172e-01 (5.80e-03)	9.191e-01 (7.12e-03)	9.178e-01 (6.07e-03)	9.098e-01 (4.56e-03)	8.095e-01 (9.18e-03)	9.030e-01 (4.55e-03)	9.054e-01 (4.87e-03)	9.086e-01 (4.19e-03)	9.071e-01 (4.80e-03)	9.109e-01 (3.91e-03)	7.507e-01 (1.11e-02)	9.045e-01 (5.59e-03)
wfg5	7.583e-01 (4.74e-03)	7.569e-01 (4.04e-03)	7.577e-01 (4.51e-03)	7.564e-01 (1.92e-03)	7.711e-01 (3.24e-03)*	7.409e-01 (2.35e-03)	7.627e-01 (3.58e-03)	7.613e-01 (2.99e-03)	7.611e-01 (3.30e-03)	7.593e-01 (3.00e-03)	7.448e-01 (3.23e-03)	7.509e-01 (3.83e-03)
wfg6	8.058e-01 (9.24e-05)	8.077e-01 (5.57e-05)*	8.058e-01 (8.36e-05)	7.855e-01 (3.75e-04)	8.037e-01 (7.75e-04)	7.725e-01 (1.08e-03)	7.996e-01 (9.84e-04)	8.003e-01 (1.09e-03)	7.990e-01 (9.38e-04)	7.934e-01 (9.73e-04)	7.851e-01 (1.76e-03)	7.859e-01 (1.43e-03)
wfg7	9.557e-01 (4.04e-03)	9.562e-01 (4.20e-03)	9.555e-01 (4.62e-03)	9.488e-01 (4.64e-03)	7.693e-01 (8.52e-03)	9.406e-01 (4.61e-03)	9.353e-01 (5.35e-03)	9.350e-01 (4.80e-03)	9.349e-01 (5.29e-03)	9.460e-01 (4.62e-03)	7.039e-01 (1.64e-02)	9.357e-01 (5.19e-03)
wfg8	8.824e-01 (5.58e-03)	8.824e-01 (8.23e-03)	8.808e-01 (5.45e-03)	8.744e-01 (7.55e-03)	7.580e-01 (1.07e-02)	8.681e-01 (6.55e-03)	8.793e-01 (6.20e-03)	8.802e-01 (6.34e-03)	8.809e-01 (5.64e-03)	8.805e-01 (5.86e-03)	7.123e-01 (1.28e-02)	8.743e-01 (6.50e-03)
wfg9	8.734e-01 (2.90e-03)	8.735e-01 (2.20e-03)	8.731e-01 (2.94e-03)	8.670e-01 (1.88e-03)	8.747e-01 (2.26e-03)	8.672e-01 (2.01e-03)	8.765e-01 (1.41e-03)	8.765e-01 (1.59e-03)	8.765e-01 (2.22e-03)	8.730e-01 (1.73e-03)	8.589e-01 (3.81e-03)	8.717e-01 (2.31e-03)

TABLE II

AVERAGE AND STANDARD DEVIATION OF HYPERVOLUME VALUES OF HDE USING DIFFERENT SCALARIZING FUNCTIONS AND WEIGHT VECTORS IN MINUS PROBLEMS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" REPRESENTS THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

	SLD						UDH					
	AASF	AGSF2	ASF	ATCH	PBI	TCH	AASF	AGSF2	ASF	ATCH	PBI	TCH
dtlz1^{-1}	2.742e-01 (6.22e-05)	2.798e-01 (5.60e-04)	2.741e-01 (8.29e-05)	2.842e-01 (4.18e-04)	2.763e-01 (8.08e-04)	2.843e-01 (3.75e-04)	2.857e-01 (4.07e-04)	2.891e-01 (3.91e-04)*	2.855e-01 (4.36e-04)	2.812e-01 (4.07e-04)	2.778e-01 (1.49e-03)	2.812e-01 (4.90e-04)
dtlz2^{-1}	9.046e-01 (5.70e-04)	9.015e-01 (5.31e-04)	9.043e-01 (6.22e-04)	9.246e-01 (1.64e-04)	9.261e-01 (3.49e-04)	9.234e-01 (1.70e-04)	9.300e-01 (5.53e-04)	9.290e-01 (2.32e-04)	9.299e-01 (5.20e-04)	9.279e-01 (2.24e-04)	9.169e-01 (1.24e-03)	9.267e-01 (1.44e-04)
dtlz3^{-1}	6.360e-01 (1.26e-02)	6.331e-01 (1.03e-02)	6.397e-01 (9.25e-03)	6.382e-01 (9.12e-03)	6.114e-01 (8.08e-03)	6.350e-01 (1.07e-02)	6.449e-01 (9.66e-03)	6.400e-01 (9.03e-03)	6.412e-01 (9.54e-03)	6.337e-01 (8.98e-03)	5.924e-01 (1.03e-02)	6.310e-01 (1.01e-02)
dtlz4^{-1}	9.066e-01 (9.68e-04)	9.020e-01 (1.17e-03)	9.047e-01 (1.10e-03)	9.265e-01 (1.79e-04)	9.252e-01 (1.22e-03)	9.233e-01 (5.19e-04)	9.313e-01 (7.71e-04)*	9.292e-01 (6.59e-04)	9.296e-01 (6.78e-04)	9.264e-01 (3.13e-04)	9.173e-01 (1.40e-03)	9.263e-01 (7.53e-04)
dtlz5^{-1}	8.033e-01 (6.44e-04)	8.010e-01 (5.27e-04)	8.032e-01 (5.91e-04)	8.221e-01 (1.53e-04)	8.210e-01 (8.40e-04)	8.215e-01 (1.34e-04)	8.263e-01 (3.18e-04)	8.268e-01 (3.05e-04)*	8.263e-01 (2.37e-04)	8.215e-01 (3.20e-04)	8.112e-01 (1.93e-03)	8.206e-01 (3.72e-04)
dtlz6^{-1}	7.234e-01 (5.20e-04)	7.211e-01 (6.82e-04)	7.235e-01 (6.16e-04)	7.396e-01 (2.30e-04)	7.428e-01 (5.11e-04)	7.388e-01 (1.64e-04)	7.480e-01 (2.41e-04)	7.500e-01 (2.19e-04)*	7.479e-01 (2.23e-04)	7.385e-01 (3.33e-04)	7.415e-01 (9.53e-04)	7.369e-01 (3.14e-04)
dtlz7^{-1}	1.200e+00 (4.82e-04)	1.201e+00 (2.37e-04)	1.201e+00 (2.80e-04)	1.201e+00 (2.23e-04)	1.200e+00 (3.73e-04)	1.201e+00 (3.49e-03)	1.200e+00 (1.60e-03)	1.201e+00 (3.91e-04)	1.200e+00 (8.59e-04)	1.201e+00 (2.82e-04)	1.197e+00 (9.36e-04)	1.202e+00 (1.84e-04)*
wfg1^{-1}	1.355e-01 (4.72e-03)	1.368e-01 (5.50e-03)	1.379e-01 (4.73e-03)	1.378e-01 (3.88e-03)	1.401e-01 (3.49e-03)	1.402e-01 (3.75e-03)	1.414e-01 (4.67e-03)	1.422e-01 (4.25e-03)	1.412e-01 (4.24e-03)	1.375e-01 (4.25e-03)	1.390e-01 (3.79e-03)	1.397e-01 (3.82e-03)
wfg2^{-1}	5.435e-01 (1.25e-02)	5.430e-01 (1.15e-02)	5.426e-01 (1.06e-02)	5.416e-01 (1.11e-02)	4.977e-01 (9.91e-03)	5.377e-01 (1.26e-02)	5.454e-01 (1.19e-02)	5.481e-01 (8.62e-03)	5.479e-01 (1.01e-02)	5.414e-01 (1.10e-02)	4.892e-01 (9.58e-03)	5.387e-01 (1.03e-02)
wfg3^{-1}	3.240e-01 (1.10e-02)	3.232e-01 (1.02e-02)	3.229e-01 (1.04e-02)	3.194e-01 (1.04e-02)	3.144e-01 (1.26e-02)	3.186e-01 (9.95e-03)	3.206e-01 (1.05e-02)	3.209e-01 (9.08e-03)	3.203e-01 (9.55e-03)	3.137e-01 (8.62e-03)	3.017e-01 (1.53e-02)	3.141e-01 (9.91e-03)
wfg4^{-1}	6.901e-01 (5.80e-04)	6.878e-01 (5.94e-04)	6.899e-01 (4.97e-04)	7.046e-01 (3.79e-04)	7.105e-01 (3.83e-04)	7.028e-01 (3.36e-04)	7.151e-01 (9.12e-04)	7.169e-01 (7.93e-04)*	7.148e-01 (9.56e-04)	7.032e-01 (5.77e-04)	7.091e-01 (1.49e-03)	7.013e-01 (6.14e-04)
wfg5^{-1}	7.037e-01 (5.71e-03)	7.036e-01 (6.24e-03)	7.038e-01 (6.38e-03)	7.216e-01 (5.99e-03)	6.895e-01 (7.45e-03)	7.176e-01 (4.64e-03)	7.203e-01 (5.84e-03)	7.206e-01 (4.00e-03)	7.211e-01 (4.69e-03)	7.221e-01 (4.38e-03)	6.453e-01 (8.60e-03)	7.161e-01 (5.12e-03)
wfg6^{-1}	7.243e-01 (1.71e-03)	7.205e-01 (2.51e-03)	7.240e-01 (2.38e-03)	7.368e-01 (2.03e-03)	7.312e-01 (3.92e-03)	7.347e-01 (1.85e-03)	7.401e-01 (2.48e-03)	7.410e-01 (1.90e-03)	7.398e-01 (2.59e-03)	7.348e-01 (2.40e-03)	7.213e-01 (3.63e-03)	7.323e-01 (1.72e-03)
wfg7^{-1}	6.919e-01 (4.72e-04)	6.897e-01 (4.85e-04)	6.919e-01 (4.27e-04)	7.059e-01 (4.82e-04)	7.128e-01 (4.57e-04)	7.042e-01 (2.88e-04)	7.157e-01 (4.54e-04)	7.182e-01 (5.39e-04)*	7.158e-01 (4.16e-04)	7.044e-01 (5.95e-04)	7.126e-01 (8.50e-04)	7.027e-01 (4.09e-04)
wfg8^{-1}	7.021e-01 (6.55e-04)	6.997e-01 (5.55e-04)	7.018e-01 (5.16e-04)	7.173e-01 (6.02e-04)	7.219e-01 (9.67e-04)	7.148e-01 (3.23e-04)	7.268e-01 (6.44e-04)	7.286e-01 (7.37e-04)*	7.264e-01 (6.22e-04)	7.159e-01 (7.60e-04)	7.196e-01 (2.29e-03)	7.134e-01 (5.39e-04)
wfg9^{-1}	6.965e-01 (3.25e-03)	6.940e-01 (2.27e-03)	6.978e-01 (2.69e-03)	7.119e-01 (3.51e-03)	7.032e-01 (4.17e-03)	7.092e-01 (2.27e-03)	7.152e-01 (2.82e-03)	7.167e-01 (2.35e-03)*	7.153e-01 (2.37e-03)	7.112e-01 (2.56e-03)	6.935e-01 (3.86e-03)	7.098e-01 (2.74e-03)

VI. OUR PROPOSED APPROACH

We saw in the previous section that the performance of HDE depends on the scalarizing function and weight vectors adopted. For example, the SLD weight vectors had the best performance in conventional problems, while in the minus problems, the UDH weight vectors had the best performance. Therefore, we hypothesized that if we could design an ensemble of different scalarizing functions and weight vectors, we can improve the overall performance of HDE, turning it into a more general multi-objective optimizer.

Based on our previous discussion, we propose here an *Ensemble of Scalarizing functions and Weight vectors* (ESW). Our proposed approach consists of four different pairs of scalarizing functions and weight vectors. Particularly, we selected AGSF2 with SLD, AASF with UDH, ASF with SLD, and AASF with UDH since they had the best performance in our previous experiment. However, it is evident that other pairs can also be adopted.

Each pair has a population that generates offspring independently using DE/$rand/1/bin$. Moreover, every population will use its corresponding pair to compute the LAP assignment cost and to select its individuals using the Hungarian algorithm (as in the original HDE). Nevertheless, it will be considered the parent, its offspring, and the other parents' offspring in all the selection processes. Therefore, the parents can be replaced by the offspring of other populations. We argue that this information exchange can improve the diversity of the populations and allows populations to help each other.

The solutions' distribution and convergence speed between subpopulations may differ because each has a different scalarizing function and weight vectors. Therefore, we include an external archive that stores the non-dominated solutions for merging the information collected by the subpopulations. If the archive exceeds a predefined size, the solution with the worst contribution of s-energy [14] is deleted. The s-energy is a performance indicator that measures the uniform distribution of a set in a d-dimensional manifold. Furthermore, its minimization leads to a uniform distribution. Therefore, we use the s-energy as a density estimator.

The s-energy is defined as [14]:

$$E_s(A) := \sum_{i \neq j} ||a_i - a_j||^{-s}, \qquad (12)$$

where $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_{|A|}\}$, $\boldsymbol{a} \in \mathbb{R}^m$, and $s \geq 0$. Thus, the s-energy contribution of a solution $\boldsymbol{a} \in A$ is defined by

$$\Delta E_s(\boldsymbol{a}, A) := \frac{1}{2}[E_s(A) - E_s(A \setminus \{\boldsymbol{a}\})]. \qquad (13)$$

Algorithm 2 displays the pseudocode of ESW, and its flowchart is presented in Figure 1.

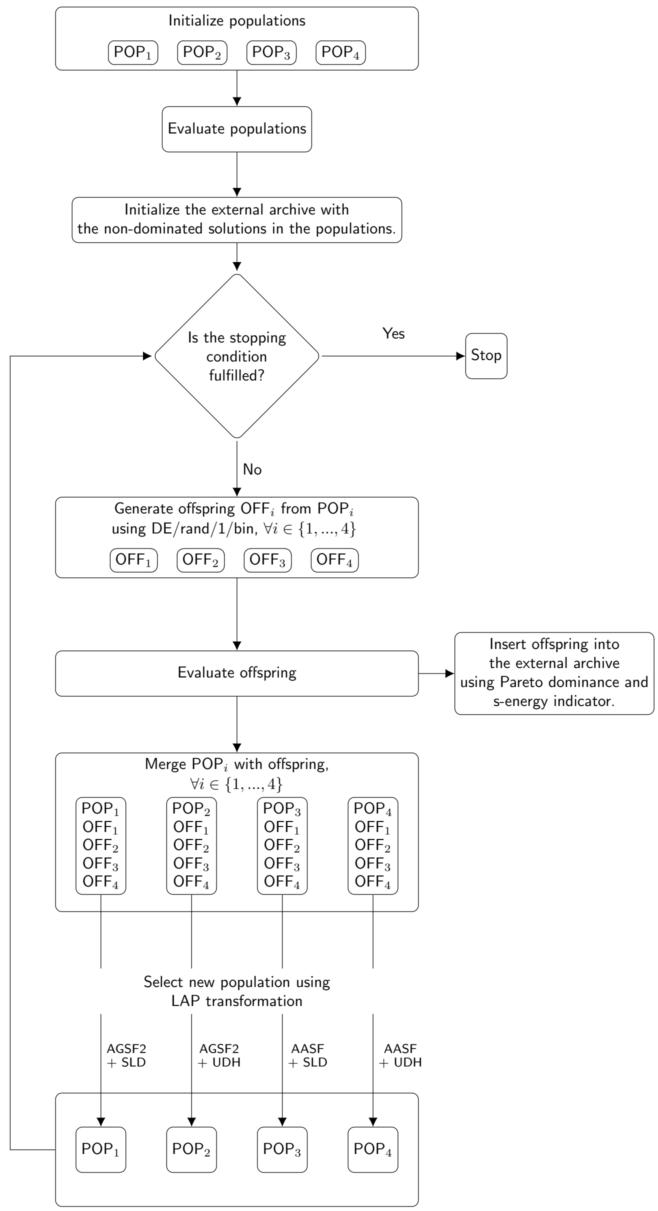

Fig. 1. General flowchart of ESW

VII. Experimental Study

In this section, we validate the performance of ESW using two experiments. The first experiment compares ESW with HDE using the pairs (scalarizing function + weight vectors) separately. We present this experiment in subsection VII-A. The second experiment compares ESW with state-of-the-art algorithms, and it is presented in subsection VII-B. In both cases, we use the hypervolume and the s-energy indicator for performance assessment. The s-energy indicator measures the solutions' distribution of a set, and a lower value is preferred. Also, in all the tables, the two best values of each problem are highlighted in gray, where the darker tone indicates the best value. In addition, the "*" symbol means that the result is statistically significant using Wilcoxon's rank-sum test with a significance level of 5%.

A. Comparison with stand-alone pairs

For this comparison, we performed 30 independent runs of ESW and HDE with the following separate pairs: AGSF2 with SLD, AGSF2 with UDH, AASF with SLD, and AASF with UDH. We adopted the benchmark problems (including their configuration), the scalarizing functions' parameters and the weight vectors' parameters from Section V. Furthermore, Table III displays the parameters of the algorithms used in this experiment.

TABLE III
Parameters of the algorithms used in experiment 1

Algorithms	Parameters settings
HDE (in all versions)	F = 1.0, C_r = 0.4, g_{max} = 1100, n = 120
ESW	F = 1.0, C_r = 0.4, max function evaluations = 132000, n = 55, max archive size = 120

Algorithm 2 Ensemble of Scalarizing functions and Weight vectors (ESW)

Require: Multi-objective problem, max archive size, stopping condition, subpopulation size (n), parameters C_r and F for DE$/rand/1/bin$
Ensure: A (External Archive)
$w_{\text{SLD}} \leftarrow$ Generate weight vectors of size n using SLD
$w_{\text{UDH}} \leftarrow$ Generate weight vectors of size n using UDH
$W = \{w_{\text{SLD}}, w_{\text{UDH}}, w_{\text{SLD}}, w_{\text{UDH}}\}$
$SF = \{\text{AGSF2}, \text{AGSF2}, \text{AASF}, \text{AASF}\}$
Generate initial population POP$_i$ randomly, $\forall i = \{1, ..., 4\}$
Evaluate populations
$A \leftarrow$Obtain the non-dominated solutions in the populations
while the stopping condition is not fulfilled **do**
  for $i \leftarrow 1$ to 4 **do**
    OFF$_i$ $\leftarrow$ Generate offspring from POP$_i$ using DE$/rand/1/bin$.
  end for
  Evaluate offspring
  Insert offspring into A using Pareto dominance. If the max archive size is exceeded, the individual with the worst s-energy contribution is deleted from A.
  for $i \leftarrow 1$ to 4 **do**
    $Q_i \leftarrow \bigcup_{j=1}^{4} \text{OFF}_j \cup \text{POP}_i$
    $NQ_i \leftarrow$Normalize objectives of Q_i
    $C \leftarrow$ Construct a cost matrix using (11) with NQ_i, SF_i and W_i.
    $I_i \leftarrow$ Obtain the best assignment in C_i using the Hungarian method
    POP$_i \leftarrow \{\boldsymbol{x}_j | j \in I, \boldsymbol{x}_j \in Q_i\}$
  end for
end while

We present the results of the experiment in Tables VII and VIII. Regarding the hypervolume indicator, our proposed approach outperforms the other algorithms in 84% of the problems. Regarding the s-energy indicator, our approach has the best performance in 93% of the problems. Therefore, we can conclude that ESW outperforms HDE in almost all the test problems adopted.

B. Comparison with respect to state-of-the-art algorithms

We selected three well-known algorithms for our experimental study at the second stage: NSGA-III [15], MOEA/DD [16], and SMS-EMOA [17]. In this case, we tested many-objective problems. For this sake, we adopted a version of SMS-EMOA that uses the algorithm of HYPE [18] to approximate the hypervolume contributions when dealing with problems having more than three objectives.

We used the DTLZ1-DTLZ4 and DTLZ7 problems from the DTLZ test suite. Moreover, we adopted the minus versions of the same problems. We set the number of objectives (m) to: 3, 5, 7, and 10. The number of variables was set to $n = k+m-1$, where k takes the same values as indicated in Section V.

TABLE IV
NUMBER OF SLD PARTITIONS USED BY NSGA-III AND MOEA/DD

Objectives (m)	3	5	7	10
Number of partitions (H)	14	6	5	2,3

TABLE V
NUMBER OF SLD PARTITIONS AND SUBPOPULATION SIZES USED BY ESW

Objectives (m)	3	5	7	10
Number of partitions (H)	9	4	3	2,3
Subpopulation sizes (n)	55	70	84	275

We selected the SLD method to generate the weight vectors of MOEA/DD and NSGA-III. For problems having ten objectives, we used the two-layer approach proposed in [15] to generate the weight vectors. Table IV displays the corresponding H values for each objective. In the case of EFW, Table V displays the subpopulation sizes given the H values. We also used the two-layer approach for ten objectives. Table VI shows the population size (or maximum archive size) and the maximum number of function evaluations that the algorithms used.

The parameters of SBX and polynomial-based mutation were set to $pc = 1.0$, $pm = 1/n$, $\eta_c = 30$ and $\eta_m = 20$. We set the parameters of the DE operator to $F = 1.0$ and $Cr = 0.4$. MOEA/DD also used a neighborhood size $T = 20$, a neighborhood selection probability $\delta = 0.9$, and the PBI scalarizing function with $\theta = 5$.

Table IX shows the average and standard deviation of the hypervolume values. Regarding the DTLZ problems, the MOEA/DD had the best performance since it is ranked first place in 10 of 20 instances. The algorithm with the second-best performance was the SMS-EMOA with nine instances, and the third-best algorithm was the ESW with seven instances. In the case of Minus-DTLZ problems, the best algorithm is the SMS-EMOA that obtained first place in 13 of 20 instances. The second-best was the ESW, with seven instances. The NSGA-III and the MOEA/DD did not obtain first places in this case. We can observe from these results that the performance of the SMS-EMOA and the ESW does not depend on the Pareto front shape since both can obtain first places in all test suites. This is not the case of MOEA/DD that had the best performance in almost all the DTLZ test suite problems. However, it can not obtain first place in the remaining problems. In general, the best algorithm regarding the hypervolume indicator was the SMS-EMOA with 22 of 40 instances in the first place. The second best algorithm was our proposed ESW, which obtained

TABLE VI
GENERAL PARAMETERS USED IN EXPERIMENT 2

Number of objectives (m)	3	5	7	10
Population size (or archive size)	120	210	210	276
Max function evaluations	132000	231000	231000	303600

14 of 40 instances.

On the other hand, Table X shows the average and standard deviation of the s-energy values. For the DTLZ problems, the ESW had the best performance, with 13 of 20 instances in the first place. MOEA/DD had the second-best performance since it obtained 7 of 20 instances. In Minus-DTLZ problems, the ESW also outperforms the other algorithms with 18 of 20 instances in the first place, followed by the MOEA/DD with two instances. We can see that ESW outperforms the other algorithms in almost all the problem instances regarding the s-energy indicator.

In summary, we can conclude that ESW is a competitive approach with respect to state-of-the-art algorithms.

VIII. Conclusions

In this paper, we first analyzed the performance of HDE using different scalarizing functions and weight vector generators. We found that the SLD method was preferred for the conventional test suites (WFG and DTLZ), and that the UDH method was better for the minus test suites (Minus-WFG and Minus-DTLZ). Moreover, we concluded that AGSF2 and AASF were the top recommended scalarizing functions to be adopted in HDE.

We proposed with this information an Ensemble of Scalarizing functions and Weight vectors (ESW) that merges the best tested pairs of scalarizing functions and weight vectors. Our experimental results show that ESW outperforms HDE with the alone pairs. Moreover, we showed that ESW is competitive with respect to state of the art algorithms.

As part of our future work, we would like to test other weight vector generators since the SLD does not allow any size set. We also want to analyze the minimum subpopulation size that produces a good ESW performance since this would allow us to decrease the total number of function evaluations performed.

Acknowledgements

The first author acknowledges support from CONACyT and CINVESTAV-IPN to pursue graduate studies in computer science. The second author kindly acknowledges support from CONACyT project no. 1920 and from a 2018 SEP-Cinvestav grant (application no. 4).

References

[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 561–579, 2010.

[3] J. A. M. Berenguer and C. A. Coello Coello, "Evolutionary Many-Objective Optimization Based on Kuhn-Munkres' Algorithm," in *Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015*, A. Gaspar-Cunha, C. H. Antunes, and C. Coello Coello, Eds. Guimarães, Portugal: Springer. Lecture Notes in Computer Science Vol. 9019, March 29 - April 1 2015, pp. 3–17.

[4] L. Miguel Antonio, J. A. Molinet Berenguer, and C. A. Coello Coello, "Evolutionary Many-Objective Optimization Based on Linear Assignment Problem Transformations," *Soft Computing*, vol. 22, no. 16, pp. 5491–5512, August 2018.

[5] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Simultaneous Use of Different Scalarizing Functions in MOEA/D," in *Proceedings of the 12th annual conference on Genetic and Evolutionary Computation (GECCO'2010)*. Portland, Oregon, USA: ACM Press, July 711 2010, pp. 519–526, iSBN 978-1-4503-0072-8.

[6] R. Hernández Gómez and C. A. Coello Coello, "A Hyper-Heuristic of Scalarizing Functions," in *2017 Genetic and Evolutionary Computation Conference (GECCO'2017)*. Berlin, Germany: ACM Press, July 15-19 2017, pp. 577 –584, iSBN 978-1-4503-4920-8.

[7] M. Pescador-Rojas, R. Hernández Gómez, E. Montero, N. Rojas-Morales, M.-C. Riff, and C. A. Coello Coello, "An Overview of Weighted and Unconstrained Scalarizing Functions," in *Evolutionary Multi-Criterion Optimization, 9th International Conference, EMO 2017*, H. Trautmann, G. Rudolph, K. Klamroth, O. Schütze, M. Wiecek, Y. Jin, and C. Grimme, Eds. Münster, Germany: Springer. Lecture Notes in Computer Science Vol. 10173, March 19-22 2017, pp. 499–513, iSBN 978-3-319-54156-3.

[8] A. V. Bernabé Rodríguez and C. A. Coello Coello, "Generation of New Scalarizing Functions Using Genetic Programming," in *Parallel Problem Solving from Nature – PPSN XVI*, T. Bäck, M. Preuss, A. Deutz, H. Wang, C. Doerr, M. Emmerich, and H. Trautmann, Eds. Leiden, The Netherlands: Springer. Lecture Notes in Computer Science, 2020, pp. 3–17.

[9] I. Das and J. E. Dennis, "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[10] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. London: Springer London, 2005, pp. 105–145.

[11] S. Huband, L. Barone, L. While, and P. Hingston, "A Scalable Multi-objective Test Problem Toolkit," in *Evolutionary Multi-Criterion Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 280–295.

[12] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 169–190, April 2017.

[13] E. Zitzler, "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Suiza, Nov. 1999.

[14] D. Hardin and E. Saff, "Discretizing Manifolds via Minimum Energy Points," *Notices of the American Mathematical Society*, vol. 51, no. 10, pp. 1186–1194, 2004.

[15] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, August 2014.

[16] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, October 2015.

[17] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 16 September 2007.

[18] J. Bader and E. Zitzler, "HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, Spring 2011.

TABLE VII

AVERAGE AND STANDARD DEVIATION OF HYPERVOLUME VALUES OF THE COMPARISON OF STAND-ALONE PAIRS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" INDICATES THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

		SLD		UDH	
	ESW	AASF	AGSF2	AASF	AGSF2
dtlz1	1.331e+00 (5.62e-07)	1.331e+00 (3.59e-07)	1.322e+00 (3.23e-03)	1.331e+00 (6.00e-06)	1.324e+00 (1.74e-03)
dtlz2	7.562e-01 (2.87e-04)	7.544e-01 (5.43e-05)	7.565e-01 (3.28e-05)*	7.465e-01 (7.80e-04)	7.468e-01 (1.10e-03)
dtlz3	1.331e+00 (1.85e-04)	1.331e+00 (2.04e-06)	1.330e+00 (1.89e-03)	1.331e+00 (6.22e-06)	1.330e+00 (4.31e-03)
dtlz4	7.515e-01 (2.10e-02)	7.538e-01 (9.13e-05)	7.531e-01 (1.48e-02)*	7.434e-01 (9.28e-03)	7.432e-01 (6.00e-03)
dtlz5	2.671e-01 (2.63e-05)*	2.571e-01 (7.82e-06)	2.591e-01 (3.63e-06)	2.649e-01 (3.88e-06)	2.649e-01 (3.02e-05)
dtlz6	2.671e-01 (2.40e-05)*	2.571e-01 (9.78e-06)	2.591e-01 (8.01e-07)	2.649e-01 (1.32e-05)	2.648e-01 (4.79e-05)
dtlz7	6.186e-01 (6.30e-04)*	6.102e-01 (1.01e-04)	6.108e-01 (8.84e-05)	6.090e-01 (9.65e-04)	6.078e-01 (7.19e-04)
wfg1	1.168e+00 (1.98e-02)	1.190e+00 (1.46e-02)	1.194e+00 (1.70e-02)	1.120e+00 (1.86e-02)	1.122e+00 (1.55e-02)
wfg2	1.248e+00 (1.54e-03)*	1.235e+00 (3.70e-03)	1.238e+00 (5.33e-03)	1.235e+00 (2.24e-03)	1.235e+00 (2.71e-03)
wfg3	8.828e-01 (2.06e-03)*	8.700e-01 (2.37e-03)	8.703e-01 (2.26e-03)	8.626e-01 (2.52e-03)	8.566e-01 (2.59e-03)
wfg4	7.589e-01 (5.84e-03)*	7.397e-01 (6.32e-03)	7.403e-01 (8.02e-03)	7.240e-01 (2.76e-03)	7.252e-01 (3.50e-03)
wfg5	7.444e-01 (3.53e-03)*	7.250e-01 (3.24e-03)	7.248e-01 (2.34e-03)	7.283e-01 (2.94e-03)	7.257e-01 (2.98e-03)
wfg6	7.557e-01 (2.40e-04)*	7.532e-01 (4.64e-05)	7.553e-01 (2.31e-05)	7.441e-01 (8.33e-04)	7.431e-01 (1.04e-03)
wfg7	7.662e-01 (9.00e-04)*	7.507e-01 (3.75e-03)	7.503e-01 (3.22e-03)	7.385e-01 (1.88e-03)	7.381e-01 (1.74e-03)
wfg8	7.520e-01 (3.55e-03)*	7.271e-01 (5.53e-03)	7.269e-01 (4.52e-03)	7.179e-01 (4.55e-03)	7.202e-01 (4.79e-03)
wfg9	8.380e-01 (1.07e-03)*	8.218e-01 (2.44e-03)	8.227e-01 (2.76e-03)	8.280e-01 (1.94e-03)	8.285e-01 (1.30e-03)
dtlz1^{-1}	3.050e-01 (1.69e-04)*	2.741e-01 (5.31e-05)	2.807e-01 (2.29e-04)	2.865e-01 (1.79e-04)	2.901e-01 (1.21e-04)
dtlz2^{-1}	9.388e-01 (2.62e-04)*	9.041e-01 (2.74e-04)	9.014e-01 (3.21e-04)	9.298e-01 (2.39e-04)	9.289e-01 (2.60e-04)
dtlz3^{-1}	7.194e-01 (9.72e-04)*	6.903e-01 (7.52e-04)	6.881e-01 (5.39e-04)	7.105e-01 (1.13e-03)	7.119e-01 (1.31e-03)
dtlz4^{-1}	9.387e-01 (2.81e-04)*	9.058e-01 (3.82e-04)	9.017e-01 (6.27e-04)	9.316e-01 (3.21e-04)	9.290e-01 (4.70e-04)
dtlz5^{-1}	9.647e-01 (2.14e-04)*	9.320e-01 (2.20e-04)	9.289e-01 (2.23e-04)	9.562e-01 (4.65e-04)	9.557e-01 (6.91e-04)
dtlz6^{-1}	8.122e-01 (2.87e-04)*	7.776e-01 (1.75e-04)	7.745e-01 (2.64e-04)	8.042e-01 (1.76e-04)	8.054e-01 (2.04e-04)
dtlz7^{-1}	1.156e+00 (1.02e-04)*	1.152e+00 (6.53e-04)	1.154e+00 (3.21e-04)	1.151e+00 (2.43e-03)	1.154e+00 (6.05e-04)
wfg1^{-1}	1.504e-01 (3.89e-03)*	1.334e-01 (4.69e-03)	1.380e-01 (3.83e-03)	1.400e-01 (3.86e-03)	1.431e-01 (2.57e-03)
wfg2^{-1}	4.082e-01 (9.21e-04)*	4.002e-01 (2.61e-04)	4.004e-01 (3.12e-04)	4.004e-01 (6.79e-04)	4.001e-01 (7.57e-04)
wfg3^{-1}	3.116e-01 (7.05e-04)*	2.814e-01 (5.12e-04)	2.874e-01 (3.84e-04)	2.914e-01 (2.17e-03)	2.940e-01 (1.53e-03)
wfg4^{-1}	7.214e-01 (4.59e-04)*	6.894e-01 (2.12e-04)	6.874e-01 (3.09e-04)	7.151e-01 (7.41e-04)	7.172e-01 (8.55e-04)
wfg5^{-1}	7.317e-01 (1.79e-03)*	6.970e-01 (1.60e-03)	6.951e-01 (1.28e-03)	7.017e-01 (2.12e-03)	7.006e-01 (2.14e-03)
wfg6^{-1}	7.485e-01 (2.07e-03)*	7.167e-01 (1.31e-03)	7.139e-01 (1.38e-03)	7.284e-01 (2.16e-03)	7.297e-01 (2.50e-03)
wfg7^{-1}	7.227e-01 (3.95e-04)*	6.904e-01 (2.22e-04)	6.886e-01 (2.34e-04)	7.151e-01 (4.71e-04)	7.174e-01 (3.63e-04)
wfg8^{-1}	7.228e-01 (4.00e-04)*	6.908e-01 (2.91e-04)	6.887e-01 (3.01e-04)	7.155e-01 (7.06e-04)	7.177e-01 (7.35e-04)
wfg9^{-1}	7.222e-01 (4.69e-03)*	6.952e-01 (3.18e-03)	6.933e-01 (3.64e-03)	7.122e-01 (3.11e-03)	7.126e-01 (3.11e-03)

TABLE VIII

AVERAGE AND STANDARD DEVIATION OF S-ENERGY VALUES OF THE COMPARISON OF STAND-ALONE PAIRS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" REPRESENTS THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

		SLD		UDH	
	ESW	AASF	AGSF2	AASF	AGSF2
dtlz1	9.978e+08 (1.59e+08)	1.093e+09 (5.54e+04)	8.506e+07 (3.98e+08)	1.337e+09 (3.16e+06)	1.151e+07 (2.13e+07)
dtlz2	8.475e+04 (7.18e+01)*	8.948e+04 (8.53e+01)	8.628e+04 (4.68e+01)	1.002e+05 (7.06e+02)	9.865e+04 (4.27e+02)
dtlz3	5.650e+08 (9.79e+08)	3.940e+11 (1.05e+12)	1.627e+11 (8.30e+11)*	1.047e+12 (2.29e+12)	6.824e+08 (1.82e+08)
dtlz4	9.058e+04 (2.88e+04)*	9.006e+04 (3.27e+02)	4.427e+08 (2.38e+09)	1.321e+05 (1.21e+05)	1.792e+05 (4.31e+05)
dtlz5	1.494e+06 (4.36e+03)	7.951e+12 (8.84e+12)	1.590e+11 (1.47e+11)	6.285e+08 (2.00e+09)	1.265e+08 (9.90e+07)
dtlz6	1.493e+06 (3.50e+03)*	8.742e+12 (7.93e+12)	1.725e+11 (1.99e+11)	3.374e+08 (3.38e+08)	5.618e+08 (9.96e+08)
dtlz7	1.527e+05 (2.54e+03)*	2.208e+12 (4.68e+12)	5.290e+09 (4.89e+09)	1.398e+12 (3.44e+12)	2.897e+11 (1.56e+12)
wfg1	2.254e+05 (1.40e+04)*	4.779e+11 (1.28e+12)	3.468e+11 (6.17e+11)	3.394e+10 (1.81e+11)	5.183e+08 (2.23e+09)
wfg2	1.453e+05 (4.82e+03)*	2.990e+11 (8.05e+11)	3.350e+11 (7.79e+11)	8.551e+08 (2.30e+09)	1.432e+10 (6.11e+10)
wfg3	3.618e+05 (1.00e+04)*	1.081e+12 (1.01e+12)	1.066e+12 (1.22e+12)	5.437e+08 (1.63e+09)	4.583e+08 (1.99e+09)
wfg4	8.874e+04 (4.32e+02)*	1.198e+11 (3.98e+11)	4.513e+10 (1.83e+11)	1.222e+09 (3.50e+09)	3.446e+09 (1.71e+10)
wfg5	8.810e+04 (9.77e+02)*	3.449e+07 (3.68e+07)	1.490e+08 (4.21e+08)	4.975e+05 (1.66e+06)	4.962e+06 (1.87e+07)
wfg6	8.463e+04 (6.55e+01)*	8.928e+04 (7.44e+01)	8.610e+04 (3.57e+01)	1.004e+05 (4.60e+02)	9.936e+04 (3.04e+02)
wfg7	8.608e+04 (1.17e+02)*	7.882e+11 (1.01e+12)	2.369e+11 (3.11e+11)	4.671e+08 (1.63e+09)	9.104e+08 (3.16e+09)
wfg8	9.903e+04 (9.84e+02)*	8.889e+11 (1.20e+12)	6.502e+11 (1.06e+12)	2.166e+10 (5.75e+10)	2.228e+10 (6.45e+10)
wfg9	1.043e+05 (5.23e+02)*	2.343e+11 (5.49e+11)	5.213e+11 (8.09e+11)	2.619e+06 (6.14e+06)	3.630e+07 (1.63e+08)
dtlz1^{-1}	1.412e+05 (1.92e+02)*	4.218e+12 (2.81e+12)	1.338e+10 (2.49e+10)	6.819e+10 (3.59e+11)	5.942e+05 (1.35e+05)
dtlz2^{-1}	1.401e+05 (1.51e+02)*	1.810e+09 (4.84e+09)	1.186e+07 (4.12e+06)	9.128e+07 (4.69e+08)	6.670e+10 (3.59e+11)
dtlz3^{-1}	8.550e+04 (2.92e+02)*	5.656e+12 (3.58e+12)	4.574e+12 (2.72e+12)	1.188e+10 (2.54e+10)	1.154e+10 (3.18e+10)
dtlz4^{-1}	1.402e+05 (1.49e+02)*	1.748e+11 (3.48e+11)	3.238e+07 (5.52e+07)	1.219e+09 (4.85e+09)	1.028e+06 (5.13e+05)
dtlz5^{-1}	1.853e+05 (2.09e+02)*	2.140e+10 (8.13e+10)	3.591e+08 (1.34e+08)	1.451e+06 (1.86e+06)	1.939e+08 (5.99e+08)
dtlz6^{-1}	1.097e+05 (1.02e+02)*	5.409e+09 (2.03e+10)	5.080e+07 (6.82e+07)	8.816e+05 (1.17e+06)	8.136e+06 (2.43e+07)
dtlz7^{-1}	5.292e+05 (1.52e+04)*	9.253e+12 (2.21e+13)	8.558e+10 (4.48e+11)	9.083e+12 (2.62e+13)	8.051e+11 (2.49e+12)
wfg1^{-1}	2.357e+05 (3.69e+04)*	2.053e+10 (5.92e+10)	1.534e+12 (1.43e+12)	5.005e+10 (2.18e+11)	3.581e+11 (1.18e+12)
wfg2^{-1}	8.977e+04 (1.93e+02)*	2.771e+13 (3.02e+13)	2.315e+13 (1.93e+13)	2.184e+10 (6.10e+10)	8.469e+10 (2.52e+11)
wfg3^{-1}	1.427e+05 (2.62e+02)*	2.050e+13 (2.48e+13)	3.406e+13 (2.68e+13)	2.404e+10 (6.81e+10)	4.770e+09 (1.73e+10)
wfg4^{-1}	8.478e+04 (1.00e+02)*	2.148e+11 (8.31e+11)	2.131e+08 (7.62e+08)	9.581e+06 (4.67e+07)	2.261e+07 (1.15e+08)
wfg5^{-1}	9.645e+04 (1.34e+03)*	1.548e+11 (3.90e+11)	1.174e+11 (2.58e+11)	1.383e+08 (4.71e+08)	5.313e+07 (1.24e+08)
wfg6^{-1}	9.037e+04 (4.82e+02)*	9.099e+11 (1.04e+12)	7.835e+11 (1.05e+12)	1.728e+08 (3.95e+08)	1.506e+08 (3.38e+08)
wfg7^{-1}	8.485e+04 (9.93e+01)*	6.028e+11 (1.24e+12)	1.636e+11 (7.27e+11)	1.059e+11 (3.73e+11)	1.731e+10 (8.97e+10)
wfg8^{-1}	8.490e+04 (8.99e+01)*	1.371e+11 (4.99e+11)	4.776e+09 (2.56e+10)	4.763e+09 (2.56e+10)	8.681e+06 (4.07e+07)
wfg9^{-1}	9.902e+04 (8.49e+02)*	1.042e+10 (2.23e+10)	1.781e+10 (6.09e+10)	1.129e+06 (3.23e+06)	3.112e+05 (3.49e+05)

TABLE IX

AVERAGE AND STANDARD DEVIATION OF HYPERVOLUME VALUES OF THE COMPARISON WITH STATE-OF-THE-ART ALGORITHMS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" REPRESENTS THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

	m	ESW	NSGA-III	MOEA/DD	SMS-EMOA(m = 3) / SMS-EMOAHype (m = 5,7,10)
dtlz1	3	1.2974e+0 (3.5e-2)	1.3062e+0 (4.3e-5)	1.3066e+0 (2.1e-5)	*1.3067e+0 (9.7e-6)
	5	1.6103e+0 (8.2e-7)	1.6103e+0 (4.5e-5)	1.6103e+0 (1.1e-6)	1.6099e+0 (1.6e-4)
	7	1.9478e+0 (2.2e-5)	1.9474e+0 (1.8e-3)	*1.9479e+0 (3.7e-6)	1.9472e+0 (2.6e-4)
	10	2.5937e+0 (8.9e-16)	2.5937e+0 (3.6e-7)	2.5937e+0 (3.6e-7)	2.5937e+0 (8.9e-16)
dtlz2	3	7.6057e-1 (2.8e-4)	7.5514e-1 (1.5e-4)	7.5890e-1 (1.1e-6)	*7.6811e-1 (4.7e-5)
	5	*1.3508e+0 (7.0e-4)	1.3468e+0 (5.3e-4)	1.3478e+0 (4.8e-6)	1.3474e+0 (1.5e-3)
	7	1.8399e+0 (1.1e-3)	1.8434e+0 (3.5e-4)	*1.8446e+0 (4.7e-6)	1.8378e+0 (1.2e-3)
	10	2.5936e+0 (2.5e-4)	2.5842e+0 (1.7e-2)	*2.5937e+0 (4.4e-16)	2.5936e+0 (5.9e-5)
dtlz3	3	1.3226e+0 (3.3e-2)	1.3308e+0 (1.4e-5)	1.3308e+0 (3.7e-7)	*1.3308e+0 (3.7e-7)
	5	1.6105e+0 (4.6e-5)	1.6105e+0 (2.2e-16)	1.6105e+0 (2.2e-16)	1.6105e+0 (2.2e-16)
	7	1.9487e+0 (4.4e-16)	1.9487e+0 (4.4e-16)	1.9487e+0 (4.4e-16)	1.9487e+0 (4.4e-16)
	10	2.5937e+0 (8.9e-16)	2.5937e+0 (6.0e-7)	2.5937e+0 (8.9e-16)	2.5937e+0 (8.9e-16)
dtlz4	3	7.849e-1 (2.e-2)	7.6273e-1 (1.1e-1)	7.8725e-1 (1.4e-6)	*7.1642e-1 (1.3e-1)
	5	1.3351e+0 (1.3e-3)	1.3311e+0 (2.5e-4)	1.3314e+0 (4.2e-6)	*1.3365e+0 (1.3e-3)
	7	1.8510e+0 (1.4e-3)	1.8190e+0 (7.5e-2)	*1.8533e+0 (1.8e-6)	1.8529e+0 (9.4e-4)
	10	2.5937e+0 (7.e-5)	2.5937e+0 (6.0e-5)	*2.5937e+0 (8.9e-16)	2.5937e+0 (3.1e-6)
dtlz7	3	1.2285e+0 (9.2e-5)	1.2222e+0 (3.7e-3)	1.2233e+0 (1.3e-4)	*1.228e+0 (5.6e-3)
	5	*1.4522e+0 (5.9e-4)	1.439e+0 (2.0e-3)	1.3898e+0 (4.e-2)	1.2622e+0 (1.1e-1)
	7	*1.6759e+0 (1.5e-2)	1.5796e+0 (5.2e-2)	4.5235e-1 (7.6e-2)	8.2143e-1 (5.1e-1)
	10	1.9117e+0 (2.2e-1)	1.9058e+0 (8.7e-2)	2.4389e-1 (3.9e-2)	1.9654e+0 (3.8e-1)
dtlz1^{-1}	3	*3.0505e-1 (1.7e-4)	2.8573e-1 (1.3e-3)	2.6546e-1 (6.1e-4)	1.9195e-1 (1.2e-2)
	5	1.9193e-2 (1.5e-4)	1.2290e-2 (1.1e-3)	1.0192e-2 (1.6e-4)	1.8941e-2 (1.1e-3)
	7	4.4404e-4 (6.3e-6)	3.3421e-4 (2.8e-5)	2.0216e-4 (6.9e-6)	*4.8428e-4 (3.7e-5)
	10	7.8126e-7 (3.7e-8)	1.0556e-6 (1.4e-7)	1.3846e-7 (1.6e-8)	*1.2840e-6 (1.2e-7)
dtlz2^{-1}	3	9.3876e-1 (2.6e-4)	9.2134e-1 (1.8e-3)	9.1874e-1 (6.6e-4)	*9.4007e-1 (1.8e-4)
	5	*4.7853e-1 (1.8e-3)	4.2668e-1 (4.4e-3)	3.5354e-1 (1.5e-3)	4.3612e-1 (7.4e-3)
	7	1.3525e-1 (8.7e-4)	1.1411e-1 (4.4e-3)	8.4628e-2 (1.6e-3)	*1.4347e-1 (3.7e-3)
	10	1.4428e-2 (2.4e-4)	1.3372e-2 (8.5e-4)	6.8312e-3 (3.1e-4)	*2.0770e-2 (5.4e-4)
dtlz3^{-1}	3	*7.1942e-1 (9.7e-4)	7.0608e-1 (2.7e-3)	7.0371e-1 (6.5e-4)	5.0008e-1 (1.9e-2)
	5	1.6358e-1 (2.1e-3)	1.2602e-1 (6.3e-3)	8.2782e-2 (2.7e-3)	*1.9704e-1 (5.8e-3)
	7	9.7422e-3 (1.1e-3)	9.0438e-3 (1.4e-3)	8.5155e-3 (4.5e-4)	*3.5046e-2 (1.3e-3)
	10	7.6167e-5 (1.5e-5)	4.1322e-4 (1.0e-4)	5.7394e-4 (8.1e-5)	*3.8553e-3 (3.4e-4)
dtlz4^{-1}	3	9.3872e-1 (2.8e-4)	9.2217e-1 (1.9e-3)	9.1861e-1 (4.1e-4)	*9.4017e-1 (1.8e-4)
	5	*4.7745e-1 (8.e-3)	4.264e-1 (4.7e-3)	3.4831e-1 (2.2e-3)	4.4191e-1 (6.4e-3)
	7	1.3462e-1 (8.2e-4)	1.0158e-1 (5.5e-3)	8.2317e-2 (7.e-4)	*1.4507e-1 (2.8e-3)
	10	8.429e-3 (1.6e-3)	1.0796e-2 (1.e-3)	6.9314e-3 (4.6e-4)	*2.1365e-2 (5.8e-4)
dtlz7^{-1}	3	1.3116e+0 (1.1e-5)	1.3105e+0 (5.9e-4)	1.3111e+0 (4.4e-5)	*1.3117e+0 (1.4e-6)
	5	1.5785e+0 (3.5e-4)	1.5649e+0 (3.0e-3)	1.5462e+0 (6.3e-2)	1.5728e+0 (1.1e-2)
	7	*1.8984e+0 (6.7e-4)	1.8419e+0 (8.2e-3)	1.1554e+0 (3.1e-2)	1.8642e+0 (1.4e-2)
	10	2.3558e+0 (3.5e-2)	2.3513e+0 (1.3e-2)	1.2544e+0 (3.5e-2)	*2.4844e+0 (4.1e-2)

TABLE X

AVERAGE AND STANDARD DEVIATION OF S-ENERGY VALUES OF THE COMPARISON WITH STATE-OF-THE-ART ALGORITHMS. THE TWO BEST VALUES ARE HIGHLIGHTED IN GRAY (DARK GRAY IS THE BEST, AND LIGHT GRAY IS THE SECOND BEST). THE "*" REPRESENTS THAT THE RESULT IS STATISTICALLY SIGNIFICANT.

	m	ESW	NSGA-III	MOEA/DD	SMS-EMOA(m = 3) / SMS-EMOAHype (m = 5,7,10)
dtlz1	3	*5.4852e+5 (8.8e+4)	6.7586e+5 (1.9e+3)	6.0073e+5 (3.5e+2)	6.1437e+5 (1.5e+3)
	5	*5.8482e+8 (9.9e+7)	4.8462e+15 (2.6e+16)	1.7714e+9 (1.1e+6)	1.9274e+11 (5.4e+11)
	7	*2.7427e+7 (2.6e+6)	2.0833e+33 (1.1e+34)	2.8939e+7 (7.1e+4)	1.6979e+30 (9.1e+30)
	10	*1.0902e+30 (4.e+30)	1.5708e+54 (1.9e+54)	3.1455e+29 (1.1e+28)	1.2687e+54 (2.2e+54)
dtlz2	3	*8.5177e+4 (7.2e+1)	1.1143e+5 (3.6e+3)	8.9408e+4 (1.3e+0)	1.1827e+5 (1.8e+3)
	5	*2.9926e+5 (9.6e+2)	3.5488e+5 (4.6e+2)	3.5543e+5 (2.1e+1)	2.6909e+10 (1.3e+11)
	7	*2.8434e+5 (6.e+3)	4.6017e+5 (1.5e+3)	4.6397e+5 (8.8e+1)	3.333e+20 (1.8e+21)
	10	*6.8291e+8 (2.9e+9)	3.4296e+48 (1.8e+49)	6.9118e+9 (3.2e+8)	1.0096e+29 (4.5e+29)
dtlz3	3	*1.5886e+7 (3.6e+7)	2.782e+7 (6.6e+6)	2.0173e+7 (3.2e+4)	2.3244e+7 (3.7e+5)
	5	*1.7255e+13 (2.1e+13)	4.0516e+13 (1.3e+14)	1.6377e+13 (6.3e+10)	1.5793e+25 (2.1e+25)
	7	*1.0611e+20 (5.1e+20)	2.6420e+34 (1.4e+35)	9.6665e+18 (7.9e+16)	2.0949e+36 (2.2e+36)
	10	*3.3870e+48 (1.8e+49)	1.6999e+55 (1.3e+55)	5.5979e+34 (1.3e+34)	4.3020e+55 (7.7e+55)
dtlz4	3	*9.4209e+4 (3.e+4)	1.1207e+5 (2.3e+4)	9.2487e+4 (1.3e+0)	8.8115e+5 (1.3e+6)
	5	*2.8891e+5 (4.8e+3)	3.3831e+5 (4.9e+2)	3.3869e+5 (1.4e+1)	7.9963e+11 (4.2e+12)
	7	*3.2145e+5 (5.9e+3)	5.6666e+29 (3.1e+30)	4.8228e+5 (6.3e+1)	3.9256e+16 (2.1e+17)
	10	*2.6341e+16 (1.4e+17)	9.3176e+46 (5.0e+47)	2.5528e+11 (3.e+10)	3.6538e+33 (2.e+34)
dtlz7	3	3.4745e+5 (8.5e+3)	6.7388e+7 (1.3e+8)	*5.1791e+5 (7.e+5)	7.6398e+5 (2.6e+5)
	5	*1.0179e+13 (5.5e+13)	1.3371e+15 (6.8e+15)	4.2344e+10 (1.3e+11)	3.9672e+19 (2.1e+20)
	7	*9.4473e+4 (3.1e+3)	1.913e+19 (6.7e+19)	9.2988e+10 (1.6e+10)	7.9183e+23 (4.2e+24)
	10	*2.3439e+3 (1.3e+2)	1.5425e+34 (8.3e+34)	7.8423e+12 (3.4e+13)	1.9316e+34 (7.1e+34)
dtlz1^{-1}	3	*1.4117e+5 (1.9e+2)	2.4779e+11 (9.2e+11)	7.8507e+5 (2.2e+6)	2.0554e+6 (6.6e+5)
	5	*1.3196e+6 (1.2e+4)	6.6751e+22 (3.6e+23)	2.7882e+9 (3.5e+9)	6.4933e+11 (3.4e+12)
	7	*2.5558e+6 (1.9e+4)	1.5617e+25 (8.2e+25)	1.2222e+17 (6.5e+17)	2.5159e+16 (1.0e+17)
	10	*1.6882e+8 (3.4e+7)	1.9274e+32 (8.4e+32)	9.5644e+23 (4.7e+24)	4.3405e+23 (2.3e+24)
dtlz2^{-1}	3	*1.4005e+5 (1.5e+2)	2.3182e+9 (1.2e+10)	1.8862e+5 (5.2e+4)	2.0916e+5 (1.7e+3)
	5	*7.5610e+5 (1.3e+4)	1.6784e+20 (9.e+20)	7.4462e+9 (3.5e+10)	1.9575e+10 (9.2e+10)
	7	*8.7556e+5 (6.8e+3)	5.2344e+19 (2.7e+20)	3.5629e+13 (7.7e+13)	4.1913e+14 (1.5e+15)
	10	*2.9202e+6 (6.4e+4)	1.7751e+29 (6.9e+29)	2.3295e+21 (1.0e+22)	1.4417e+20 (6.3e+20)
dtlz3^{-1}	3	*8.5503e+4 (2.9e+2)	2.2985e+7 (7.6e+7)	1.1212e+5 (4.1e+4)	5.6999e+6 (1.6e+6)
	5	*2.8263e+5 (2.8e+3)	1.0907e+13 (3.3e+13)	2.9823e+9 (1.5e+10)	1.4955e+11 (4.8e+11)
	7	*2.5871e+5 (9.2e+3)	5.6391e+18 (3.0e+19)	2.7063e+14 (1.5e+15)	2.2508e+15 (1.2e+16)
	10	*1.9501e+6 (1.3e+5)	6.8219e+31 (3.7e+32)	7.5909e+20 (3.8e+21)	3.7049e+17 (1.8e+18)
dtlz4^{-1}	3	*1.4023e+5 (1.5e+2)	1.4933e+9 (4.2e+9)	1.3408e+10 (3.4e+10)	2.0823e+5 (1.6e+3)
	5	*7.6196e+5 (5.2e+4)	6.9684e+22 (3.6e+23)	2.088e+23 (7.9e+23)	2.5420e+11 (9.3e+11)
	7	*8.8492e+5 (7.2e+3)	8.3333e+33 (4.5e+34)	1.0620e+33 (5.6e+33)	1.3106e+15 (6.8e+15)
	10	*5.6494e+6 (1.7e+6)	9.7702e+37 (5.3e+38)	6.4856e+47 (1.9e+48)	1.0617e+23 (5.2e+23)
dtlz7^{-1}	3	4.6499e+6 (1.8e+5)	7.4787e+10 (2.9e+11)	*7.2603e+5 (1.1e+5)	1.3095e+7 (2.2e+5)
	5	*7.3814e+6 (1.1e+6)	4.6987e+17 (2.1e+18)	2.1525e+7 (2.3e+7)	1.2511e+19 (6.7e+19)
	7	*1.8044e+6 (3.8e+4)	8.3523e+20 (4.0e+21)	4.7305e+11 (1.4e+12)	7.5992e+25 (3.8e+26)
	10	9.0018e+23 (4.3e+24)	6.3153e+29 (3.4e+30)	9.8209e+14 (3.1e+15)	3.1702e+38 (1.7e+39)