



Universiteit
Leiden
The Netherlands

Dominance-based variable analysis for large-scale multi-objective problems

Irawan, D.; Naujoks, B.; Bäck, T.H.W.; Emmerich, M.T.M.

Citation

Irawan, D., Naujoks, B., Bäck, T. H. W., & Emmerich, M. T. M. (2022). Dominance-based variable analysis for large-scale multi-objective problems. *Natural Computing*, 22, 243-257. doi:10.1007/s11047-022-09910-5

Version: Publisher's Version

License: [Creative Commons CC BY 4.0 license](https://creativecommons.org/licenses/by/4.0/)

Downloaded from: <https://hdl.handle.net/1887/3486274>

Note: To cite this publication please use the final published version (if applicable).



Dominance-based variable analysis for large-scale multi-objective problems

Dani Irawan^{1,2} · Boris Naujoks² · Thomas Bäck¹ · Michael Emmerich^{1,3}

Accepted: 25 July 2022 / Published online: 25 August 2022
© The Author(s) 2022

Abstract

Optimization problems with multiple objectives and many input variables inherit challenges from both large-scale optimization and multi-objective optimization. To solve the problems, decomposition and transformation methods are frequently used. In this study, an improved control variable analysis is proposed based on dominance and diversity in Pareto optimization. Further, the decomposition method is used in a cooperative coevolution framework with orthogonal sampling mutation. The algorithm's performances are compared against the weighted optimization framework. The results show that the proposed decomposition method has much better accuracy compared to the traditional method. The results also show that the cooperative coevolution framework with a good grouping is very competitive. Additionally, the number of search directions in orthogonal sampling can be easily configured. A small number of search directions will reduce the search space greatly while also restricting the area that can be explored and vice versa.

Keywords Evolutionary algorithms · Large-scale · Multi-objective · Grouping · Decomposition · Cooperative coevolution

1 Introduction

Large-scale optimization problems are often solved by decomposing them into several subproblems or by using dimensionality reduction methods. The cooperative coevolution (CC) framework (Potter and De Jong (1994)) is the most common framework used to decompose the problems. As for dimensionality reduction, Yang

et al. introduced EACC-G. No definition in the paper, simply they want to call the algorithm with this name in (Yang et al. 2008). The EACC-G framework is based on the CC framework but instead of optimizing the subproblems created by grouping, the EACC-G optimizes weights that are associated with the groups. The framework effectively reduces the dimensionality to the number of groups created which is less than or equal to the number of variables.

These techniques for large-scale optimization are also used when there are several objectives to be optimized simultaneously, i.e., in a large-scale multi-objective (LSMO) optimization problem. Some examples are the cooperative coevolution with generalized differential evolution (CCGDE3) (Antonio and Coello 2013) and the weighted optimization framework (WOF) (Zille et al. 2017) which are based on the CC framework and EACC-G, respectively. In addition to those, new frameworks designed for LSMO optimization problems are also available, such as the multiobjective evolutionary algorithm based on decision variable analysis (MOEA/DVA) (Ma et al. 2016) as well as algorithms that utilize machine learning techniques such as the SVM + NSGA-II (Zhang et al. 2019) and PCA-MOEA (Liu et al. 2020).

✉ Dani Irawan
irawan_dani@yahoo.com

Boris Naujoks
boris.naujoks@th-koeln.de

Thomas Bäck
t.h.w.baeck@liacs.leidenuniv.nl

Michael Emmerich
m.t.m.emmerich@liacs.leidenuniv.nl

- ¹ Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
- ² Institute for Data Science, Engineering, and Analytics, TH Köln – University of Applied Sciences, Gummiesbach, Germany
- ³ Multiobjective Optimization Group, Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

In most of the mentioned frameworks and algorithms, variable decomposition/grouping plays a major role. In regards to grouping, there are many ways of specifying groups for the framework/algorithm. There are simple methods like random grouping which allocate variables to groups randomly to a fixed size, but there also exist more sophisticated methods. As an example, the random dynamic grouping (Liu et al. 2020) uses varying group sizes at each iteration which is determined by a roulette wheel selection based on the quality of the groups. Combined with the WOF and using MMOPSO as the optimizer, the random dynamic grouping shows good performance on WFG (Huband et al. 2006) and UF (Zhang et al. 2009) test problems.

Other common methods used to group variables are based on variable interactions or separability. A problem is called partially separable iff:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = (\arg \min_{x_1} f(\mathbf{x}), \dots, \arg \min_{x_n} f(\mathbf{x})) \quad (1)$$

Further, a function is additively separable iff:

$$f(\mathbf{x}) = \sum_{s=1}^{|\mathcal{S}|} f_s(x_s), \quad |\mathcal{S}| > 1 \quad (2)$$

The variables are grouped together when they do not fulfill the above criteria for separability, i.e., they are non-separable. This ensures that variables that affect each other are optimized together. The differential grouping (DG) family (Omidvar et al. 2014, 2017; Sun et al. 2019) focuses on grouping based on this additive separability. Another method that has been used to group variables is differential analysis (DA). These methods will be discussed further in Sect. 3.

In LSMO problems, such variable interactions must be checked for each objective function. Variables that are separable with respect to one objective can be non-separable on other objectives and it is non-trivial on how to process such variables. To define how such variables are being considered separable or non-separable, the so-called transfer strategies are used (Sander et al. 2018).

As an alternative to grouping based on interaction, variables can be grouped based on whether they affect convergence or diversity of solutions in the objective space. Some examples of such grouping methods are the control variable analysis (CVA) (Ma et al. 2016), and decision variable clustering (Zhang et al. 2018). The advantage of these methods is that no transfer strategies are needed. To summarize, the different grouping methods are listed in Table 1.

This paper provides studies on extensions and modifications to existing methods for solving LSMO problems. The topics covered are:

- an improvement on the CVA grouping method;
- implementing the improved CVA in a cooperative coevolution framework;
- implementing the novel mirrored orthogonal sampling method introduced in Wang et al. (2019) to solve LSMO problems.

This paper is organized as follows: Sect. 2 introduces several existing frameworks for large scale optimization that use grouping in their routine, Sect. 3 discusses existing grouping methods that have been used to solve LS optimization problems, Sect. 4 introduces a modification to the CVA grouping method to improve its accuracy, Sect. 5 discusses the implementation of the CVA and orthogonal sampling in a CC framework, and lastly, Sect. 6 summarizes the work.

2 Large-scale optimization frameworks

In this section, existing frameworks for solving large-scale problems are described. The frameworks discussed in this section are based on decomposition. Discussion on the decomposition methods is available in Sect. 3.

2.1 Cooperative coevolution

The challenge with large-scale optimization problems is that common solver's performances deteriorate rapidly as the number of variables increases due to the exponential expansion of the search space (Li et al. 2013). The cooperative coevolution (CC) framework (Potter and De Jong 1994) is intended to scale up these solvers by decomposing the large-scale problem into several smaller subproblems that are solved cooperatively.

The first step of the CC framework is the grouping of the variables. The grouping methods will be discussed further in Sect. 3.

After grouping, the second step is the optimization itself. The variables in different groups are optimized separately, while variables in the same group will be optimized together in one subproblem. Each subproblem optimizes a subvector \mathbf{x}_S from the large-scale problem, with S being the index set defining the subproblem, $S \subseteq \{1, \dots, n\}$.

The optimization subproblems are constructed as optimizing \mathbf{x}_S while all other variables $\mathbf{x}_{\bar{S}}$ are filled with the so-called context vector (CV) and kept constant (with \bar{S} the complement of the index set S). The subproblems' formulation can be seen in Eq. 3.

$$\min_{\mathbf{x}_S} f(\mathbf{x}'), \quad x'_i = \begin{cases} x_i & \text{if } i \in S \\ x_i^{(cv)} & \text{otherwise} \end{cases} \quad (3)$$

Table 1 Commonly used grouping/decomposition methods

Grouping method	# groups	# objective	Grouping principle	Prominent application
Differential grouping	Problem dependent	1	Separability	CC framework Sun et al. (2019), Omidvar et al. (2014)
Differential analysis	Problem dependent	1	Contributions	CC framework, MOFBVE Mahdavi et al. (2017)
Random	User defined	≥ 1	Random	CC framework Liu et al. (2020), Yang et al. (2008), WOF Zille et al. (2017)
Linear	User defined	≥ 1	Variable index	CC framework, WOF Zille et al. (2017)
SVM	User defined	≥ 1	Clustering	SVM+NSGA-II Zhang et al. (2019)
CVA	3	≥ 2	Dominance levels	MOEA/DVA Ma et al. (2016)
Decision var. clustering	2	≥ 2	Cluster of angles relative to the normal of (f_1, f_2, \dots, f_m)	LMEA Zhang et al. (2018), PCA-MOEA Liu et al. (2020)

In Eq. 3, the CV denoted as $\mathbf{x}^{(cv)}$ and variable whose index is in \mathcal{S} are substituted with $\mathbf{x}_{\mathcal{S}}$. The CV is built from the representative individuals of each subproblem, e.g. the best individuals of each subvector (Trunfio 2015).

After $\mathbf{x}_{\mathcal{S}}$ is optimized with respect to the current CV, the CV is updated with the optimized $\mathbf{x}_{\mathcal{S}}$. This concept is illustrated in Fig. 1. The process is then repeated for several iterations (known as cycles) until a stopping criterion is triggered.

2.2 MOEA/DVA

The difficulties of solving LSMO problems are two-fold. There are challenges related to the multi-objectivity and also challenges related to the large number of variables involved. Common solvers for multi-objective optimization problems (MOPs) struggle due to these challenges. Similar to the CC framework, the MOEA/DVA is intended to scale up these solvers (Ma et al. 2016).

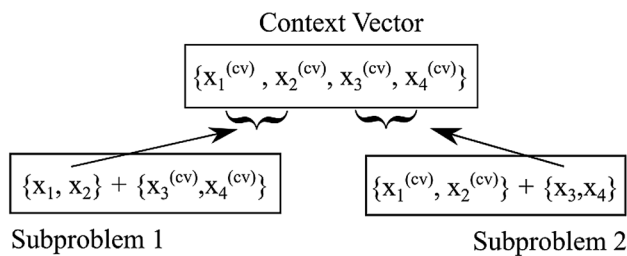


Fig. 1 Optimization and update of the context vector. In each cycle, each subproblem is optimized consecutively. In the example above, a 4-variable problem is decomposed into two subproblems. Subproblem 1 optimizes the first two variables, while the other two variables are taken from the context vector. Solving subproblem 1 yields optimized x_1 and x_2 which are used to update the context vector. Subsequently, subproblem 2 will optimize x_3 and x_4 while the value for x_1 and x_2 are taken from the updated context vector

The MOEA/DVA also uses grouping in its routine. There are two stages of grouping, first, it groups variables using control variable analysis (CVA, see Sect. 3.3), and then it creates subgroups on the convergence and mixed variables based on interaction using differential grouping (DG, see Sect. 3.1). Decomposing large scale optimization problems using several grouping methods are common to control group sizes and interaction sensitivity (e.g., Zhenyu et al. 2008; Yue and Sun 2021), or increase grouping efficiency (e.g., Irawan et al. 2020). The two-stage grouping in MOEA/DVA differs in that the first stage is to address the multi-objective nature of the problem. Additionally, as mentioned in Sect. 1, when dealing with LSMO problems, non-separability can happen differently in each objective. In MOEA/DVA, variables interacting indirectly on different objectives must be grouped together. In MOEA/DVA, the optimization is divided into two big phases. In the first phase, several points are sampled uniformly on the diversity variables subspace and will be kept static during this phase. During this phase, optimization effort is focused on convergence and mixed variable groups to push the solution closer to the Pareto front.

After a certain stopping criterion is fulfilled, the second phase of MOEA/DVA is to optimize all variables together. The primary aim of this last phase is to uniformly spread solutions in the objective space. Ma et al. (2016) tested MOEA/DVA on various large-scale test problems with good results although the number of function evaluations is very high, mainly due to the DG.

2.3 Weighted optimization framework

The weighted optimization framework (WOF) reduces the problem’s dimensionality by changing the values of a group of decision variables simultaneously. This is

achieved by utilizing the transformation function, ψ which specifies a search direction for a group of variables. The offspring are generated along this search direction from an individual $\mathbf{x}^{(k)}$ in the population by varying a weight vector \mathbf{W} . The optimization problem is then transformed into finding a weight vector that produces good objective values, i.e.

$$\begin{aligned} \min_{\mathbf{W}} \mathbf{f}_k(\mathbf{W}) &= \min_{\mathbf{W}} \mathbf{f}(\psi(\mathbf{W}, \mathbf{x}^{(k)})) \\ \psi(\mathbf{W}, \mathbf{x}^{(k)}) &= \{W_1 x_1^{(k)}, \dots, W_l x_l^{(k)}, \dots, \\ &\quad W_{\gamma} x_{n-l+1}^{(k)}, \dots, W_{\gamma} x_n^{(k)}\} \end{aligned} \tag{4}$$

In Eq. 4 it is assumed that the variables are grouped into γ groups with the same size $l = n/\gamma$. The transformed problem is created with the hope that the search directions intersect with the Pareto set. The transformed problem and the original problem are solved in a loop, one after the other.

To summarize, in the WOF, the problem is transformed into optimizing weights to a specific search direction where each group of variables is represented with a straight line. These search directions are defined by a good candidate solution position and random grouping of variables. Other algorithms have also emerged based on this concept. He et al. (2019) used a similar method with two search directions. Qin et al. (2021) further increased the number of search directions for the directed sampling in their LMOEA-DS to increase the chance for the direction vectors to intersect with the Pareto set in the decision space. The search directions in LMOEA-DS are generated based on the solutions closest to the ideal point of the objective space.

3 Grouping methods

As mentioned before, in solving large-scale problems, the use of grouping to decompose the large-scale problem is prominent. This section will discuss several well-known grouping methods in more detail.

3.1 Differential grouping

Recall that one of the most common methods used to group variables is based on separability. Detecting additive separability in a function is possible by evaluating the second-

order differentials of the function. The second-order differential is computed following Eq. 5.

$$\begin{aligned} \Delta_i &= f(\dots, x'_i, \dots) - f(\dots, x_i, \dots) \\ \Delta_{i,j} &= f(\dots, x'_i, \dots, x'_j, \dots) - f(\dots, x_i, \dots, x'_j, \dots) \\ \Theta_{i,j} &= \begin{cases} 1 & \Delta_i \neq \Delta_{i,j} \\ 0 & \Delta_i = \Delta_{i,j} \end{cases} \tag{5} \\ i, j &\in (1, \dots, n) \end{aligned}$$

In Eq. 5, Δ_i is the effect of perturbation on x_i while $\Delta_{i,j}$ is the effect of perturbation on x_i after an initial perturbation on x_j . The $\Theta_{i,j}$ is a binary matrix which indicates whether the variables interact with each other. A complete $\Theta_{i,j}$ with i and j spanning all variables yields what is called a Design Structure Matrix (DSM) (Omidvar et al. 2017). Two variables x_i and x_j are additively separable when $\Theta_{i,j} = 0$. A full DSM can be constructed using the so-called DG2 method, consuming $\frac{n^2-n}{2}$ function evaluations (Omidvar et al. 2017).

For multi-objective problems, the DG must be conducted on each objective function. The computational cost is then multiplied by the number of objectives. Variables may be separable on one objective, but not on other objectives. Care should be taken as indirect interactions across different objectives are possible. As an example, suppose that x_1 interacts with x_2 on the first objective, and x_2 interacts with x_3 on the second objective. This implies x_1 is indirectly interacting with x_3 .

3.2 Differential analysis

While the DG uses second-order differentials, another method, known as differential analysis (DA) (Morris 1991; Campolongo et al. 2005), uses the first-order differential. In DA, multiple samples of the first-order differentials are drawn and the mean and variance of the differences are taken as scores (known as sensitivity indices) for each variable. It is important to note that the DA by itself does not produce groups. The groups are created separately after the analysis. In Mahdavi et al. (2017), the groups are created based on the scores using a clustering method, while in Irawan et al. (2020) the groups are created based on the ranking of one of the scores.

For DA, each variable in the search space is divided into p intervals. The scores are then calculated based on the

elementary effects (EE). The elementary effect of variable x_j is calculated according to Eq. 6.

$$EE_i(\mathbf{x}) = \frac{f(\dots, x_i + \Delta, \dots) - f(\mathbf{x})}{\Delta}, \quad i = 1, \dots, n \quad (6)$$

$\Delta > 0$ is a grid jump which is chosen from a multiple of $1/(p - 1)$, \mathbf{x} is a random point in the search space such that $\mathbf{x} + \Delta$ is still within the search space which we refer to as base points. Several samples of EE_i are collected. If r samples are desired for each variable then $N = r(n + 1)$ function evaluations are required. The mean μ_i and variance σ_i of EE_i are used in Mahdavi et al. (2017) for grouping, while Irawan et al. (2020) used the mean of absolute values of EE_i , i.e. the μ^* as defined in Campolongo et al. (2005).

3.3 Control variable analysis

In MOPs, several objectives need to be considered simultaneously. The control variable analysis (CVA) is used in MOEA/DVA (Ma et al. 2016) to detect whether the variables affect convergence or diversity (or both) with respect to the Pareto front. CVA is based on the first-order effects of the variables.

For CVA, a base solution \mathbf{x} is evaluated. Afterwards, one variable $x_i, i \in \{1, \dots, n\}$, is shifted and new objective values are evaluated at the shifted position, thus $\mathbf{x}' = (\dots, x_i + \delta, \dots)$. Several δ values are used such that the whole range of x_i is filled uniformly. The sampling is then followed by non-dominated sorting to identify the order of non-domination levels (see Deb 2001 for further details on dominance and non-dominated sorting). The variable is then classified based on 3 possible results of non-dominated sorting on the set of objective vectors obtained from shifting x_i several times:

1. If all solutions belong to the same non-dominated front, then x_i is affecting diversity (will be referred to as diversity variable).
2. If each non-dominated front only contains one objective vector then x_i is affecting convergence (will be referred to as convergence variable).
3. If none of the previous two criteria is fulfilled, then x_i is affecting both diversity and convergence (will be referred to as mixed variable).

The CVA as proposed by Ma et al. (2016) was tested on WFG problems. The WFG problems have a parameter k which specifies the number of diversity variables; however, these variables can act as mixed variables instead of only affecting diversity.

In Fig. 2 we show that the traditional CVA is inconsistent. Figure 2 shows how solutions for two-dimensional WFG2 are distributed over the objective space when only

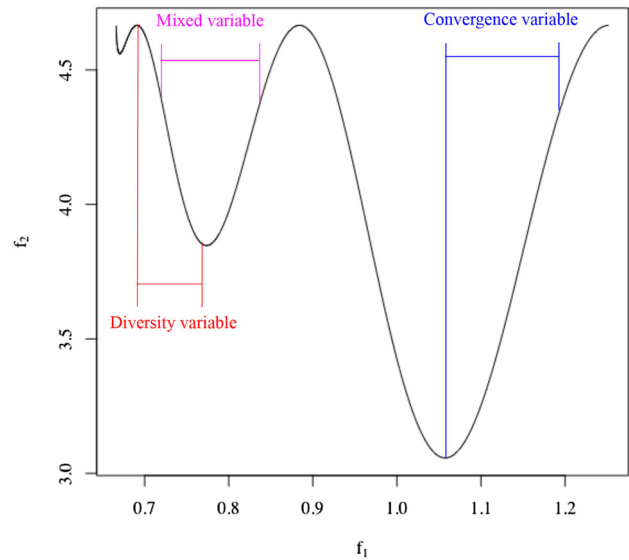


Fig. 2 Distribution of solutions in the objective space of 2 objective WFG2 by changing only the diversity variables. Depending on where diversity variables are being sampled, the variables can be identified as diversity, convergence, or even mixed variables

the diversity variables are varied. Based on the function definition in Huband et al. (2006), the objective values are determined by the average of the diversity variables. The CVA samples around the lower bound of the variables which means that it will only sample the left side of Fig. 2. When the number of variables is low, the effect of each diversity variable on their average is high and solutions will spread over a large part of the objective space and most likely will be identified as a mixed variable. However, when the number of diversity variables is high, each variable has only a minuscule effect on the average and the CVA will only identify a small portion of Fig. 2. By adding a shift to the first sampled point, the variables may be identified as any of the three possible types irrespective of how many samples are being taken. This can happen on any problem with mixed variables.

3.4 Decision variable clustering

The decision variable clustering (DVC) takes off from the same idea as the CVA, it attempts to identify which variables affect diversity and which variables affect convergence. Zhang, et al. (2018) identified some optimization problems where the diversity variables should be considered as convergence variables to guarantee convergence to the Pareto set. An example of such a problem is as follows:

$$\begin{aligned}
 f_1(x_1, x_2) &= x_1 x_2 \\
 f_2(x_1, x_2) &= \frac{(1 - x_1 x_2) + 1 - x_2}{2} \\
 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1
 \end{aligned} \quad (7)$$

If CVA is used in the problem in Eq. 7, both x_1 and x_2 will be considered diversity variables. Perturbing x_1 while keeping x_2 generates points that do not dominate each other in the objective space, under CVA, x_1 is a diversity variable (see Fig. 3). The same also applies to x_2 . However, it should be noted that as x_2 approaches 1, the objectives actually get closer to the Pareto front. In fact, the Pareto front can only be achieved if x_2 is equal to 1, therefore, x_2 is a variable that affects convergence.

To take such a case into account, the DVC starts similarly to CVA where it perturbs the value of a variable while keeping all other variables constant. It then follows up by creating a line fit on the sampled points, one line for each variable. Finally, the angle between the normal vector of the (normalized) hyperplane $f_1 + f_2 + \dots + f_m = 1$ and the fitted lines are measured and a k -means clustering method is used to determine which variables have small angles (convergence variables) and which variables have large angles (diversity variables).

4 Proposed grouping method: CVA-DA

4.1 Method description

Let us recall the inconsistency issue on CVA. The inconsistency occurs because the CVA only samples a small portion of the search space, around the lower bound. This issue was addressed in DVC (Zhang et al. 2018) by taking

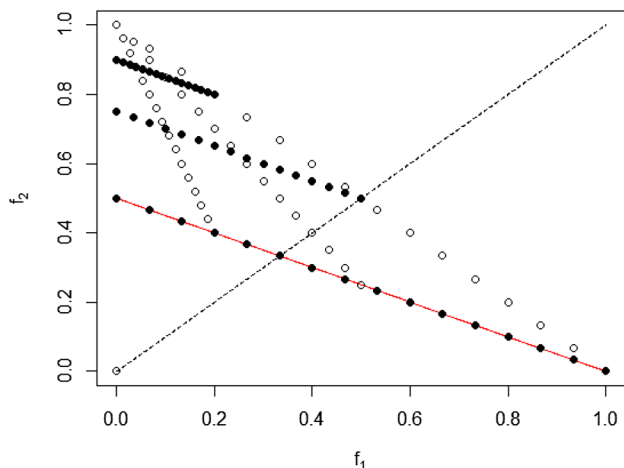


Fig. 3 The objective values obtained by using various x_1 and x_2 on the problem described in Eq. 7. Objective vectors obtained by varying x_1 while keeping x_2 at 0.2, 0.5, or 1 are depicted with solid circles where each value of x_2 forms the top, middle, and bottom row of circles, respectively. Similarly, objective vectors obtained by varying x_2 while keeping x_1 at 0.2, 0.5, or 1 are depicted with empty circles making the left, middle, and right column of circles, respectively. The solid line is the Pareto front, while the dashed line is the normal of the line $f_1 + f_2 = 1$

several base points in the search space and perturbing the variables around these base points, similar to how the DA is performed. In this work, we propose to improve the CVA in a similar way to DVC; however, we keep the usage of domination levels as the base for grouping. Domination levels must be checked separately for each base point because the control variables are different. The modified CVA is presented in Algorithm 1 and referred to as CVA-DA.

The CVA-DA differs from the DVC firstly by having an additional third category, similar to the traditional CVA: the mixed variables. In DVC, this category is deemed not informative and forced to fall into either diversity or convergence category through the clustering. However, we would argue that there are indeed variables that should be regarded as mixed variables as seen in Fig. 2. Secondly, the CVA-DA relies on domination levels using non-dominated sorting. The domination level is chosen here because the k -means clustering used in DVC has several weaknesses, such as the implicit assumption that the clusters have equal radii and the sensitivity to outliers (Raykov et al. 2016). For example, let us modify and generalize the problem in Eq. 7 into the following:

$$\begin{aligned}
 f_1(x_1, x_2, x_3) &= x_1 x_2 - x_3 + 1 \\
 f_2(x_1, x_2, x_3) &= a(-x_1 x_2) - b x_2 - c x_3 + a + b + c \\
 0 \leq x_1 &\leq 1, \\
 0 \leq x_2 &\leq 1, \\
 0 \leq x_3 &\leq 1, \\
 a, b, c &> 0
 \end{aligned}
 \tag{8}$$

In Eq. 8, variable x_1 will only affect diversity, while variable x_2 and x_3 is important for convergence to the Pareto front as the front can only be achieved if both x_2 and x_3 are equal to 1. However, using the DVC, the measured angle used for clustering the variables will depend on the values of a , b , and c thus the k -means clustering will produce different results. This means that the identified roles of the variables can change despite no change to their actual role. In the end, the DVC does not perfectly solve the issue of misclassification.

In terms of time complexity, the DVC and CVA-DA use the same amount of resources for sampling the points; however, they differ in the classification phase. The non-dominated sorting used in CVA-DA depends more on the number of objectives and the number of points being sorted, N , typically in the order of $\mathcal{O}(mN^2)$ in most implementations (Roy et al. 2016). However, there exist some fast algorithms which have $\mathcal{O}(N \log^{m-1} N)$ time complexity (Buzdalov and Shalyto 2014). On the other hand, the time complexity for DVC scales with the number of variables as it uses $\mathcal{O}(n)$ in the typical implementation of k -means clustering.

Algorithm 1 $C = \text{CVA-DA}(\mathbf{f}, m, r, n\text{sample})$

```

1:  $\mathbf{c} \leftarrow \text{zeros}(n\text{var})$  // initialize class at unknown
2:  $B \leftarrow$  draw  $r$  points in the search space
3: Evaluate  $\mathbf{f}(B)$ 
4: for  $i$  in  $1, \dots, r$  do
5:   for  $j$  in  $1, \dots, n\text{var}$  do
6:     initialize an empty list  $\mathcal{F}^{(j)}$ 
7:     for  $k$  in  $1, \dots, n\text{sample}$  do
8:        $x \leftarrow B^{(i)} + \Delta_k$  // perturbation from the base
9:       evaluate and add  $\mathbf{f}(x)$  to  $\mathcal{F}^{(j)}$ 
10:    end for
11:     $nds \leftarrow$  do non-dominated sorting on  $\mathcal{F}^{(j)}$ 
12:     $n\text{rank}_{i,j} \leftarrow$  number of fronts in  $nds$ 
13:    if  $c_j = 0$  then
14:       $c_j \leftarrow \begin{cases} 1, & \text{if } n\text{rank}_{i,j} = 1 \text{ (distrib. var.)} \\ 2, & \text{if } n\text{rank}_{i,j} = n\text{sample (conv. var.)} \\ 3, & \text{otherwise (mixed var.)} \end{cases}$ 
15:    else
16:       $c_j \leftarrow 3$ , if  $n\text{rank}_{i,j} \neq n\text{rank}_{i-1,j}$  // always set to mixed if class is changed
17:    end if
18:  end for
19: end for

```

In CVA–DA, samples are taken for each base point, therefore the cost will be multiplied by the number of base points. The main improvement in using this CVA–DA is that mixed variables are more likely to be correctly classified (recall Fig. 2). However, because we are only taking samples, it is still possible to obtain the wrong classifications. In a many-objective optimization problem, this sampling poses a greater challenge because non-dominated solutions are easier to obtain. This means the variables are more likely to be classified as distribution variables.

4.2 Numerical experiment

The performance of CVA–DA is compared against the original CVA. The algorithms are tested on the WFG test suite (Huband et al. 2006) with 3 different settings. The first setting follows Ma et al. (2016) where 24 input variables are used with the number of diversity variables, k , set to 4. The other settings test the algorithms’ performances on larger input size, at 100 input variables, with $k = 20$ and $k = 80$.

For both algorithms, the number of samples taken is varied with a multiplier r and scaled with respect to the number of objectives, $m \in \{2, 3, 5\}$. The rationale for scaling it with m is that the domination can expand in any direction in the objective space, so it should be scaled by $2m^2$. The total number of samples used is therefore $2rm^2$, with $r \in \{2, 5, 10\}$. The correct classification is determined from all samples in the experiment. If one method classifies a variable as one type while other methods classify it as another type, then the classification for the variable is

”mixed”. The accuracy is measured as the number of correct classifications by the CVA or CVA–DA divided by the number of variables. The results are presented in Figs. 4, 5, and 6, where results for those test problems on which not all algorithms consistently obtained an accuracy of 1.0.

The test results show that the number of samples does not affect the accuracy of the original CVA because the randomness in the sampling only causes a small, local perturbation. Therefore, for CVA, only a single value is reported for each setting of the test problems. The CVA–DA, on the other hand, is sensitive to the number of samples as the randomness affects both the base points and the size of the perturbation. Despite the sensitivity to the number of samples, the CVA–DA always outperforms CVA on all WFG functions, except WFG1.

In all figures, it can be seen that the CVA always struggles in WFG7, WFG8, and WFG9. These are problems where parameter-dependent biases are used. This means that in these problems, the diversity variables may interact with the convergence variables and change the type of the variables from either diversity- or convergence-related to mixed variables. In CVA, the parameter-dependent biases are not properly recognized and they are left to be detected in the second phase grouping by the interaction analysis (DG). In CVA–DA, the type-switch can be properly detected provided that enough samples are taken.

The CVA fails to correctly classify variables in the WFG2 problems with 100 variables and $k = 80$. These are the cases discussed in Sect. 3.3 where the CVA struggles to detect mixed variables. The CVA–DA, on the other hand, correctly classifies the variables because it takes samples

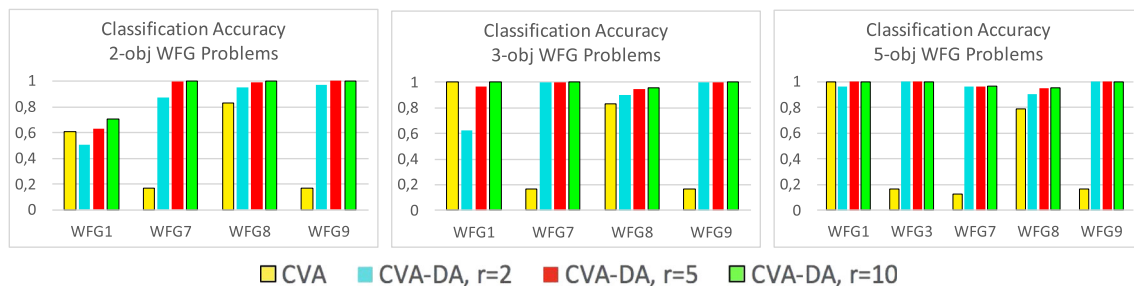


Fig. 4 Accuracy on WFG test problems with 24 variables and $k = 4$. Results are only presented for those test problems on which not all algorithms consistently obtained an accuracy of 1.0

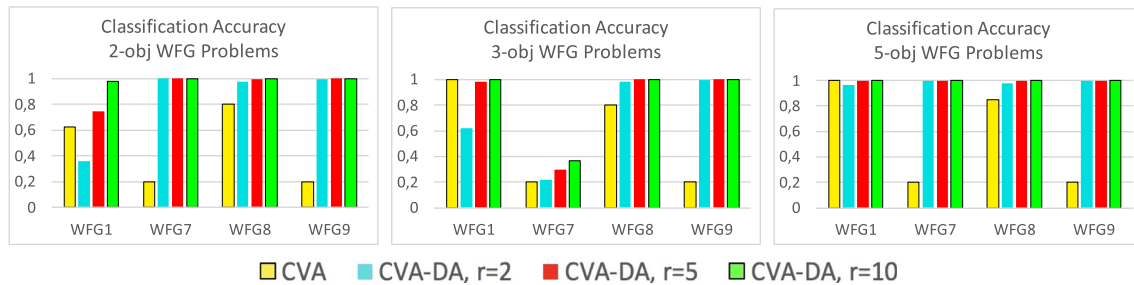


Fig. 5 Accuracy on WFG test problems with 100 variables and $k = 20$. Results are only presented for those test problems on which not all algorithms consistently obtained an accuracy of 1.0

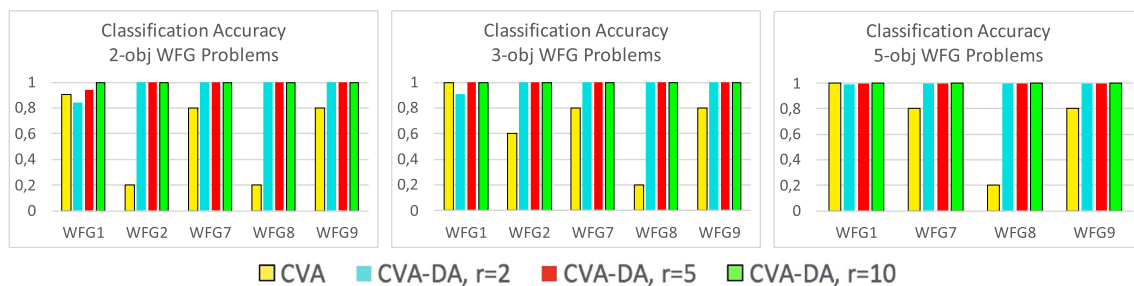


Fig. 6 Accuracy on WFG test problems with 100 variables and $k = 80$. Results are only presented for those test problems on which not all algorithms consistently obtained an accuracy of 1.0

around several base points and easily recognizes the mixed variables. With the performance improvement confirmed, we can move forward to apply CVA-DA in an MOEA based on the cooperative coevolution framework.

5 Cooperative coevolution SMS-EMOA

In our study, we further use the CVA-DA in an MOEA using the same grouping rule as MOEA/DVA, i.e. CVA-DA followed by DG. The DG used will be the recursive variant (RDG3 Sun et al. 2019) because it is very efficient as it only costs $\mathcal{O}(n \log n)$ function evaluations. The framework used here is the CC framework. This means that the diversity variables are included and are optimized from

the start as opposed to being kept constant as in MOEA/DVA. The variable groups are generated exclusively for each class (diversity, convergence, or mixed).

The solver is based on SMS-EMOA (Beume et al. 2007) where one offspring is added to the population, and the least contributor to the hypervolume is removed (see Algorithm 2) to keep the population size at 100. The algorithms are also compared against a basic SMS-EMOA algorithm without any grouping. In all experiments, the maximum number of function evaluations is set at 100,000 and ncv is set at 3.

For all algorithms, a minimum group size is also imposed. If the group size is below the minimum, it will be merged with another group. The minimum group size is set to 100 variables. This is used because currently, no

budget allocation methods are implemented and each group is set to use the same number of function evaluations (20) in each cycle. The minimum group size will then limit the number of groups generated so each cycle will not take too much resources.

To generate offspring, the genetic operator used is only a mutation based on the mirrored orthogonal sampling (Wang et al. 2019). Similar to search directions in WOF, the search directions for orthogonal sampling used here are also defined based on the variable groups; however, the number of search directions can be varied from one up to the number of variables in the group. If the upper limit is used, then the problem size is not reduced. By using the orthogonal sampling, the problems are not explicitly transformed into a weight optimization problem; however, as we are generating offspring only in the subspace

spanned by the search directions, it is effectively equivalent to weight optimization. The samples are generated around a parent using truncated normal distribution so that the offspring always stays within the box constraint. For a truncated normal distribution, if the standard deviation is too large, the distribution will be more flat; if the standard deviation is too small, the probability that the offspring will be generated near the boundary will be minuscule, limiting the search to very close proximity around the parent. By setting the standard deviation to 0.3 multiplied by the possible range, we found that the offspring have a reasonable chance to be generated near the boundaries while also maintaining the bell-shaped curve in the probability distribution.

Algorithm 2 $P = \text{CC-SMS-EMOA}(\mathbf{f}, \text{lbound}, \text{ubound}, ncv, noff, minsize)$

```

1:  $\mathbf{c} \leftarrow \text{CVA-DA}(\mathbf{f}, m, r, nsample)$  // classify variables
2:  $P \leftarrow$  initial population
3:  $\mathcal{V}_{conv} \leftarrow \emptyset, \mathcal{V}_{div} \leftarrow \emptyset, \mathcal{V}_{mixed} \leftarrow \emptyset$ 
4: for  $i$  in  $1, \dots, m$  do
5:   if  $c_i = 1$  then
6:      $\mathcal{V}_{div} \leftarrow \mathcal{V}_{div} \cup \{i\}$    elseif  $c_i = 2$ 
7:      $\mathcal{V}_{conv} \leftarrow \mathcal{V}_{conv} \cup \{i\}$ 
8:   else
9:      $\mathcal{V}_{mixed} \leftarrow \mathcal{V}_{mixed} \cup \{i\}$ 
10:  end if
11: end for
12:  $\mathcal{G}_{div} \leftarrow \text{RDG3}(\mathbf{f}, \mathcal{V}_{div})$  // RDG3 on diversity variables
13:  $\mathcal{G}_{div} \leftarrow \text{CombineGroup}(\mathcal{G}_{div}, minsize)$ 
14:  $\mathcal{G}_{conv} \leftarrow \text{RDG3}(\mathbf{f}, \mathcal{V}_{conv})$  // RDG3 on conv. variables
15:  $\mathcal{G}_{conv} \leftarrow \text{CombineGroup}(\mathcal{G}_{conv}, minsize)$ 
16:  $\mathcal{G}_{mixed} \leftarrow \text{RDG3}(\mathbf{f}, \mathcal{V}_{mixed})$  // RDG3 on mixed variables
17:  $\mathcal{G}_{mixed} \leftarrow \text{CombineGroup}(\mathcal{G}_{mixed}, minsize)$ 
18:  $\mathcal{G} \leftarrow$  groups from  $\mathcal{G}_{div}, \mathcal{G}_{conv},$  and  $\mathcal{G}_{mixed}$ 
19:  $ngroup \leftarrow |\mathcal{G}_{div}| + |\mathcal{G}_{conv}| + |\mathcal{G}_{mixed}|$ 
20: while budget available do
21:    $CM \leftarrow$  pick  $ncv$  individuals with highest HV contribution
22:   for  $i$  in  $1, \dots, ncv$  do
23:      $CV \leftarrow CM^{(i)}$  // pick one individual from CM
24:      $hvc \leftarrow 0$ 
25:      $CV^{(temp)} \leftarrow CV$ 
26:     for  $j$  in  $1, \dots, ngroup$  do
27:        $S \leftarrow \mathcal{G}_j$  // search only on variables in group  $j$ 
28:        $D \leftarrow \text{SearchDir}(CV, \text{lbound}, \text{ubound}, S, ndir)$ 
29:       for  $k$  in  $1, \dots, noff$  do
30:          $Q \leftarrow \text{Offspring}(CV, D, \text{lbound}, \text{ubound})$ 
31:          $P_{temp} \leftarrow P \cup Q$ 
32:          $hvc_{temp} \leftarrow$  HV contribution of  $Q$  w.r.t.  $P_{temp}$ 
33:         if  $hvc_{temp} > hvc$  then
34:            $CV_S^{(temp)} \leftarrow Q$ 
35:         end if
36:       end for
37:     end for
38:      $P \leftarrow P \cup CV^{(temp)}$ 
39:      $P \leftarrow$  remove least HV contributor from  $P$ 
40:   end for
41: end while

```

Algorithm 3 $\mathcal{G}^{\text{new}} = \text{CombineGroup}(\mathcal{G}, \text{minsize})$

```

1:  $\mathcal{S} \leftarrow \emptyset$ 
2:  $\text{groupcount} \leftarrow 1$ 
3: for  $i$  in  $1, \dots, |\mathcal{G}|$  do
4:    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{G}_i$ 
5:   if  $|\mathcal{S}| \geq \text{minsize}$  OR  $i = |\mathcal{G}|$  then
6:      $\mathcal{G}_{\text{groupcount}}^{\text{new}} \leftarrow \mathcal{S}$ 
7:      $\mathcal{S} \leftarrow \emptyset$ 
8:      $\text{groupcount} = \text{groupcount} + 1$ 
9:   end if
10: end for

```

Algorithm 4 $V = \text{SearchDir}(\mathbf{x}, \text{lbound}, \text{ubound}, \mathcal{S}, \text{ndir})$

```

1:  $\mathbf{d} \leftarrow$  Vector with maximum distance from  $\mathbf{x}_S$  to  $\text{lbound}_S$  or  $\text{ubound}_S$ 
2:  $V \leftarrow$  randomly generate  $\text{ndir}$  vectors
3:  $V_{-S} \leftarrow 0$  // restrict search vector on  $\mathcal{S}$  axes
4:  $V_S^{(1)} \leftarrow \mathbf{d}$  // replace the first vector with  $\mathbf{d}$ 
5:  $V \leftarrow \text{GramSchmidt}(V^{(1)}, \dots, V^{(\text{ndir})})$  // return a matrix of size  $n \times \text{ndir}$ 

```

Algorithm 5 $V = \text{GramSchmidt}(\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(\text{ndir})})$

```

1: for  $i$  in  $2, \dots, \text{ndir}$  do
2:   for  $j$  in  $1, \dots, (i-1)$  do
3:      $\mathbf{v}^{(i)} \leftarrow \mathbf{v}^{(i)} - (\mathbf{v}^{(i)T} \mathbf{v}^{(j)}) / \|\mathbf{v}^{(j)}\|^2$ 
4:   end for
5: end for
6: for  $i$  in  $1, \dots, \text{ndir}$  do
7:    $V^{(i)} \leftarrow \mathbf{v}^{(i)} / \|\mathbf{v}^{(i)}\|$  // return a matrix of  $\text{ndir}$  vectors
8: end for

```

Algorithm 6 $\mathbf{Q} = \text{Offspring}(\mathbf{CV}, D, \text{lbound}, \text{ubound})$

```

1:  $\text{ndir} \leftarrow \text{ncol}(D)$ 
2: for  $i$  in  $1, \dots, \text{ndir}$  do
3:    $\mathbf{Q} \leftarrow \mathbf{CV} + rD^{(i)}$ , with  $r$  a random number such that  $\mathbf{Q}$  is within the box constraint,  $i$  is the column index
4: end for

```

The orthogonal sampling is more flexible as more than one search direction can be generated for each group. So, instead of line search along different directions as done in WOF, LMOEA-DS, and other similar algorithms, the orthogonal sampling allows us to search on hyperplanes in each of the subproblems. With respect to the WOF algorithm, we can think of each additional search direction as an additional weight to be optimized and it is guaranteed that the search directions are orthogonal to each other. This also means that the extent of dimensionality reduction in each group can be controlled. The search directions are generated following Algorithms 4 and 5.

In this section, two numerical experiments are conducted. The first numerical experiment is to assess the algorithm sensitivity to the number of samples used in CVA-DA. The purpose of these experiments is to check whether CVA-DA grouping is beneficial for the

optimization. The grouping itself has some associated costs and may deteriorate performance when the budget is limited. After all, the final goal is obtaining the Pareto front, not the grouping. To assess this, the groups obtained from CVA-DA are used in a CC framework and compared against the same algorithm with random grouping.

The second set of numerical experiments is focused on the EA instead of the grouping. In these experiments, we evaluate the algorithm performance with different numbers of search directions for the mirrored orthogonal sampling. The performances from this set of experiments are also compared against the performances of the WOF algorithm.

The WFG Huband et al. (2006) and UF Zhang et al. (2009) test problems are used for benchmarking. The problems are chosen because they represent problems with different diversity types (Hong et al. 2019). The UF problems also represent problems with complex Pareto set

topology. Additionally, the problems are also chosen because the performance data for WOF is available in Zille et al. (2017) for these particular problems.

5.1 Sensitivity to the Number of Samples for CVA–DA

Recall that the number of samples and the number of base points affect the accuracy of CVA–DA (see Sect. 4.2). However, these numbers also affect how many function evaluations will be used for the CVA–DA. While more samples and bases improve the grouping accuracy, more resources are also required for CVA–DA and the optimization phase will have a more limited budget. As an example, using a total number of samples $2rm^2d$, with $r = 5$, 1000-variable problems will require 90,000 function evaluations (90% of the budget) for 3-objective problems and 250,000 function evaluations (not feasible) for 5-objective problems only for classifying variables. If instead we set $r = 1$, the CVA–DA will only consume 18,000 and 50,000 function evaluations for 3- and 5-objective problems, respectively.

To check how the budget for CVA–DA affects optimization performance, the algorithms are tested using various values of r , including testing against random grouping and SMS-EMOA without grouping which can be considered as the cases where $r = 0$. In this set of experiments, only a single search direction is used. The results are shown in Table 2. It can be seen that the base SMS-EMOA never performed best in all tests which implies that grouping/decomposition is crucial and improves performance on LSMO problems.

Table 2 shows that the CVA–DA dominates the best median performances. This indicates that the grouping using CVA–DA and RDG3 is better than random grouping. However, it can also be seen that CVA–DA with $r = 5$ does not perform well on 3 objective problems. This is because the grouping consumes more than 90% of the budget so it may be better to sacrifice grouping accuracy to allow more iterations for the solver. A potential solution to reduce the CVA–DA cost is to stop the analysis of a variable when it is found that the variable should be classified as a mixed variable. In cases where there are mixed variables, this can significantly reduce the CVA–DA cost and in the worst-case scenario, where there are no mixed variables, there will be no extra cost.

Table 2 also shows that the random grouping performs best on UF1, UF2, UF6, UF7, UF8, and UF9. In UF1–UF7, all variables interact with the first variable but not with others. On UF1–UF7, both the algorithms based on CVA–DA and random grouping behave very similarly. The main difference is that in the algorithm based on random

grouping the search directions are repeatedly changed. This indicates a potential strength of dynamic grouping on these types of problems.

In UF8–UF10, only the first two variables interact with other variables. This means the problems are separable except for one or two variables so there are no disadvantages for random grouping. On these problems, both CVA–DA and random grouping behave differently. The CVA–DA detected interactions between the first two variables with each other, as well as the interactions between them with other variables. This results in the RDG3 assigning all variables into a single big group; in other words: no decomposition even after some computation budget is used for grouping. Random grouping, on the other hand, keeps on doing decomposition with no cost which leads to superior performance.

5.2 Sensitivity to the number of search directions

As mentioned before, the orthogonal sampling is more flexible because the number of search directions can be configured. However, more search directions will lead to a more complex search space and may reduce convergence rate.

In this section, the performances of CC-SMS-EMOA with different numbers of search directions are compared against the best results from the WOF. For these experiments, r is set to 1 following the best results obtained in the previous section. In addition to that, the directed sampling method from Qin et al. (2021) is also implemented on CC-SMS-EMOA to see how multiple line search fares against orthogonal search directions. The directed sampling is set similar to how it was used in Qin et al. (2021). First, $10 + m$ clusters are generated in the objective space and a representative solution is taken from each cluster as the basis for search directions. For each representative solution, two search directions are generated, one pointing to the lower bound, the other pointing to the upper bound, creating $2(10 + m)$ search directions. In each search direction, 30 points are sampled.

The data for WOF are not generated from our own tests but rather taken directly from the table in Zille et al. (2017).

The results of the experiments are presented in Table 3. These show that the proposed CC-SMS-EMOA can outperform the WOF on some test problems. Despite having less budget for the optimization due to the grouping steps, the CC-SMS-EMOA is competitive on most test problems. This further shows how good grouping can improve performance as opposed to random grouping.

Table 3 also shows an interesting pattern. It can be seen that on WFG test problems the median performances drop

Table 2 Sensitivity of CC-SMS-EMOA to the number of samples taken for grouping

2 Objectives					
Function	r = 5	r = 2	r = 1	Random	SMS-EMOA
WFG1	0.4326	0.4327	0.4407*	0.2971	0.3495
WFG2	0.9998	0.9999*	0.9999 *	0.6709	0.5061
WFG3	0.9854	0.9863	0.9865 *	0.7665	0.6521
WFG4	0.9975	0.9979 *	0.9979*	0.7375	0.7217
WFG5	0.9634*	0.9536	0.9570	0.6722	0.7084
WFG6	0.9968	0.9953	0.9975*	0.8885	0.7403
WFG7	0.9976	0.9980*	0.9507	0.6835	0.5670
WFG8	0.9021	0.9034	0.9076*	0.5158	0.5728
WFG9	0.9601*	0.95462	0.9589	0.8511	0.7244
UF1	0.7125	0.7124	0.6673	0.7839*	0.6478
UF2	0.6147	0.6148	0.6166	0.6658 *	0.5683
UF3	0.9568	0.9829*	0.9810	0.6650	0.5543
UF4	0.9056	0.9075	0.9076*	0.8568	0.7980
UF5	0.6720	0.6987*	0.6778	0.4249	0.0719
UF6	0.6789	0.6804	0.6791	0.7673*	0.6117
UF7	0.6695	0.6786	0.5892	0.7405*	0.6214
3 objectives					
WFG1	0.3516	0.4273	0.4437*	0.2670	0.34112
WFG2	0.7023	1.0146	1.0147 *	0.6343	0.56223
WFG3	0.8773	0.9497	0.9514*	0.6131	0.46927
WFG4	0.8479	0.8745	0.8746*	0.6437	0.62027
WFG5	0.8101	0.8420	0.8441*	0.6016	0.60233
WFG6	0.8426	0.8743*	0.8734	0.7854	0.63809
WFG7	0.8573	0.8754*	0.8679	0.5903	0.44978
WFG8	0.7460	0.8096	0.8118 *	0.4256	0.45603
WFG9	0.7797	0.8703*	0.8208	0.7804	0.62808
UF8	0.2795	0.1957	0.1957	0.8531*	0.51010
UF9	0.4905	0.5005	0.2814	0.5605 *	0.46815
UF10	0.0000	0.0000	0.0000	0.0000	0.00000

The medians of relative hypervolume with respect to a target hypervolume are presented. Each algorithm is repeated 20 times on each problem. Asterisks (*) indicate best performances and bold text indicates a non-significant difference compared to the best

significantly as the number of search directions increases. On the UF test problems, on the other hand, there are cases where the performances improve as the number of orthogonal search directions increases. This may be attributed to the fact that some of the UF problems are designed to have complicated Pareto set structures as opposed to the straight lines for each diversity variable in the WFG test problems. A single search direction may struggle to get close to this Pareto set. With more search directions, larger parts of the search space can be explored. The orthogonal sampling method allows for flexibility in this regard. However, it is difficult to determine the optimum number of search directions beforehand because the shape of the Pareto set is unknown before optimization. It

may be beneficial to tune/adapt the number of search directions as the search space is being explored.

Another observed result is that in most test problems in the experiments, directed sampling is outperformed by orthogonal sampling. Multiple line search is inferior to using orthogonal sampling in CC-SMS-EMOA.

6 Conclusion and future work

In this paper, an improved control variable analysis is proposed resembling the decision variable clustering and differential analysis which is referred to as CVA-DA. The method has higher accuracy than CVA, but it is also sensitive to the number of samples drawn.

Table 3 Sensitivity of CC-SMS-EMOA to the number of orthogonal search directions

2 Objective							
Function	WOF	Dir. sampling	1 direction	2 directions	5 directions	10 directions	
WFG1	0.6525	0.3810	0.4407*	0.3403	0.3227	0.3174	
WFG2	0.9834	0.7874	0.9999 *+	0.9205	0.8937	0.8809	
WFG3	0.8558	0.8264	0.9865 *+	0.95382+	0.91835+	0.90702+	
WFG4	0.9754	0.9372	0.9979 *+	0.9635	0.9532	0.9504	
WFG5	0.9465	0.8562	0.9570 *+	0.9177	0.8867	0.8514	
WFG6	0.9989	0.8200	0.9975 *	0.9949	0.9921	0.9937	
WFG7	0.9625	0.9084	0.9507 *	0.9207	0.8897	0.8793	
WFG8	0.8850	0.7298	0.9076 *+	0.8119	0.7857	0.7765	
WFG9	0.9628	0.7927	0.9589 *	0.9519	0.9337	0.9202	
UF1	0.9015	0.6382	0.6673	0.7839	0.8049	0.8218 *	
UF2	0.9320	0.5431	0.6166	0.8491	0.8681 *	0.8659	
UF3	0.9892	0.6627	0.9810 *	0.8690	0.8203	0.7942	
UF4	0.9152	0.8350	0.9076 *	0.9041	0.8916	0.8901	
UF5	0.1999	0.6999 *+	0.6778 +	0.6860 +	0.66621+	0.65929+	
UF6	0.7019	0.7217	0.6791	0.8561+	0.8812 *+	0.8800 +	
UF7	0.8990	0.5707	0.5892	0.7006	0.7117	0.7152 *	
3 Objective							
WFG1	0.6042	0.3805	0.4437 *	0.3307	0.3084	0.3152	
WFG2	0.9564	0.6715	1.0147 *+	0.8942	0.8683	0.8534	
WFG3	0.9486	0.8301	0.9514 *+	0.8987	0.8383	0.8284	
WFG4	0.8857	0.8260	0.8746 *	0.8487	0.8452	0.8437	
WFG5	0.8360	0.6961	0.8441 *+	0.8075	0.7564	0.7241	
WFG6	0.9676	0.6966	0.8734	0.8745 *	0.8729	0.8700	
WFG7	0.8568	0.8341	0.8679 *+	0.8453	0.8165	0.7989	
WFG8	0.8121	0.7189	0.8118 *+	0.7638	0.7346	0.7156	
WFG9	0.9036	0.7243	0.8208	0.8852 *	0.8413	0.8229	
UF8	0.8580	0.1948	0.1957	0.7363 *	0.6093	0.4889	
UF9	0.6946	0.2803	0.2814	0.5145 *	0.4543	0.3964	
UF10	0.8552	0.0000	0.0000	0.0000	0.0000	0.0000	

The medians of relative hypervolume with respect to a target hypervolume are presented. Each algorithm is repeated 20 times on each problem. Asterisks (*) indicate best performances and bold text indicates a non-significant difference compared to the best (excluding WOF). Additional plus (+) signs indicate that the corresponding median is better than the best median result obtained by the WOF framework

It is also shown that, in a cooperative coevolution framework, the CVA–DA combined with the RDG3 can achieve higher performance than random grouping when the variables are strongly interacting. As the computational costs for the grouping methods are low, the weighted optimization framework may also benefit from using the proposed method, as opposed to random grouping, in problems with strong variable interactions.

For orthogonal sampling, in problems with a simple Pareto set, a single search direction is sufficient. However, in problems with a more complicated Pareto set, the optimal number of search directions cannot be determined easily. An adaptive scheme can potentially be applied to

determine this. Additionally, the step size of the mutation can also be adapted to balance the exploration and exploitation of the search space.

For future work, Ma et al. (2021) mentioned that the decision variable clustering (and similarly, CVA–DA) cannot discern how much each variable affects convergence and/or diversity. It should be noted that the samples used in decision variable clustering and CVA–DA are suitable for DA and the purpose of doing DA is exactly to measure how much each variable affects the objectives. The only issue is that the DA normally can only process scalar functions. For a vectorial output, pre-processing is needed. An example of how DA is used on vectorial output

is available in Strachan et al. (2015), Monari and Strachan (2017) where principal component analysis (PCA) is used to summarize the output. By using these techniques, the sensitivity indices can be exploited, e.g., as a basis for budget allocation.

It would also be compelling to implement mirrored orthogonal sampling in other frameworks such as the LMOEA-DS. The orthogonal sampling samples the offspring on sub-hyperplanes of the original problem instead of one-dimensional search directions as in LMOEA-DS and WOF. Such experiments would allow us to determine on which problems multiple line search directions are preferred and on which problems the orthogonal sampling is performing better.

Acknowledgements Dani Irawan and Boris Naujoks acknowledge the European Commission's H2020 programme, H2020-MSCA-ITN-2016 UTOPIAE (grant agreement No. 722734). Boris Naujoks also acknowledges the DAAD (German Academic Exchange Service), Project-ID: 57515062.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Author contributions DI: Conceptualization, Methodology, Software, Analysis, Writing - original draft. BN: Conceptualization, Analysis, Resources, Funding acquisition, Review and Editing, Supervision. TB: Analysis, Resources, Review, Editing, and General Supervision. ME: Conceptualization, Analysis, Review and Editing, Supervision.

References

- Antonio LM, Coello CAC (2013) Use of cooperative coevolution for solving large scale multiobjective optimization problems. In: IEEE congress on evolutionary computation, pp. 2758–2765. <https://doi.org/10.1109/CEC.2013.6557903>
- Beume N, Naujoks B, Emmerich M (2007) SMS-EMOA: multiobjective selection based on dominated hypervolume. *Eur J Oper Res* 181(3):1653–1669
- Buzdalov M, Shalyto A (2014) A provably asymptotically fast version of the generalized Jensen algorithm for non-dominated sorting. In: Bartz-Beielstein T, Branke J, Filipič B, Smith J (eds) *Parallel problem solving from nature—PPSN XIII*. Springer, Cham, pp 528–537
- Campolongo F, Cariboni J, Saltelli A, Schoutens W (2005) Enhancing the Morris method. In: *Sensitivity analysis of model output*, pp 369–379
- Deb K (2001) *Multi objective optimization using evolutionary algorithms*. Wiley, West Sussex
- He C, Li L, Tian Y, Zhang X, Cheng R, Jin Y, Yao X (2019) Accelerating large-scale multiobjective optimization via problem reformulation. *IEEE Trans Evol Comput* 23(6):949–961
- Hong W, Tang K, Zhou A, Ishibuchi H, Yao X (2019) A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization. *IEEE Trans Evol Comput* 23(3):525–537. <https://doi.org/10.1109/TEVC.2018.2881153>
- Huband S, Hingston P, Barone L, While L (2006) A review of multiobjective test problems and a scalable test problem toolkit. *Trans Evol Comput* 10(5):477–506
- Irawan D, Naujoks B, Emmerich M (2020) Cooperative-coevolution-CMA-ES with two-stage grouping. In: *2020 IEEE congress on evolutionary computation (CEC)*, pp 1–80 <https://doi.org/10.1109/CEC48606.2020.9185616>
- Li X, Tang K, Omidvar MN, Yang Z, Qin K, China H (2013) Benchmark functions for the CEC2013 special session and competition on large-scale global optimization. *Gene* 7(33):8
- Liu R, Liu J, Li Y, Liu J (2020) A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems. *Swarm Evol Comput* 55:100684
- Liu R, Ren R, Liu J, Liu J (2020) A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems. *Appl Soft Comput* 89:106120
- Ma X, Liu F, Qi Y, Wang X, Li L, Jiao L, Yin M, Gong M (2016) A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *Trans Evol Comput* 20(2):275–298
- Ma L, Huang M, Yang S, Wang R, Wang X (2021) An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization. *IEEE Trans Cybern*. <https://doi.org/10.1109/TCYB.2020.3041212>
- Mahdavi S, Rahnamayan S, Shiri ME (2017) Multilevel framework for large-scale global optimization. *Soft Comput* 21(14):4111–4140
- Monari F, Strachan P (2017) Characterization of an airflow network model by sensitivity analysis: parameter screening, fixing, prioritizing and mapping. *J Build Perform Simul* 10(1):17–36
- Morris MD (1991) Factorial sampling plans for preliminary computational experiments. *Technometrics* 33(2):161–174
- Omidvar MN, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. *Trans Evol Comput* 18(3):378–393
- Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) DG2: a faster and more accurate differential grouping for large-scale black-box optimization. *Trans Evol Comput* 21(6):929–942
- Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: *Parallel problem solving from nature*. Springer, pp. 249–257
- Qin S, Sun C, Jin Y, Tan Y, Fieldsend J (2021) Large-scale evolutionary multi-objective optimization assisted by directed sampling. *IEEE Trans Evol Comput*. <https://doi.org/10.1109/TEVC.2021.3063606>
- Raykov YP, Boukouvalas A, Baig F, Little MA (2016) What to do when k-means clustering fails: a simple yet principled alternative algorithm. *PLoS ONE* 11:1–28
- Roy PC, Islam MM, Deb K (2016) Best order sort: A new algorithm to non-dominated sorting for evolutionary multi-objective optimization. In: *Proceedings of the 2016 on genetic and evolutionary computation conference companion. GECCO '16 Companion*. Association for Computing Machinery, New York, NY, USA, pp 1113–1120. <https://doi.org/10.1145/2908961.2931684>
- Sander F, Zille H, Mostaghim S (2018) Transfer strategies from single- to multi-objective grouping mechanisms. In: *Proceedings*

- of the genetic and evolutionary computation conference. GECCO '18. Association for Computing Machinery, New York, NY, USA, pp 729–736. <https://doi.org/10.1145/3205455.3205491>
- Strachan P, Monari F, Kersken M, Heusler I (2015) IEA annex 58: Full-scale empirical validation of detailed thermal simulation programs. *Energy Procedia* 78:3288–3293. 6th International Building Physics Conference, IBPC
- Sun Y, Li X, Ernst A, Omidvar MN (2019) Decomposition for large-scale optimization problems with overlapping components. In: IEEE congress on evolutionary computation, pp 326–333. <https://doi.org/10.1109/CEC.2019.8790204>
- Trunfio GA (2015) A cooperative coevolutionary differential evolution algorithm with adaptive subcomponents. *Procedia Comput Sci* 51:834–844. International Conference On Computational Science, ICCS 2015
- Wang H, Emmerich M, Bäck T (2019) Mirrored orthogonal sampling for covariance matrix adaptation evolution strategies. *Evol Comput* 27(4):699–725
- Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. *Inf Sci* 178(15):2985–2999
- Yue HD, Sun Y (2021) Cooperative coevolution with two-stage decomposition for large-scale global optimization problems. *Discrete Dyn Nat Soc* 2021:2653807. <https://doi.org/10.1155/2021/2653807>
- Zhang X, Tian Y, Cheng R, Jin Y (2018) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. *IEEE Trans Evol Comput* 22(1):97–112. <https://doi.org/10.1109/TEVC.2016.2600642>
- Zhang J, Xing L, Peng G, Yao F, Chen C (2019) A large-scale multiobjective satellite data transmission scheduling algorithm based on SVM + NSGA-II. *Swarm Evol Comput* 50:100560
- Zhang Q, Zhou A, Zhao SZ, Suganthan PN, Liu W, Tiwari S (2009) Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical report, Technical Report CES-887, University of Essex and Nanyang Technological University. <https://doi.org/10.1007/s13042-019-01030-4>
- Zhenyu Yang, Ke Tang, Xin Yao: Multilevel cooperative coevolution for large scale optimization. In: World congress on computational intelligence. IEEE, pp 1663–1670 (2008). <https://doi.org/10.1109/CEC.2008.4631014>
- Zille H, Ishibuchi H, Mostaghim S, Nojima Y (2017) A framework for large-scale multiobjective optimization based on problem transformation. *Trans Evol Comput* 22(2):260–275

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.