Linear Search Mechanism for Multi- and Many-Objective Optimisation

Heiner Zille[0000−0002−7262−9487] and Sanaz Mostaghim[0000−0002−9917−5227]

Institute for Intelligent Cooperative Systems, Otto von Guericke University, Magdeburg, Germany
{heiner.zille,sanaz.mostaghim}@ovgu.de
http://www.ci.ovgu.de

Abstract. This article proposes a search mechanism based on linear combinations of population members to increase the solution quality of multi-objective and many-objective optimisation algorithms. Our approach makes use of the inherent knowledge in the solution population at a given time step, and forms new solutions through linear combinations of the existing ones. A population of coefficient vectors is formed and optimised by a metaheuristic to explore and exploit promising areas of the search space. In addition, our proposed method provides a reduction of dimensionality for large search spaces. The concept is formally introduced and implemented into a generic algorithm structure to be used in arbitrary metaheuristics. The experimental evaluation uses four multi- and many-objective algorithms (NSGA-II, GDE3, NSGA-III and RVEA) and is performed on a total of 60 test instances from three benchmark families with 2 to 5 objective functions and 30 to 514 decision variables. The results indicate that the performance of existing methods can be significantly improved by the proposed search strategy, especially in high-dimensional search spaces and for many-objective problems.

Keywords: Multi-objective Optimisation · Many-objective Optimisation · Large-scale Optimisation · Evolutionary Algorithm · Exploration · Linear Combination · Dimensionality Reduction.

1 Introduction

The search for a well-spread non-dominated front in multi- and many-objective optimisation is still an ongoing challenge. This is especially true in large-scale optimisation which contains a very large number of decision variables or many objective functions. Previous methods try to balance the trade-off between convergence and diversity in different ways. The research in the last years has led to a variety of many-objective optimisation methods (e.g. [6, 10, 2]), as well as a number of methods that can deal with hundreds or thousands of decision variables (e.g. [20, 17, 11, 18]). Concepts that can be found in this area are dimensionality reduction (e.g. [14]), variable interaction analyses or the division of design variables into convergence-related and diversity-related parameters (e.g. [17, 11]), some of which involve increased computational costs.

In this work, we propose a method to increase exploration in multi- and many-objective optimisation by searching in subspaces defined by the current populations' design variables. The approach is based on the assumption that the evolutionary process finds promising areas of the search space (i.e. areas where good solutions are located) and adjusts the variables in the population accordingly to cover and explore these areas. Once certain promising results have been found, a recombination of these solutions through linear combinations is subject to optimisation in order to create new solutions which benefit from the inherent information in the population. Our newly proposed exploration method can easily be included into any metaheuristic optimisation algorithm and can also lead to a dimensionality reduction without the need for dividing variables into subcomponents. In this work, the mathematical concept is introduced and analysed, and an experimental evaluation shows its benefits when embedded into multi- and many-objective algorithms. The equipped algorithms are tested on a total of 60 different benchmark function instances from the literature with 2 to 5 objective functions and 30 to 514 decision variables.

The remainder of this article is structured as follows. In Section 2 the basic principles of multi-objective optimisation are outlined briefly and a short overview about related work on multi-objective and large-scale approaches is given. In Section 3 the proposed linear-combination approach is introduced. The mathematical concept is explained first before describing the inclusion of the concept into existing algorithms. The experimental evaluation in Section 4 equips a number of well-known metaheuristics with the proposed exploration method and compares their performance on a variety of benchmark functions. Finally, a summary and outlook on future work directions is given in Section 5.

2 Multi-objective Optimisation and Related Work

Problems in nature and science often contain multiple conflicting goals which need to be optimised simultaneously. These problems are called multi-objective problems (MOPs) and can be formulated as:

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

where $m \geq 2$. This kind of MOP maps the decision space $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n | \, \boldsymbol{g}(\boldsymbol{x}) \leq 0\}$ of dimension n to the objective space of dimension m. In most problems, some of the constraints define a domain for each variable with lower and upper bounds, i.e. $x_i \in [x_{i,min}, x_{i,max}]$, $i = 1, .., n$. For such problems, a single optimal solution can often not be determined, since there is usually a trade-off between the objective functions. Modern problem solving methods instead concentrate on finding an approximation of a Pareto-optimal solution set [4, 6, 9].

Two key challenges in finding such a set of solutions are convergence and diversity of the solution set. Convergence refers to the search for better, non-dominated solutions, which improve all objective functions from the current solution set, and therefore bring the whole set closer to the true Pareto-optimal

solutions. Diversity on the other hand is necessary to obtain a widely spread set of solutions. The output of a metaheuristic algorithm should provide a diverse set of different solutions which represent different trade-offs among the objective functions and cover the whole Pareto-optimal front as good as possible. Finding a non-dominated set of solutions that is as close to the true Pareto-set and as diverse as possible is an ongoing challenge, especially when many-objective and large-scale problems are involved [20, 17, 11, 18].

In the area of large-scale (i.e. many-variable) optimisation, the topic of reducing the dimensionality of a problem is often of importance. Concepts like Cooperative Coevolution [1] aim to optimise smaller subspaces of the n-dimensional search space by dividing the variables into groups or subcomponents based on different criteria. Approaches to achieve a better balance between convergence and diversity have been used e.g. in [17, 11]. These works carry out an analysis to identify variables which influence the diversity of the solution set before starting the optimisation process. In addition, both methods utilize an interaction detection to from groups of variables. A major drawback of these and similar approaches is that the analysis of variables and formation of variable groups requires an additional and very large computational budget for this preprocessing step, while the actual benefit compared to a random assignment of variables to the groups or less expensive methods is not always guaranteed [13]. Another method called WOF [19, 20] shows a superior convergence behaviour in large-scale problems with up to thousands of variables [21]. WOF aims to balance diversity and convergence through the selection of certain solutions from the population, which are used in a fast-converging transformation and optimisation step of the algorithm.

In the area of many-objective optimisation, a variety of algorithms has been developed in recent years, among them many who adapt the concept of reference vectors like NSGA-III [6] , MOEA/DD [10] or RVEA [2]. Reference vectors or directions are a common concept that is used to solve the problem of decreasing selection pressure when Pareto-dominance-based approaches are used for many-objective problems. Such methods have increased the capabilities of metaheuristics to solve many-objective problems. Due to that, an area that might draw increased focus in the future is solving problems with many objectives and a large number of decision variables at the same time, while keeping computational budget as low as possible. This work therefore aims to propose a mechanism that can be used to reduce the dimensionality of such problems and by that improve the solution quality of existing many-objective methods.

3 Proposed Approach

In this section, we propose a search strategy that can be used to enhance exploration of the search space and at the same time reduce the dimensionality of a problem without using variable groups. In metaheuristic evolutionary optimisation, one general assumption is that the encoding of the problem ensures that promising solutions can be generated from a combinations of other good

solutions. By extension, Pareto-optimal solutions might share certain characteristics, i.e. decision variable values, which might be similar throughout the whole Pareto-optimal set (as for instance the convergence-related variables of common benchmark families [3, 8]), and which might be approximated by an optimiser. The main goal of this approach is to exploit this information that is inherent in the population of an evolutionary algorithm at a given time, i.e. the information which (sub-)vector-space of the n-dimensional search space contains the (at that point) best or most promising solutions. Based on this, a search inside this subspace in conducted. This concept is also related to that of "innovization" from the literature ([5, 7]), which aims to extract information or design principles from the outcome or during the process of optimisation. In the following, the concept of the proposed search strategy is explained.

3.1 Concept

Suppose we have an optimisation problem with n real-valued decision variables and m objectives as given in Equation 1. Let the population of an algorithm be P and its size be $s := |P|$. At each given time of the optimisation process the population consists of s solution vectors each of dimensionality n: $P = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(s)}\}$. Each solution is a vector $\in \mathbb{R}^n$:

$$\boldsymbol{x}^{(i)} = (x_1^{(i)}\ x_2^{(i)}\ ...\ x_n^{(i)}) \tag{2}$$

and the set of solutions P defines a vector (sub)space. The dimensionality of this subspace is given by the rank of the matrix of the spanning vectors. We therefore compose the matrix $\hat{X} \in \mathbb{R}^{s \times n}$ which contains in each row one solution of the population.

$$\hat{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(s)} & x_2^{(s)} & \cdots & x_n^{(s)} \end{pmatrix} \tag{3}$$

An evolutionary algorithm (EA) combines the solutions in the current population by using crossover operators. However, instead of classical crossover methods, it is also possible to combine the existing solutions linearly. This can be done through convex, conical or arbitrary linear combinations. In the following we focus on general linear combinations as they include the convex and conical combinations as subsets. A linear combination of the solutions in the population is defined as follows:

$$\boldsymbol{x}' = \boldsymbol{y}\hat{X} = y_1\boldsymbol{x}^{(1)} + y_2\boldsymbol{x}^{(2)} + ... + y_s\boldsymbol{x}^{(s)} \tag{4}$$

where $\boldsymbol{y}$ is the vector of coefficients of the combination:

$$\boldsymbol{y} = (y_1\ y_2\ ...\ y_s) \tag{5}$$

With this concept it is possible to search the subspace spanned by the s vectors in the population, and thereby find improved solutions from combinations

of the existing ones. The actual dimension of this subspace is defined by the rank of the matrix $\hat{X}$ which is bounded between $1 \leq rank(\hat{X}) \leq min\{n, s\}$.

In a way, this method can be seen as a s-parent crossover method. In contrast to a crossover, in our proposed method the parameters of the vector $\boldsymbol{y}$ are subject to an optimisation process. While a multi-parent crossover would produce a random combination and let the evolutionary process judge whether this was a good one, the proposed procedure uses an evolutionary process to find "optimal" combinations.

In order to find a good linear combination, the values of the vector y are optimised instead of the original variables. As a consequence, this newly formed optimisation problem has only s decision variables compared to the original one with n variables. This means, in case a problem with $n = 30$ variables is optimised with a population size of $s = 100$, there might be redundancy in the newly formed linear-combination-problem, and the algorithm now searches in 100 dimensions, even though the actual space in which the solutions are created is still 30-dimensional, and some of the vectors of the linear combinations are not independent in this case. However, the situation differs when applied to a high-dimensional problem with for instance $n = 500$ variables. The s solutions can at most define a s-dimensional subspace. If all solutions are randomly created in the beginning, it is not guaranteed that good solutions actually lie in the defined subspace. However, when the algorithm is allowed a certain progress to find a preliminary approximation of the optimal areas, we can assume that promising parameter combinations might have been found already, and the spanned subspace might contain additional good solutions. In that case, optimising the linear-combination-solutions can also be regarded as a dimensionality reduction technique, as it enables the algorithm to search in a 100-dimensional subspace instead of the 500-dimensional original search space. This makes the method not only promising for multi- and many-objective problems, but also for the area of large-scale optimisation.

3.2 Inclusion into other Algorithms

The proposed concept can be used inside arbitrary metaheuristic optimisation algorithms. To do so, we define a population Q of $\boldsymbol{y}$-vectors, where each vector in the population defines one linear combination of the members of P as described above. By this, we can use any metaheuristic optimiser on this newly formed population to find suitable linear combinations of the underlying original solutions in the population P. Since the optimisation of the population Q relies on the assumption, see above, that the population P defines a promising subspace of Ω, the proposed method is included into other metaheuristics as an additional search step. In particular, we apply the original (arbitrary) metaheuristic in turns with the proposed linear-combination-search. As a further step to concentrate on promising solutions, the linear combinations are only performed on the non-dominated solutions in the population.

Let $\hat{X}$ be the matrix of the decision variable values of all non-dominated solutions in P as seen above, where each row in $\hat{X}$ corresponds to one non-

dominated solution in P. As a result, $\hat{X}$ is an $s' \times n$ matrix, where s' is the number of non-dominated solutions. In the same way, let $\hat{Y}$ be the matrix of the decision variable values (i.e. coefficients of linear combinations) of the solutions in Q. The population size of Q is t, therefore $\hat{Y} \in \mathbb{R}^{t \times s'}$. The original objective function evaluation can be applied to the new population by simply multiplying $\hat{X}$ with $\hat{Y}$ and computing $\boldsymbol{f}(\hat{Y}\hat{X})$, i.e. applying $\boldsymbol{f}$ to each row in $\hat{Y}\hat{X}$. For practical reasons and to limit the search space of the newly found problem, the variables y_i are also equipped with lower and upper bounds, i.e. $y_i \in [y_{i,min}, y_{i,max}]$, $i = 1, .., s'$.

The outline of the resulting algorithm looks as follows:

1. Optimise the population P with any multi-objective method for a specified time.
2. Use the first non-dominated front of the current population P to build the matrix $\hat{X}$ out of its decision variables' values.
3. Create a random population Q of linear-combination-vectors.
4. Optimise Q for a certain time using an arbitrary optimisation method. Store all evaluated solutions in an Archive A.
5. Merge the population P with A and proceed with the normal optimisation process (Step 1).

4 Evaluation

To evaluate the proposed method, we have included it into several well-known optimisation algorithms from the areas of multi- and many-objective optimisation. These algorithms are NSGA-II [4], GDE3 [9] as representatives of traditional evolutionary methods, both classical and differential evolution, and NSGA-III [6] and RVEA [2] to represent many-objective methods. The aim of the experiments is not to show the superiority of one of these methods over one another, but rather to show that the proposed exploration method has a positive influence when applied to existing algorithms. Due to space limitations, an inclusion into dedicated large-methods like LMEA [17], MOEA/DVA [11] or WOF [20], and the analysis of this methods' capabilities as a dimensionality reduction mechanism, is subject to future work. Each of the four used algorithms has been equipped with the proposed method by applying in terms 100 generations of the original algorithm and after that 100 generations of the search in the formed subspace as described above. The created solutions are then merged back into the original population using the usual selection method of the respective algorithm. For the optimisation of the population Q, the NSGA-II algorithm is used in all cases. This is done so that all algorithms use the same exploration mechanism for searching the formed subspace. Future research might deal with different mechanisms in this regard, as the NSGA-II mechanism might not be the optimal choice for many-objective problems. This procedure is repeated until the maximum amount of function evaluations is reached. The modified versions of the algorithms are denoted with an "x" in front of their names, i.e. xNSGA-II, xGDE3, xNSGA-III and xRVEA. To test the performance, we use a total of

60 test problem instances from three common benchmark families with 2 to 5 objectives and 30 to 514 decision variables. The used problems are as follows:

- Six problems from the LSMOP (large-scale multi- and many-objective test problems) family [3]: LSMOP1-6. Each of them is tested with 2, 3, 4 and 5 objective functions, resulting in 206, 307, 413 and 514 decision variables.
- Six problems from the popular WFG family [8]: WFG2-5, WFG7 and WFG8. All of them are tested with 2 and 3 objectives, in combination with both 40 and 400 decision variables. The WFG problems were chosen by an analysis done in [11], where WFG2 and 3 represent problems with a sparse number of interacting variables, WFG4 and 5 have no interactions and WFG7 and 8 have a high number of interacting variables.
- Six problems from the CEC 2009 unconstrained benchmarks: UF1-3 are 2-objective problems, UF8-10 are 3-objective problems. All of them are tested with 30 and 300 variables.

For implementation, the PlatEMO framework [15] version 1.5 is used. For each experiment we perform 31 independent runs and report the median and interquartile range (IQR) values of the relative hypervolume (HV) indicator [16]. The relative HV is the hypervolume obtained by a solution set in relation to the hypervolume obtained by a sample of the Pareto-front of the problem, consisting of $10,000$ solutions as provided by the PlatEMO framework. The used reference point for the indicator is obtained by using the nadir point of our Pareto-front sample (i.e. the point in the objective space containing the worst value in each dimension throughout the sample) and multiply it by 2.0 in each dimension. Statistical significance is tested using a two-sided Mann-Whitney-U Test with the null hypothesis that the tested samples have equal medians. Statistical significance is assumed for a value of $p < 0.01$.

4.1 Parameter Settings

The maximum number of function evaluations for all algorithms and problem instances is set to 100,000. The number of position-related variables for the WFG problems has been set to $n/4$ and the parameter n_k in the LSMOP benchmarks was set to 5. The population size is set to 40 in all instances of NSGA-II and GDE3. The population sizes of NSGA-III and RVEA are set to 40, 36, 35 and 40 for $m = 2, 3, 4$ and 5 objectives respectively, due to the uniform generation of reference vectors. All algorithms use polynomial mutation with a distribution index of 20.0 and a probability of $1/n$. All algorithms, except GDE3, use the simulated binary crossover with a distribution index of 20.0 and a probability of 1.0. In GDE3 are $CR = 1$ and $F = 0.5$. In RVEA, $\alpha = 2$ and $f_r = 0.1$ as in the original work. The bounds of the coefficients for the linear combination are set to $y_{i,min} = -10.0$, and $y_{i,max} = 10.0$.

4.2 Results

The results of the experiments are shown in Tables 1 and 2, where each algorithm is compared with its respective linear-combination-enhanced counterpart.

Table 1. Obtained median and IQR values of the relative Hypervolume for NSGA-II and xNSGA-II as well as GDE3 and xGDE3. An asterisk in the column of an x-Algorithm indicates statistical significance to the respective original version of that algorithm. Best performances are marked in bold and shaded where significant.

	m	n	NSGA-II	xNSGA-II	GDE3	xGDE3
LSMOP1	2	206	0.75134 (1.13E-1)	**0.85510** * (9.07E-2)	0.56263 (4.52E-2)	**0.86759** * (2.47E-2)
LSMOP2		206	0.94481 (4.97E-3)	**0.96903** * (6.71E-3)	0.94321 (2.31E-3)	**0.98270** * (1.41E-3)
LSMOP3		206	— (—)	**0.23474** * (1.38E-1)	— (—)	**0.03916** * (3.23E-3)
LSMOP4		206	0.90440 (8.53E-3)	**0.94727** * (4.55E-3)	0.94008 (9.01E-3)	**0.96784** * (6.26E-3)
LSMOP5		206	**0.79282** (6.00E-2)	0.62224 * (—)	0.30990 (9.27E-2)	**0.81408** * (3.48E-2)
LSMOP6		206	0.46222 (1.79E-2)	**0.57904** * (5.69E-3)	0.50444 (3.28E-2)	**0.58816** * (1.95E-2)
LSMOP1	3	307	— (—)	**0.80958** * (2.78E-2)	— (—)	**0.71837** * (2.01E-2)
LSMOP2		307	0.97074 (2.47E-3)	0.97026 (2.91E-3)	0.97153 (1.59E-3)	**0.97553** * (1.70E-3)
LSMOP3		307	— (—)	**0.51111** * (—)	— (—)	**0.51111** * (—)
LSMOP4		307	0.90478 (7.82E-3)	**0.92602** * (4.70E-3)	0.89693 (6.83E-3)	**0.93835** * (3.50E-3)
LSMOP5		307	— (—)	**0.85655** * (9.56E-4)	— (—)	**0.87305** * (1.33E-2)
LSMOP6		307	— (—)	**0.29197** * (6.70E-3)	— (—)	**0.26725** * (6.36E-3)
LSMOP1	4	413	— (—)	**0.76046** * (1.25E-2)	— (—)	**0.66323** * (1.91E-2)
LSMOP2		413	0.98192 (1.77E-3)	**0.98272** * (1.29E-3)	0.98080 (1.59E-3)	**0.98570** * (1.03E-3)
LSMOP3		413	— (—)	**0.02216** * (9.49E-4)	— (—)	**0.02356** * (3.31E-4)
LSMOP4		413	0.96053 (2.76E-3)	0.96187 (2.80E-3)	0.95663 (3.94E-3)	**0.96756** * (2.80E-3)
LSMOP5		413	— (—)	**0.91945** * (4.68E-2)	— (—)	**0.93033** * (5.17E-3)
LSMOP6		413	0.33750 (3.50E-3)	**0.66594** * (2.55E-2)	0.33652 (5.55E-3)	**0.67133** * (5.52E-2)
LSMOP1	5	514	— (—)	**0.63753** * (1.49E-2)	0.50034 (—)	**0.63655** * (2.09E-2)
LSMOP2		514	0.99283 (7.79E-4)	0.99313 (5.96E-4)	0.99323 (4.98E-4)	**0.99416** * (5.24E-4)
LSMOP3		514	— (—)	**0.50034** * (—)	— (5.00E-1)	**0.50034** * (—)
LSMOP4		514	0.96323 (2.79E-3)	**0.97878** * (1.84E-3)	0.96271 (3.12E-3)	**0.98035** * (2.37E-3)
LSMOP5		514	— (—)	**0.90318** * (5.62E-2)	— (—)	**0.69514** * (3.07E-1)
LSMOP6		514	— (—)	**0.21220** * (1.02E-1)	— (—)	**0.26520** * (1.19E-1)
UF1	2	30	0.91887 (4.11E-2)	**0.96229** * (2.89E-2)	0.97379 (1.39E-2)	0.96932 (1.53E-2)
UF2		30	0.95267 (3.99E-2)	**0.98183** * (8.40E-3)	**0.97989** (9.59E-3)	0.97173 * (8.26E-3)
UF3		30	0.71558 (5.38E-2)	**0.95430** * (6.00E-3)	**0.96908** (1.45E-2)	0.95301 * (1.46E-2)
UF1		300	0.90110 (2.48E-2)	0.90481 (6.03E-2)	**0.44383** (9.57E-2)	0.29760 * (4.25E-2)
UF2		300	0.89138 (2.24E-2)	0.88332 (1.39E-2)	0.86877 (5.74E-3)	**0.88162** * (4.62E-3)
UF3		300	0.73023 (8.34E-3)	**0.95459** * (5.81E-3)	0.73585 (1.42E-2)	**0.94991** * (1.92E-3)
UF8	3	30	0.80694 (3.35E-2)	**0.84659** * (9.42E-2)	0.24162 (2.42E-1)	**0.64186** * (6.77E-2)
UF9		30	0.72903 (9.67E-2)	**0.82914** * (8.45E-2)	0.31863 (1.46E-1)	**0.59303** * (5.85E-2)
UF10		30	0.38003 (1.29E-1)	**0.82597** * (1.22E-2)	— (—)	**0.54190** * (3.09E-2)
UF8		300	0.79119 (2.05E-2)	**0.85501** * (9.36E-4)	0.65592 (4.89E-2)	**0.82907** * (8.35E-3)
UF9		300	0.63539 (2.04E-2)	0.62206 (2.44E-2)	0.50443 (2.73E-2)	**0.62197** * (1.26E-2)
UF10		300	0.03827 (5.22E-2)	**0.84973** * (2.89E-3)	— (—)	**0.73813** * (8.86E-2)
WFG2	2	41	0.85891 (7.60E-3)	**0.97933** * (1.27E-2)	0.97452 (5.03E-2)	0.97187 (2.02E-2)
WFG3		41	0.84987 (2.91E-3)	0.84869 (6.62E-3)	**0.84444** (1.00E-2)	0.83283 * (7.49E-3)
WFG4		40	**0.99235** (1.06E-3)	0.99019 * (2.60E-3)	0.94351 (6.56E-3)	0.94394 (9.17E-3)
WFG5		40	**0.97948** (2.92E-3)	0.97841 * (2.40E-3)	**0.95348** (1.31E-2)	0.93875 * (2.60E-2)
WFG7		40	**0.99345** (1.02E-3)	0.99262 * (8.23E-4)	0.96279 (3.02E-2)	0.95901 (1.44E-2)
WFG8		40	0.89218 (5.87E-3)	**0.89593** * (6.48E-3)	**0.85652** (1.44E-2)	0.83074 * (1.64E-2)
WFG2	3	401	0.72887 (3.64E-2)	**0.86591** * (2.49E-3)	0.76243 (6.42E-3)	**0.87080** * (2.68E-3)
WFG3		401	0.63592 (7.54E-3)	**0.78669** * (7.04E-2)	0.66454 (1.52E-2)	**0.73419** * (6.82E-3)
WFG4		400	0.66487 (1.36E-2)	**0.83624** * (1.42E-2)	0.79988 (2.20E-2)	**0.82247** * (1.51E-2)
WFG5		400	0.64103 (1.90E-2)	**0.85848** * (1.42E-2)	0.82362 (6.03E-3)	**0.85548** * (8.87E-3)
WFG7		400	0.67641 (1.19E-2)	**0.86891** * (2.39E-2)	0.74902 (8.54E-3)	**0.78553** * (9.29E-3)
WFG8		400	0.57725 (1.06E-2)	**0.78999** * (1.48E-2)	0.57692 (2.33E-2)	**0.78252** * (1.79E-2)
WFG2	4	40	0.90095 (9.05E-3)	**0.97075** * (8.38E-3)	**0.94492** (1.92E-2)	0.93262 * (3.45E-2)
WFG3		40	0.93396 (2.31E-2)	0.93407 (2.58E-2)	0.84310 (3.27E-2)	**0.86879** * (4.69E-2)
WFG4		40	**0.96941** (3.62E-3)	0.95785 * (6.89E-3)	**0.88129** (2.56E-2)	0.85787 * (2.53E-2)
WFG5		40	**0.94920** (4.28E-3)	0.94571 * (2.34E-3)	0.88469 (2.12E-2)	0.88316 (2.18E-2)
WFG7		40	0.96833 (2.17E-2)	0.97117 (3.13E-2)	0.86183 (3.23E-2)	0.86343 (2.79E-2)
WFG8		40	**0.92742** (6.25E-3)	0.91145 * (8.30E-3)	0.80365 (1.82E-2)	0.80627 (1.82E-2)
WFG2	5	400	0.68917 (9.54E-3)	**0.85622** * (5.57E-3)	0.69053 (5.17E-3)	**0.84083** * (5.10E-3)
WFG3		400	0.57700 (2.56E-2)	**0.79436** * (4.23E-2)	0.57939 (1.48E-2)	**0.67991** * (4.62E-3)
WFG4		400	0.59041 (1.47E-2)	**0.71919** * (6.97E-2)	0.70378 (1.63E-2)	**0.71675** * (2.00E-2)
WFG5		400	0.55374 (1.23E-2)	**0.75534** * (4.74E-2)	0.69452 (1.47E-2)	**0.72009** * (1.29E-2)
WFG7		400	0.59516 (1.35E-2)	**0.71240** * (8.34E-3)	0.68170 (9.85E-3)	**0.69439** * (9.10E-3)
WFG8		400	0.52326 (1.77E-2)	**0.72452** * (1.80E-2)	0.59209 (1.95E-2)	**0.72812** * (1.85E-2)

Overall, the proposed method is beneficial for the performance of all algorithms in most problem instances.

First, we take a look at the two traditional evolutionary algorithms. The xNSGA-II performs significantly better (based on the used Mann-Whitney-U

Table 2. Obtained median and IQR values of the relative Hypervolume for NSGA-III and xNSGA-III as well as RVEA and xRVEA. An asterisk in the column of an x-Algorithm indicates statistical significance to the respective original version of that algorithm. Best performances are marked in bold and shaded where significant.

	m	n	NSGA-III	xNSGA-III	RVEA	xRVEA
LSMOP1	2	206	0.64220 (1.53E-1)	**0.79579** * (1.15E-1)	0.06873 (5.66E-1)	**0.83933** * (1.61E-2)
LSMOP2		206	0.95088 (2.79E-3)	**0.98200** * (4.04E-3)	0.94387 (4.98E-3)	**0.97949** * (2.83E-3)
LSMOP3		206	— (—)	**0.08521** * (9.18E-2)	**0.56963** (9.96E-3)	0.01974 * (3.53E-4)
LSMOP4		206	0.92227 (5.51E-3)	**0.95692** * (3.62E-3)	0.89610 (5.73E-3)	**0.95196** * (6.33E-3)
LSMOP5		206	**0.80093** (5.06E-2)	0.62224 * (—)	— (—)	**0.62224** * (7.05E-4)
LSMOP6		206	0.45949 (8.70E-3)	**0.57262** * (7.48E-3)	0.49474 (5.00E-2)	**0.53730** * (2.34E-1)
LSMOP1	3	307	0.07558 (1.15E-1)	**0.82378** * (2.30E-2)	0.53715 (5.74E-2)	**0.76242** * (1.24E-2)
LSMOP2		307	0.97954 (4.31E-4)	**0.98317** * (5.45E-4)	0.97803 (5.73E-4)	**0.98107** * (7.71E-4)
LSMOP3		307	— (—)	**0.51112** * (1.37E-2)	— (—)	**0.51081** * (5.01E-4)
LSMOP4		307	0.93995 (2.68E-3)	**0.96222** * (1.53E-3)	0.93470 (3.10E-3)	**0.96314** * (2.83E-3)
LSMOP5		307	— (—)	**0.85486** * (9.61E-4)	0.53581 (6.39E-6)	**0.91034** * (2.29E-2)
LSMOP6		307	— (—)	**0.28687** * (4.61E-3)	— (—)	**0.16058** * (1.06E-1)
LSMOP1	4	413	— (—)	**0.79609** * (2.13E-2)	0.56234 (1.34E-1)	**0.76333** * (1.34E-2)
LSMOP2		413	0.98752 (2.75E-4)	**0.99087** * (3.96E-4)	0.98672 (4.39E-4)	**0.99004** * (7.35E-4)
LSMOP3		413	— (—)	**0.02076** * (1.47E-3)	0.02149 (3.03E-1)	0.02041 (2.93E-3)
LSMOP4		413	0.97775 (1.31E-3)	**0.98127** * (1.63E-3)	0.97015 (3.45E-3)	**0.98206** * (1.20E-3)
LSMOP5		413	— (—)	**0.93971** * (8.78E-4)	0.51144 (3.22E-4)	**0.92817** * (5.73E-3)
LSMOP6		413	0.36714 (7.75E-3)	**0.70352** * (8.04E-3)	0.40710 (2.81E-2)	**0.74121** * (8.87E-2)
LSMOP1	5	514	— (—)	**0.69758** * (4.51E-2)	0.67646 (1.28E-1)	**0.74755** * (1.11E-2)
LSMOP2		514	0.99601 (8.07E-5)	**0.99644** * (6.37E-5)	0.99566 (2.15E-3)	**0.99625** * (1.14E-4)
LSMOP3		514	— (—)	**0.50033** * (2.10E-5)	— (7.30E-3)	**0.50033** * (3.31E-2)
LSMOP4		514	0.98389 (1.12E-3)	**0.98972** * (6.96E-4)	0.98235 (2.83E-3)	**0.98890** * (1.14E-3)
LSMOP5		514	— (—)	**0.98680** * (1.82E-3)	0.50400 (5.49E-6)	**0.98609** * (2.35E-3)
LSMOP6		514	— (—)	**0.53851** * (1.35E-2)	0.19527 (2.39E-1)	**0.31759** * (1.33E-1)
UF1	2	30	0.90857 (5.92E-2)	**0.97172** * (2.62E-2)	0.86021 (7.86E-2)	**0.94097** * (1.38E-2)
UF2		30	0.95574 (3.03E-2)	**0.98049** * (1.00E-2)	0.95095 (3.30E-2)	0.95587 (1.82E-2)
UF3		30	0.70910 (4.33E-2)	**0.94851** * (1.07E-2)	0.69021 (1.81E-2)	**0.90003** * (2.32E-2)
UF1		300	0.88442 (6.09E-2)	0.88944 (4.52E-2)	**0.70355** (5.68E-2)	0.65673 * (4.88E-2)
UF2		300	0.88425 (1.64E-2)	0.88747 (6.78E-3)	0.85611 (9.08E-3)	**0.87460** * (4.22E-3)
UF3		300	0.70306 (1.05E-2)	**0.96051** * (1.68E-3)	0.69537 (8.73E-3)	**0.95204** * (4.75E-3)
UF8	3	30	0.84654 (6.85E-3)	**0.85433** * (1.79E-3)	0.84549 (1.19E-3)	0.84531 (1.13E-1)
UF9		30	0.72999 (3.78E-2)	**0.86360** * (7.90E-2)	0.68531 (2.59E-2)	**0.86237** * (1.95E-1)
UF10		30	0.46645 (2.00E-1)	**0.85371** * (1.78E-3)	0.44133 (8.21E-2)	**0.84551** * (2.97E-4)
UF8		300	0.82401 (6.90E-3)	**0.85408** * (1.07E-3)	0.75309 (2.85E-2)	**0.84802** * (7.03E-4)
UF9		300	0.56338 (1.33E-2)	0.56357 (1.11E-2)	0.57018 (1.40E-2)	0.56862 (9.94E-3)
UF10		300	0.49573 (1.36E-1)	**0.85499** * (9.58E-4)	0.41166 (3.40E-1)	**0.84215** * (3.51E-3)
WFG2	2	41	0.85728 (1.06E-2)	**0.97518** * (1.18E-2)	0.84715 (1.23E-2)	**0.95977** * (1.02E-2)
WFG3		41	0.84681 (6.36E-3)	0.84961 (6.54E-3)	0.83505 (1.19E-2)	**0.84317** * (8.32E-3)
WFG4		40	0.99153 (9.83E-3)	0.98916 (8.35E-3)	**0.98032** (1.09E-2)	0.97351 * (9.98E-3)
WFG5		40	**0.97941** (3.86E-3)	0.97447 * (4.24E-3)	**0.98076** (4.17E-3)	0.97430 * (3.69E-3)
WFG7		40	0.93748 (1.22E-2)	**0.99385** * (5.10E-4)	0.92685 (1.41E-2)	**0.98565** * (4.19E-3)
WFG8		40	0.86666 (2.03E-2)	**0.89541** * (1.08E-2)	0.83284 (1.65E-2)	**0.84491** * (1.49E-2)
WFG2		401	0.73729 (4.17E-2)	**0.86429** * (1.96E-3)	0.71967 (1.06E-2)	**0.85805** * (4.48E-3)
WFG3		401	0.63240 (1.17E-2)	**0.74807** * (3.22E-2)	0.60861 (8.36E-3)	**0.75809** * (8.93E-3)
WFG4		400	0.66360 (2.09E-2)	**0.84435** * (1.72E-2)	0.62383 (1.44E-2)	**0.79269** * (2.27E-2)
WFG5		400	0.64081 (9.68E-3)	**0.85923** * (1.75E-2)	0.58413 (1.36E-2)	**0.84903** * (1.56E-2)
WFG7		400	0.67987 (1.46E-2)	**0.87752** * (1.64E-2)	0.63490 (1.33E-2)	**0.80371** * (2.21E-2)
WFG8		400	0.56587 (1.33E-2)	**0.79794** * (3.07E-2)	0.52654 (1.04E-2)	**0.75307** * (2.12E-2)
WFG2	3	40	0.89508 (9.06E-3)	**0.97233** * (1.06E-2)	0.88068 (1.72E-2)	**0.95820** * (1.28E-2)
WFG3		40	0.89644 (3.21E-2)	**0.91990** * (1.88E-2)	**0.92193** (3.11E-2)	0.90139 * (1.91E-2)
WFG4		40	**0.97227** (3.95E-3)	0.96489 * (5.08E-3)	0.96010 (9.14E-3)	0.96054 (1.07E-2)
WFG5		40	**0.95600** (3.73E-3)	0.95345 * (1.85E-3)	**0.96749** (2.00E-3)	0.95277 * (1.38E-3)
WFG7		40	0.98000 (5.03E-2)	**0.98470** * (3.55E-3)	0.96277 (3.84E-2)	**0.98642** * (1.04E-3)
WFG8		40	**0.94130** (4.94E-3)	0.92604 * (7.27E-3)	0.84503 (9.21E-2)	**0.91875** * (8.59E-3)
WFG2		400	0.68287 (9.66E-3)	**0.82952** * (7.41E-3)	0.65750 (5.27E-3)	**0.83219** * (1.32E-2)
WFG3		400	0.54703 (2.47E-2)	**0.65643** * (7.34E-2)	0.33447 (4.64E-2)	**0.67913** * (3.18E-2)
WFG4		400	0.46450 (1.38E-2)	0.61422 (2.85E-1)	0.46569 (2.93E-2)	**0.50973** * (3.65E-2)
WFG5		400	0.53172 (4.44E-2)	**0.64958** * (1.86E-1)	0.49506 (2.40E-2)	**0.60250** * (2.47E-2)
WFG7		400	0.48076 (1.48E-2)	**0.68523** * (4.92E-2)	0.49942 (6.12E-2)	**0.71808** * (4.20E-2)
WFG8		400	0.39701 (5.85E-2)	**0.66022** * (2.87E-2)	0.36715 (1.04E-2)	**0.72648** * (1.53E-2)

Test) compared to the original NSGA-II in 44 out of 60 problem instances, and achieves an equal result in 9 cases. xGDE3 is significantly better or equal to its counterpart in 52 out of 60 cases. Notable is that both original algorithms can perform better just in a few 2-objective and 3-objective instances, while in all

higher dimensional problems with 4 and 5 objectives, they lack the ability to even achieve any solution beyond the reference point for the HV calculation, resulting in a HV of zero (denoted as dashes in the tables). The linear combination technique enables these algorithms to achieve significantly better results in even high-dimensional problems with 5 objectives and over 500 decision variables.

Next, we examine the two many-objective algorithms NSGA-III and RVEA. Also in these methods the proposed approach is able to improve the performance of both algorithms significantly. In NSGA-III, the modified version with linear combination performs significantly better in 49 problem instances and performed equally well in another 6. xRVEA outperforms its original version in 49 instances as well, with 5 more draws. An interesting observation is that even though both algorithms are originally designed to work with many-objective problems, their enhanced versions increase their performances even in these instances to a great extent. It is worth to note that the performance on the many-objective instances is significantly increased, even though the subspace of linear combinations is searched with the NSGA-II mechanism, which is usually not designed for many-objective problems. A possible explanation for this fact might be that NSGA-III and RVEA do not posses a mechanism for dealing with high-dimensional search spaces. Since the LSMOP problems do not only contain many objective function but also high-dimensional search spaces, this might turn out a challenge for these algorithms. The positive influence of the linear-combination-search might be, at least partly, due to the inherent reduction of dimensionality. This is also supported by the fact that for all the four algorithms, the original version did only perform better than the x-versions in low-dimensional problems, almost exclusively in UF and WFG problems with only 30 or 40 variables.

Another interesting observation concerns the type of problem where the linear-combination-search seems to work less effectively. Among the few instances where the modified algorithms do not perform best are the (low-dimensional) WFG4 and WFG5 problems, both with 2 and with 3 objectives. WFG4 and WFG5 are both fully separable. Furthermore, NSGA-II and NSGA-III outperform their modified counterparts on the 2-objective LSMOP5 problem, which is also fully separable. The separability of variables suggests that an algorithm can reach optimal solutions by altering variables completely independent of each other. These results imply that for such problems, at least in low-dimensional search spaces, a combination of solutions, which actually alters all variables at the same time through the linear coefficients, might not be suitable.

In summary, we conclude that the proposed approach of optimising linear combination of the population members is able to increase the performance of multi- and many-objective algorithms in most cases. This is especially true for higher numbers of decision variables and higher numbers of objective functions. The authors further tested the method on the remaining problems from the WFG, UF and LSMOP families, which were not reported here due to page limitations, and obtained similar superior performance of the proposed method.

5 Conclusion and Future Work

This article proposed a new mechanism for exploration and solution creation in multi- and many-objective optimisation. The mathematical concept is able to focus the search on relevant areas and at the same time reduce the dimensionality of the original problem without using (possibly expensive) variable grouping methods. After we introduced the mathematical concept, we described how this approach can be incorporated into existing metaheuristic algorithms and explored its capabilities on a variety of benchmark functions with different characteristics and dimensionality. The results indicate that this linear-combination approach can improve the performance of existing methods in both large-scale and many-objective optimisation.

Future work in this area involves exploring the possibilities of this approach further. It can be included into specific large-scale metaheuristics like the WOF as a dimensionality reduction technique. Another possible application might be in constrained problems. Linear combinations have been applied to particle swarm optimisation in [12] to preserve the feasibility of individuals. The approach described in this article can easily be adapted to only allow certain linear combinations, for instance convex ones. If the search is restrained in this way to convex combinations, the algorithm can by definition only create feasible solutions out of existing feasible ones, provided that constraints are linear. Therefore, this can be a promising direction for constraint handling in (large-scale) many-objective optimisation.

References

1. Antonio, L.M., Coello Coello, C.A.: Use of cooperative coevolution for solving large scale multiobjective optimization problems. In: IEEE Congress on Evolutionary Computation (CEC). pp. 2758–2765 (2013)
2. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation **20**(5), 773–791 (Oct 2016). https://doi.org/10.1109/TEVC.2016.2519378
3. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: Test problems for large-scale multiobjective and many-objective optimization. IEEE Transactions on Cybernetics **47**(12), 4108–4121 (Dec 2017)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
5. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 1629–1636. ACM (2006)
6. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. Evolutionary Computation, IEEE Transactions on **18**(4), 577–601 (2014)

7. Gaur, A., Deb, K.: Effect of size and order of variables in rules for multi-objective repair-based innovization procedure. In: Evolutionary Computation (CEC), 2017 IEEE Congress on. pp. 2177–2184. IEEE (2017)
8. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5), 477–506 (2006)
9. Kukkonen, S., Lampinen, J.: GDE3: The third evolution step of generalized differential evolution. In: IEEE Congress on Evolutionary Computation (CEC). vol. 1, pp. 443–450. IEEE (2005). https://doi.org/10.1109/CEC.2005.1554717
10. Li, K., Deb, K., Zhang, Q., Kwong, S.: An evolutionary many-objective optimization algorithm based on dominance and decomposition. IEEE Transactions on Evolutionary Computation **19**(5), 694–716 (Oct 2015). https://doi.org/10.1109/TEVC.2014.2373386
11. Ma, X., Liu, F., Qi, Y., Wang, X., Li, L., Jiao, L., Yin, M., Gong, M.: A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. IEEE Transactions on Evolutionary Computation **20**(2), 275–298 (2016)
12. Mostaghim, S., Halter, W., Wille, A.: Linear Multi-Objective Particle Swarm Optimization, pp. 209–238. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
13. Sander, F., Zille, H., Mostaghim, S.: Transfer strategies from single- to multi-objective grouping mechanisms. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 729–736. GECCO '18, ACM, New York, NY, USA (2018), http://doi.acm.org/10.1145/3205455.3205491
14. Singh, H.K., Isaacs, A., Ray, T.: A pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems. IEEE Transactions on Evolutionary Computation **15**(4), 539–556 (Aug 2011). https://doi.org/10.1109/TEVC.2010.2093579
15. Tian, Y., Cheng, R., Zhang, X., Jin, Y.: Platemo: A MATLAB platform for evolutionary multi-objective optimization. CoRR **abs/1701.00879** (2017), http://arxiv.org/abs/1701.00879
16. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. IEEE transactions on evolutionary computation **10**(1), 29–38 (2006)
17. Zhang, X., Tian, Y., Jin, Y., Cheng, R.: A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Transactions on Evolutionary Computation **22**(1), 97–112 (Feb 2018). https://doi.org/10.1109/TEVC.2016.2600642
18. Zille, H., Ishibuchi, H., Mostaghim, S., Nojima, Y.: Mutation operators based on variable grouping for multi-objective large-scale optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI) (Dec 2016)
19. Zille, H., Ishibuchi, H., Mostaghim, S., Nojima, Y.: Weighted optimization framework for large-scale multi-objective optimization. In: Companion of Genetic and Evolutionary Computation Conference - GECCO. ACM (2016)
20. Zille, H., Ishibuchi, H., Mostaghim, S., Nojima, Y.: A framework for large-scale multi-objective optimization based on problem transformation. IEEE Transactions on Evolutionary Computation **22**(2), 260–275 (April 2018), http://ieeexplore.ieee.org/document/7929324/
21. Zille, H., Mostaghim, S.: Comparison study of large-scale optimisation techniques on the LSMOP benchmark functions. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (November 2017)