Vol. 35, No. 2 (2025) | DOI: 10.37190/ord250204

OPEN ACCESS

Operations Research and Decisions

www.ord.pwr.edu.pl

OPERATIONS RESEARCH AND DECISIONS QUARTERLY

FixedSum: A novel algorithm for generating weight vectors in decomposition-based multiobjective optimization

Syed Zaffar Qasim*[1] Muhammad Ali Ismail[1]

[1]*Department of Computer & Information Systems, NED university, Karachi, Pakistan*

*Corresponding author, email address: *zafarqas@neduet.edu.pk*

Abstract

Many multiobjective optimization algorithms employ weight vectors (WVs) for decomposing a problem into multiple subproblems. These weight vectors should be uniformly spread along the Pareto front. In the recent past, some studies about the development of decomposition-based multiobjective evolutionary algorithms have adopted different methods for generating weight vectors. However these WVs are often clustered, either near the boundary or in inner regions of the search space. In this paper, we have proposed a novel algorithm, FixedSum, for generation of arbitrary number of WVs of any user-specified dimension. These weight vectors are more uniformly spread in the search space and we have compared our results with other methods on 5-, 8-, 10- and 12-D weight vectors. For further validation, we have applied our weight vectors along with the WVs of two other methods for solving the DTLZ problems. All the results demonstrate the improved spread ability of our method as compared to competing approaches.

Keywords: *Multiple objective programming, weight vectors, many-objective optimization, decomposition-based multiobjective algorithm, FixedSum*

1. Introduction

A weight vector, $w = (w_1, \ldots, w_m)^T$ is a vector (or point) of dimension m whose components $w_i \in [0, 1]$ such that $\sum_{i=1}^{m} w_i = 1$. For example, (1/3, 1/6, 1/2) is a WV of dimension 3. Weight vectors in fact represent points on a unit simplex of dimension $m - 1$ [9]. In classical multiobjective optimization [29], they have been used to assign weights to different objectives according to their relative importance. However, in some contemporary evolutionary approaches [2, 3, 25, 34, 39], called decomposition-based approaches, a collection of widely-spread weight vectors are required for improving the spacing of solutions in the objective space. Accordingly, a number of techniques were proposed that attempted to produce uniformly scattered points. However, the generation of points that are uniformly scattered in the ideal manner is a tedious job and hence almost all the proposed methods in fact approximately spread

Received 18 April 2024, accepted 3 January 2025, published online 8 February 2025
ISSN 2391-6060 (Online)/© 2025 Authors

the generated WVs in the respective space. In view of the shortcoming of existing methods in producing widely scattered points, we have proposed an improved algorithm, FixedSum, for generation of weight vectors. The proposed FixedSum provides a collection of WVs with enhanced spread as compared to the earlier approaches.

In order to visually demonstrate the spread of weight vectors in the m-dimensional space, we have employed the parallel coordinate (PC) plot in this study. The PC plot depicts a point in cartesian coordinates by multiple cross-lines passing through parallel axes each representing a specific dimension of the point [30]. It is typically used to illustrate the diversity of solutions obtained in multiobjective optimization with more than three objectives. In this work, the cross-lines will represent each WV such that each cross-line joins successive component values of a particular WV. The horizontal axis here marks the dimension number, $j \in [1, m]$ of WV starting from 1 and so on whereas the vertical axis represents the values w_j^i of jth component of WV w^i.

The PC plot for a set of five weight vectors in table 1 is shown in fig 1. The zig-zag pattern of cross-lines demonstrate the difference in weights in a given weight vector.

Table 1. Set of weight vectors with m=4

w_1	w_2	w_3	w_4
0.2	0.1	0.3	0.4
0.4	0.2	0.1	0.3
0.1	0.3	0.4	0.2
0.3	0.4	0.2	0.1
0.2	0.35	0.15	0.3

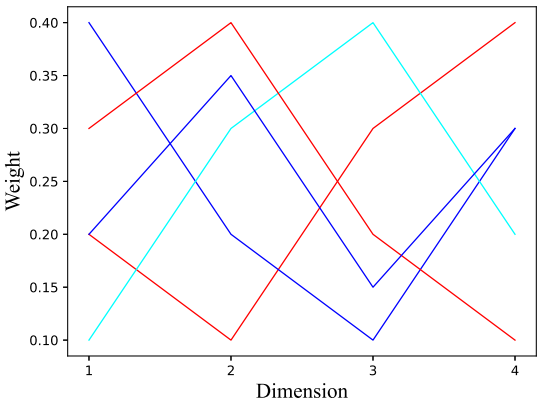


Figure 1. PC plot of weight vectors in table 1

The remaining part of this paper is organized as follows. In sec 2, we give a review of concepts which are relevant to this work. These include multi- and many-objective optimization and the decomposition-based approaches to multiobjective optimization. In the next sec 3, an appropriate account of the existing approaches for generating WVs is given along with the shortcomings of these approaches. Then, for the sake of making the description more easier and understandable, the basic algorithm for generating one weight vector at a time will be described in sec 4.1. This basic algorithm is named as Weight Vector Generation Algorithm (WVGA). After this, the actual FixedSum algorithm will be presented in sec 4.2 in

which some innovative ideas have been incorporated to attain the wider spacing between weight vectors in the m dimension space. The FixedSum will iteratively generate N weight vectors and is based on WVGA. As the next logical step, the sec 5 will discuss different experimental results for comparing FixedSum with other approaches and hence demonstrate the superior performance of FixedSum over other approaches. Finally, the conclusion of this work will be presented in sec 6.

2. Review of relevant concepts

2.1. Multiobjective Optimization

Many formal decision making situations involve the simultaneous optimization of multiple conflicting objectives. For instance, the general business environment requires minimizing the operating cost of a business while keeping a stable work force along with improving the quality of service or product which also seem to be conflicting to each other. In principle, multiobjective optimization (MO) is more difficult and time-consuming as compared to single-objective optimization because, in the former case, there may not exist one unique ideal solution which is best (global minimum or maximum) with respect to all objectives. In fact there is a (finite or infinite) collection of equivalent tradeoff solutions which are optimal as compared to the remaining solutions in feasible search area.

In order to give a deeper insight of the field, the mathematical definitions of related concepts are presented in the following [4, 8]

Multiobjective Optimization Problem (MOP): Considering a vector $F(x)$ with m objective functions, a constraint-based MOP can be expressed, without loss of generality, as follows:-

$$Minimize\ F(x) = (f_1(x), \cdots, f_m(x)) \tag{1}$$

$$\begin{gathered}\text{while meeting the constraints}\\ p_i(x) = 0, i = 1, \cdots, g, \text{ and}\\ q_j(x) \leq 0, j = 1, \cdots, h\\ \text{where x} \in \Omega\ (\Omega \subseteq \mathbb{R}^k)\end{gathered}$$

Here the evaluation function $F : \Omega \rightarrow \Lambda\,(\Lambda \subseteq \mathbb{R}^m)$ maps the vector of decision variables $x = (x_1, \cdots, x_k)$ to output vectors $y = (y_1, \cdots, y_m)$. In any optimization problem, the decision variables are those numeric quantities that are chosen using some suitable algorithm. For the above example of typical business environment, the decision variables might be the cost of raw material, wages paid to the workers etc. It is to be noted that $p_i(x) = 0$ and $q_j(x) \leq 0$ denote constraints that must be fulfilled, when minimizing $F(x)$ to enable search in the feasible space only. Next the fitness evaluation of solutions in MO is traditionally based on Pareto dominance which can be defined as follows:-

Pareto Dominance (PD): A vector $\vec{u}$ Pareto dominates other vector $\vec{w}$ when none of its component is higher than $\vec{w}$ and at least one of its component is strictly lower than $\vec{w}$.

Pareto Optimality: A solution $x \in \Omega$ is said to be Pareto Optimal w.r.t. Ω iff there is no $x' \in \Omega$ for which $w = F(x') = (f_1(x'), \cdots, f_m(x'))$ dominates $u = F(x) = (f_1(x), \cdots, f_m(x))$.

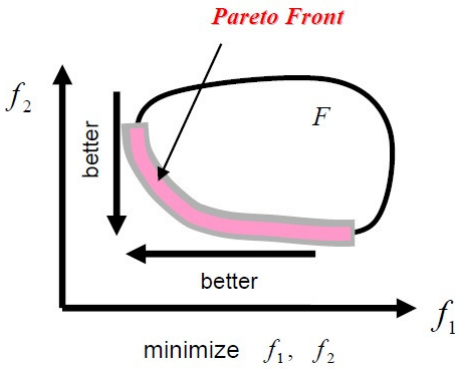


Figure 2. The Pareto front for a biobjective problem

Pareto Optimal Set (POS): The set of Pareto optimal points in the decision space. For an MOP, $F(x)$, the Pareto Optimal Set, POS, can be expressed as:

$$POS = \{x \in \Omega \mid \neg\exists x' \in \Omega, F(x') \prec F(x)\}$$

Pareto Front (PF): The set of points in the objective space corresponding to the Pareto optimal points in the decision space of a given MOP (Fig 2 [31]).

$$PF = \{\mathrm{u} = \mathrm{F(x)} \mid \mathrm{x} \in POS\}$$

Now here is the time to define the two main goals for any MOP which are the convergence and diversity of solutions [1, 7] along the Pareto front. Also the intrinsic presence of multiple optimal solutions in MOPs necessitated the use of population-based evolutionary and meta-heuristic techniques for the optimization of such problems. To name a few, such evolutionary and meta-heuristic techniques include Genetic algorithms, Particle Swarm Optimization, Simulated Annealing etc. Accordingly, the notion, evolutionary multiobjective optimization (EMO) refers to the use of these algorithms for the optimization of multiobjective problems. The most popular Pareto-dominance based multiobjective evolutionary algorithms (MOEAs) include NSGA-II [10], SPEA2 [42] and GDE3 [23]. Apart from this, some state-of-the-art work on MOP has been recently reported in [18, 19] that utilize gradient-descent and adaptive region decomposition for proposing solutions to expensive or constrained optimization problems.

2.2. Many-objective optimization

There are various decision making situations that involve the optimization of higher than three objectives. Such problems are commonly called as many-objective optimization problems [9, 17, 36, 38]. Most computing and software engineering problems identified in various surveys [26, 32, 33, 35, 37] involve optimizing four or more objectives. For instance, the software project portfolio optimization problem [22] involves about eight goals which include maximization of potential revenue, minimizing the available resources, strategic similarity of projects with software development organization, maximization of positive as well as minimization of negative synergy effects with-in-the projects selected for a portfolio.

These many-objective problems have offered great difficulties to state-of-the-art multiobjective evolutionary algorithms (MOEAs) [9, 25]. One of these difficulties is the inability of PD approach for

evaluating the fitness of solutions (in many-objective scenario) which has resulted in exploring alternate approaches for evaluating the quality of solutions. The second difficulty, worth mentioning, is that the number of solutions needed to properly represent the Pareto front increases exponentially with the number of objectives. To make matters complicated, another difficulty, arising from the second, is the complexity of computation of diversity measure of solutions in a large-dimensional space [28].

2.3. Decomposition-based methods

The decomposition-based methods split an MOP into a number of subproblems and then simultaneously optimizes them for finding solution of the original problem. The decomposing process involves choosing the same number of WVs as the subproblems that are widely scattered in the search space. Also the number of subproblems is equal to population size N and each subproblem is associated with a solution and a weight-vector. As WVs determine the search directions, there are good chances that the obtained solutions (at the termination of evolution) will be uniformly spaced over the Pareto front.

These splitting methods are in fact generic approaches since any suitable scalarization technique (e.g. weighted sum, Penalty-based Boundary Intersection (PBI) Method etc) can be inserted in their framework. For example, in the Weighted Sum Method [29], the weighted sum total of individual objectives is considered for the scalarization purposes. Suppose $w = (w_1, \cdots, w_m)^T$ be a weight vector such that $w_i \geq 0$ for all $i = 1, \cdots, m$ and $\sum_{i=1}^{m} w_i = 1$. Then each scalar optimization subproblem can be formulated as

$$minimize \qquad g^{ws}(x|w) = \sum_{i=1}^{m} w_i f_i(x) \tag{2}$$

$$\text{Subject to } x \in \Omega$$

The decomposition methods have been found very effective for optimizing many objective problems. Among the most popular decomposition-based MOEAs include MOEA/D [40] and MOEA/DD [25]. Another EMO of significant importance is NSGA-III [9] which adopts weight vectors as reference points for the provision of diversity of solutions. Finally, an improved variant of MOEA/D, named I-MOEA/D was proposed by Zheng *et al.* [41]. This algorithm proposes a new decomposition technique referred to as weighted mixture-style method. Besides decomposition-based methods, other approaches also exist for improving the diversity of solutions in MOP. Among them, the two recent works in [20, 21] are worth-mentioning for enhancing the diversity and handling constrained-optimization problems.

3. Existing approaches for WV generation

In this section, we give a suitable account of the existing approaches for the generation of weight vectors and will mention their shortcomings:-

3.1. RandomSum method

In this method [16], each weight vector (WV) of dimension m is found by generating a vector V of size m containing random (positive) integers, $irnum_0$, $irnum_1$, ..., $irnum_{m-1}$ such that the total of the numbers is T as shown in eq 3 below:-

$$T = irnum_0 + irnum_1 + \cdots + irnum_{m-1} \tag{3}$$

Then each component of weight vector, w_i, is calculated by following formula:-

$$w_i = irnum_i / T \tag{4}$$

We call this approach for generating weight vectors as RandomSum since the total T is random for each instance of a weight vector. It has been experimentally verified that this approach does not result in widely spread weight vectors (see sec 5.3 below) since the random sum T in denominator varies with the numerators (eq 4). For rectifying this problem, the FixedSum approach has been proposed in this work in which the sum T is selected first and then the set of random integers $irnum_i$ is generated so that their sum is exactly T. Since the value T remains fixed for generation of all N weight vectors whereas the set of $irnum_i$ values are changed every time, this results in much better randomization and improved spread of weight vectors.

3.2. Das and Dennis method

A systematic method was proposed in [6] for producing a collection of N weight vectors $W = (w^1, \cdots, w^N)$ on a normalized hyperplane. This hyperplane is a unit simplex of $(m-1)$ dimensions with an intercept of one on each axis. The total number of weight vectors, N, in an m-objective problem is given by

$$N = \binom{q+m-1}{m-1} \tag{5}$$

Where q is the number of divisions considered along each objective axis. For a 3-objective problem having four divisions (q=4) along each objective, a total of $\binom{4+3-1}{3-1} = 15$ weight vectors are generated. However, for a 4-objective problem, with q=4, the total number of weight vectors produced will be 35. An example of this technique is given in fig 3 [25] with m=3 and q=4.

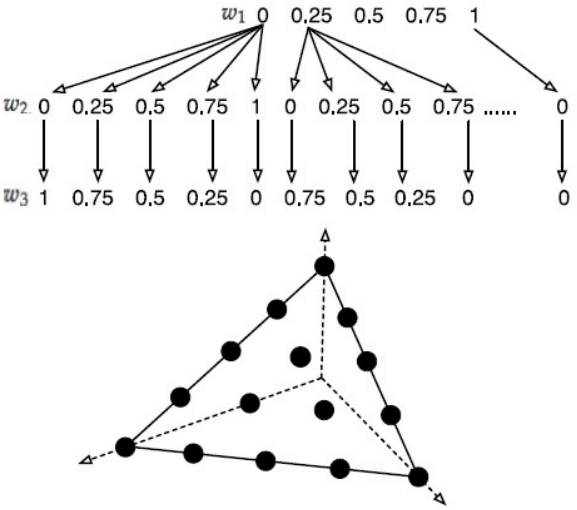


Figure 3. Das & Dennis's weight vector generation process for 3-D space with q=4

The limitations of this method are that the vectors obtained are not uniformly spread and the number N

of these weight vectors should satisfy the eq 5 above [5]. Thus N will increase nonlinearly with m (see table 2). As shown by the parallel coordinate plot in fig 4 the weight vectors (total 126) obtained using Das and Dennis's method for m=6 and q=4 are not distributed uniformly in space.

Table 2. Number of weight vectors for given m and q

m	q	N
4	4	35
4	5	56
4	6	84
5	4	70
5	5	126
5	6	210
6	4	126

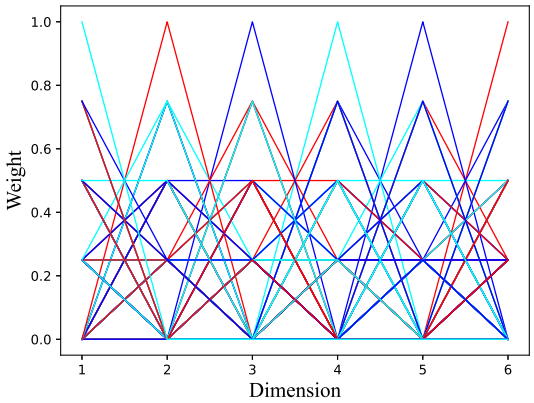


Figure 4. NonUniform weight vectors with m=6 and q=4

As discussed in [9], in order to have intermediate weight vectors (not on the boundaries) within the simplex, one should set $q \geq m$. However, in a large-dimensional objective space, there will be a large amount of weight vectors even if $q = m$ e.g., for seven-objective case, q = 7 will give $\binom{7+7-1}{7-1} = 1716$ weight vectors. This will substantially increase the computational cost of an EMO algorithm.

3.3. Two-layer weight vector generation method

In order to overcome the short-coming of the Das-Dennis method for WV generation, a two-layer WV generation method was proposed by Li *et al.* [25]. First of all, the sets of weight vectors in the boundary and inner layers (represented as $B = \{b^1, \ldots, b^{N1}\}$ and $I = \{i^1, \ldots, i^{N2}\}$, respectively, where $N_1 + N_2 = N$) are generated using the Das and Dennis's approach, with different q values. Then, the WV coordinates in the inner layer are contracted by a coordinate transformation. Specifically, as for a weight vector in the inside layer $i^p = (i^p_1, \ldots, i^p_m)^T$, $p \in \{1, \ldots, N_2\}$, its jth component is reevaluated as

$$i^p_j = \frac{1-\beta}{m} + \beta * i^p_j \tag{6}$$

where $j \in \{1, \ldots, m\}$ and $\beta \in [0, 1]$ is a shrinkage factor (typically set as 0.5). Finally, B and I are mixed together to form the final WV set W.

As a demonstration of two-layer WV generation method, we produced N1=56 WVs on the boundary with m=6 and q=3. Next, a total of N2=126 WVs were produced on the inner layer with m=6 and q=4 and then shrinked using the eq 6. Then N1 and N2 were combined to produce a total of N=182 WVs. The parallel coordinate plot of these WVs is shown in fig 5. The plot clearly shows that most of the PCPs now lie in the range of 0.0 to 0.60 (as compared to fig 4) resulting in an increase of intermediate WV with decline in the number of WVs on the boundary. However, even in this case, the WVs are not uniformly distributed and seems to be sparsely located in the space.

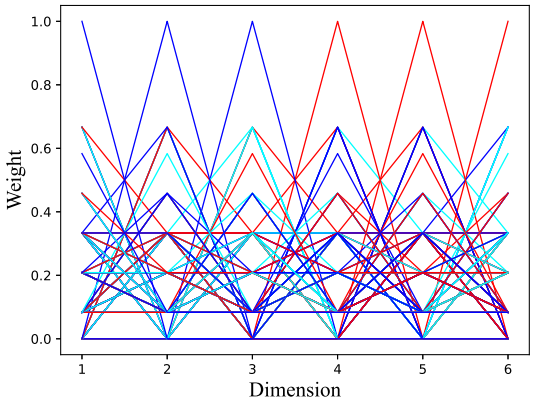


Figure 5. NonUniform weight vectors using two-layer method

In a later study on the two-layer WV specification [14], following method was suggested for updating the weight vectors of the inside layers:-

$$w_i = \gamma w_i + (1-\gamma)(1/m), \;\; 0 \leq i \leq m-1 \tag{7}$$

Here the parameter $\gamma \in (0, 1)$. In order to obtain WVs near the boundary, γ had to be kept close to 1. Conversely, for obtaining points close to the inside layer, γ had to be chosen near to 0.

More recently, the reference [13] has also discussed the limitations of two-layer method which include the lack of uniform spacing between WVs and the inability to generate any user-specified number of weight vectors.

3.4. Uniform design method

The uniform design method [12] was employed in [5] for decomposition-based many-objective optimization. It involves sampling a collection of uniformly distributed points from a bounded set $B \subset R^M$. Here B is taken as a unit hypercube of m-dimensions such that

$$B = \{(x_1, x_2, \ldots, x_M) \,|\, 0 \leq x_i \leq 1, \; i = 1, \ldots, M\} \tag{8}$$

Next, if we consider some specific point $p = (p_1, p_2, \ldots, p_M) \in B$, a hyper-rectangle from 0 to p, denoted by B_p, is represented as follows:-

$$B_p = \{(x_1, x_2, \ldots, x_M) \,|\, 0 \leq x_i \leq p_i, \; i = 1, \ldots, M\} \tag{9}$$

Considering a set of N points in B, suppose that N_p of these points lie in the hyper-rectangle B_p, then the proportion of points lying in B_p is equal to N_p/N. Given that the volume of B is 1, the fraction of volume of B_p is

$$V_{Bp} = p_1 \times p_2 \times \cdots \times p_M \tag{10}$$

The uniform design in fact is the method of determining N points in B such that the difference between N_p/N and V_{Bp} is minimized i.e.

$$\min_{p \in B} |\frac{N_p}{N} - V_{Bp}| \tag{11}$$

One underlying approach is to first find an $N \times M$ matrix $G\ (= [g_{ij}])$ whose individual entries are evaluated as follows:-

$$g_{ij} = mod(i\mu^{j-1}, N) + 1,\ i = [1, N],\ j = [1, M] \tag{12}$$

Where $\mu \in \{1, \ldots, N-1\}$ which results in $N-1$ different integer matrices G^{μ}. The next step is determining a value δ (among $N-1$ values of μ) for finding G^{δ} such that the condition 11 above is satisfied. The final step in evaluating weight vector is deriving a matrix $C(= [c_{ij}])$ from G^{δ} as follows:-

$$c_{ij} = \frac{2g_{ij}^{\delta} - 1}{2N},\ i = [1, N], j = [1, M] \tag{13}$$

One drawback of this approach is that for evaluating N weight vectors of dimension M, it requires the evaluation of $N-1$ matrices to find a particular value of δ which makes the computational complexity of step in eq 12 to be $O(MN^2)$ which is significantly higher than our proposed FixedSum (to be shown in sec 4.2). Also it necessitates the selection of point p inside B which may have a significant bearing on the effectiveness of the results.

4. Description of Methodology

This section is meant to discuss the proposed algorithm for WV generation in an algorithmic language. Here it is assumed that this language has different functions for performing different operations: $genRand(n)$ for generating a random number between 1 and n (limits inclusive) whereas n is a positive integer. The different file operations available are $createFile(F)$ for creating a new file, $appendFile(F, W)$ for appending a record W to new file whereas $closeFile(F)$ is for closing a file F.

4.1. The WVGA algorithm

The iterative algorithm, WVGA, for generating a single weight vector of dimension m is shown in algorithm 1 and is based on the idea mentioned in sec 3.1 about fixing the value of sum T. It has a single input m. T as described in sec 3.1 is the total of all m random integers $irnum_i$ but is chosen first before selecting the set of random integers $irnum_i$. Here T has been selected (step 1) as an integer value that is higher than $\phi * (m-1)$ by quantity L where the value of ϕ is selected as 100 in the experiments during this study.

The WVGA produces different values of $irnum_i$ by slicing the range of 1 to T into m random sizes. For this purpose, it uses an auxiliary variable $temp$ which, during iteration i, defines some fraction of T from which $irnum_i$ is chosen randomly (see fig 6). The $temp$ is initialized to T (step 2) and then an

Input: m (dimension of weight vector)
Output: Weight Vector W
Global variables: ϕ, T (grant total), R, $temp$, $irnum$ (integer vector), L
1. $T := \phi * (m-1) + L$
2. $temp := T$
3. $R := genRand(\phi)$ [To generate a random number R between 1 and ϕ]
4. $temp = temp - R * (m-1)$
5. $i = 0$ [index of array $irnum$]
6. **while** $(i < m-1)$ **do**
7. $\quad irnum[i] = genRand(temp)$ [To generate a random number between 1 and $temp$]
8. $\quad temp = temp - irnum[i] + R$
9. $\quad i++$
end
10. $irnum[i] = temp$
11. **for** $i := 0$ **to** $m-1$ **do**
| 12. $W[i] = irnum[i]/T$
end
13. **return** W

Algorithm 1. Weight Vector Generation Algorithm (WVGA)

amount of $R*(m-1)$ is borrowed from it by temporarily reducing it by this amount. Here R is a random variable chosen at the start between 1 and ϕ, the limits inclusive (step 3-4). As the maximum possible value of R is ϕ, the greatest possible amount borrowed from $temp$ is $\phi * (m-1)$ so it is still higher than zero after borrowing. Next the index, i, of array $irnum$ is initialized to zero (step 5) and the iterative steps of algorithm (steps 6-9) then start.

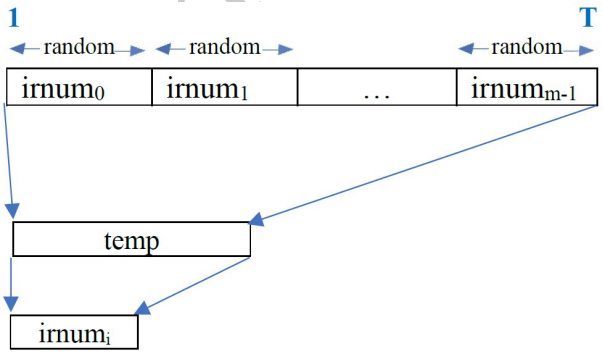


Figure 6. The m random integers $irnum_i$ are generated in WVGA to make a total of predetermined value T. However each $irnum_i$ is generated in iteration i from $temp$ which contains a partial amount of T at a time.

In each iteration, the next $irnum_i$, is produced as an integer between 1 and $temp$ (step 7). Then $temp$ is reduced by $irnum_i$ (to slice $irnum_i$ out of $temp$) and variable R is added to temp (step 8). Hence in each iteration, R is added to $temp$ to recover the value $R * (m-1)$ which was borrowed earlier (step 4) from $temp$. This means every $irnum_i$ is computed based on current value of $temp$ which holds a fraction of value of T at a time. As the iterations end, the value left in $temp$ is

$$T - \sum_{i=0}^{m-2} irnum_i \tag{14}$$

This value will finally be assigned to $irnum_{m-1}$ (step 11) and in this way, the m $irnum_i$ values are generated such that eq 3 is satisfied. Since whole $R * (m-1)$ is returned back to $temp$, the i random variables are in fact sliced out of the whole range of 1 to T. The weight vector is finally generated using for loop in step by dividing each $irnum_i$ by T i.e. using eq 4.

Complexity analysis of WVGA

This algorithm involves two simple (while and for) loops and all remaining statements are sequential. The while (steps 6 to 9) and for (steps 11 and 12) loops are executed $m-1$ and m times respectively. The statements inside these loops are $O(1)$ so the overall complexity of WVGA algorithm for generating one weight vector is $O(m)$.

4.2. The FixedSum algorithm

The WVGA algorithm generates only one weight vector in single execution whereas the decomposition-based MOEAs need as many weight vectors of dimension m as there are solutions N belonging to population. Hence the WVGA can be extended to execute iteratively to generate N weight vectors which will finally be stored in a text file. The resulting extended algorithm, FixedSum, is shown in algorithm 2.

Input: m (dimension of weight vector) and N (number of weight vectors)
Output: Data file F with N Weight Vectors
Global variables: ϕ, T (grand total), R, $temp$, $irnum$ (integer vector), L
1. $T := \phi * (m-1) + L$
2. $createFile(F)$ [creates new file F in current directory]
3. **for** $k := 1$ **to** N **do**
   4. $temp := T$
   5. $R := genRand(\phi)$ [generate a random number R between 1 and ϕ]
   6. $temp = temp - R * (m-1)$
   7. $j = (k-1)\%m$ [starting index of array $irnum$]
   8. $count = 0 : i = j$
   9. **while** $(count < m-1)$ **do**
   10. $irnum[i] = genRand(temp)$ [generates a random number between 1 and temp]
   11. $temp = temp - irnum[i] + R$
   12. $i := (i+1)\ \%\ m$
   13. $count++$
   end
   14. $irnum[i] = temp$
   15. **for** $i := 0$ **to** $m-1$ **do**
      16. $W[i] = irnum[i]/T$
   end
   17. $appendFile(F, W)$
end
18. $closeFile(F)$
19. **Exit**

Algorithm 2. FixedSum algorithm for weight vector generation

In the description of this algorithm, we will mention only the statements which have been added or modified as compared to algorithm WVGA. In step 1, the value T is selected once as mentioned in sec 4.1 and

remains fixed throughout the generation of N weight vectors. In step 2, a new text file F will be created whereas in step 3, a loop starts which will repeat, N times, the whole procedure of WVGA algorithm for generating weight vectors. After each weight vector is generated, it is appended to the text file F (step 17) and after all weight vectors are produced and appended, the file F is closed (step 18) at the end.
Another important factor to note: as the value of $temp$ is temporarily reduced by $R * (m - 1)$ at the beginning (step 6) so that random integers start generating from the reduced value; in spite of that, earlier values of $temp$ might be higher (if R is smaller) which will result in higher values of initial components of weight vectors. Hence the vectors will become biased towards few dimensions and will not be widely spread (shown graphically in sec 5.2). In order to avoid that situation, all the random integers produced for WV generation will not strictly start at array index $i = 0$ (a slight deviation from step 5 of WVGA algorithm 1. More specifically, they will start from $j = (k - 1)\ \%\ m$ where $\%$ is the modulus operator and k is the iteration counter of outer loop and represents the number of particular WV to be generated in current iteration (step 7). This will result in the assignment of sequence of $irnum_i$ values starting from index 0 to $m - 1$ with equal probability.

In short, one main purpose of outer loop in FixedSum algorithm is to select the starting index j for

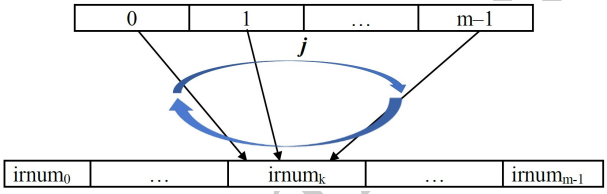


Figure 7. The starting index j of $irnum_i$ for WV generation repeatedly shifts from 0 to $m - 1$ and then wraps around to 0 and so on

storing the $irnum_i$ values and from there index i will increment in a circular fashion in the inner loop during WV generation (see fig 7). In the light of the fact that k goes from 1 to N and $N >> m$, the starting index j for a particular WV will keep on shifting, beginning with zero initially and then attaining a value $m - 1$ and next resetting to zero and so on during generation of all N weight vectors. Therefore, if k=20 (in outer loop) and m=8 then i (in inner loop) will start from 3, next 4 and then finally attain a value 2 for producing the 20th WV i.e. the modulus operator will result in incrementing i in a circular fashion. The empirical results have demonstrated (see sec 5.2) that this technique of varying i will result in better spread of solutions.

Complexity analysis of FixedSum

As described earlier, the computational complexity of the routine for computing one weight vector is $O(m)$. In the FixedSum algorithm, the same routine is invoked N times to generate N weight vectors. Hence the complexity of FixedSum algorithm is $O(mN)$.

4.3. Theoretical analysis of FixedSum

Before presenting the detailed empirical analysis of FixedSum in the next section 5, here we give a brief theoretical analysis of the diversity characteristic of our proposed approach. First of all, we further elaborate the idea presented in sec 3.1 by showing the weight vector expressions for the RandomSum

and FixedSum approaches in eq 15 and 16 as follows:-

$$WV_{rs} = \left(\frac{irnum_0}{T_j}, \frac{irnum_1}{T_j}, \ldots, \frac{irnum_{m-1}}{T_j} \right) \tag{15}$$

$$WV_{fs} = \left(\frac{irnum_0}{T}, \frac{irnum_1}{T}, \ldots, \frac{irnum_{m-1}}{T} \right) \tag{16}$$

It is quite intuitive that as the denominator T in eq 16 is fixed, the WVs for different $V = (irnum_0,\ irnum_1,\ \ldots,\ irnum_{m-1})$ will be widely varying from each other whereas the WVs produced using eq 15 for different V might be very close to each other as the denominator T_j will also vary for each weight vector j. This is empirically demonstrated in sec 5.3.

Furthermore, the algorithm WVGA (see sec 4.1) while generating the vector V makes each respective $irnum_i$ to vary in a limited range. The reason being that the different $irnum_i$ used to find a particular weight vector are in fact slices of grand total T. However at any particular iteration in fig 6, $irnum_i$ is computed using a window of T represented by $temp$ (which is also a random number) so that each $irnum_i$ will uniformly vary in a limited but wider range than RandomSum. Hence analytically speaking, FixedSum, in algorithm 2, which iteratively invokes WVGA, allows a wider diversity of WVs than RandomSum algorithm by combining the features of windowing and index shifting.

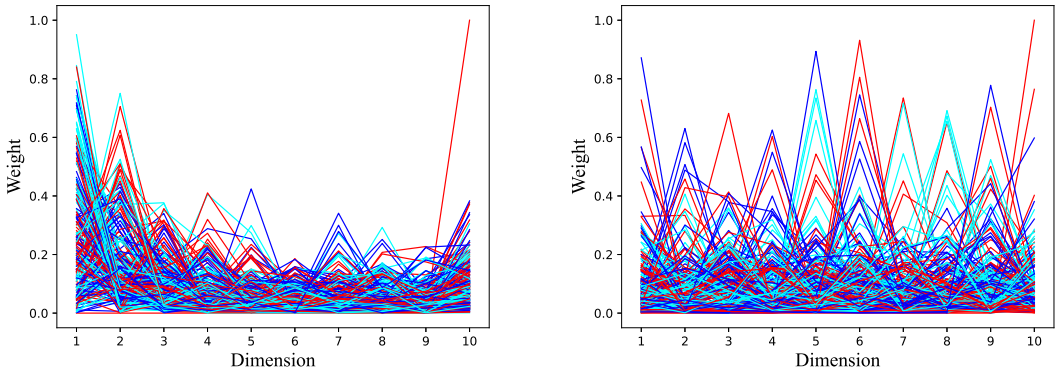
5. Experimental Study

In order to validate our proposed FixedSum, we have graphically compared its results with other three state-of-the-art methods, Two-layer, RandomSum and Uniform Design methods, mentioned in sec 3. Moreover, we have compared the performance of these methods for solving benchmark problems using different MOEAs.

5.1. Experimental Arrangements

We have generated weight vectors for dimensions 2-, 5-, 8-, 10- and 12-D using FixedSum and the other methods as identified above. In order to initialize T for FixedSum, the value of ϕ is taken as 100 whereas L is chosen as 50. Since the total T must be sufficiently big value so that it can be split into m components $irnum_i$ with different sizes for each WV, the value of 100 for ϕ was found to be sufficiently large for it.

As the number of weight vectors, N, in Das & Dennis and the two-layer WV generation methods are fixed for a given m and q, the value of N selected for all experiments is therefore based on these two parameters. All the WVs are plotted using PC graph to visually compare their spread ability in the m-dimensional space. For further validation and demonstration of their effectiveness in solving MOPs, the WVs produced for different m, using different methods, are used for solving the DTLZ1 to DTLZ4 problems using the MOEA/DD algorithm. MOEA/DD [25] is the state-of-the-art algorithm for solving MOPs (especially with many objectives) and is based on dominance and decomposition whereas DTLZ1 to DTLZ4 [11] are challenging scalable benchmark problems designed for evaluating the ability of MOEAs to effectively converge and diversify solutions along the estimated Pareto front. All these experiments are executed for a total of 30 runs with 250 iterations in each run.



a). WV without index shifting **b).** WV with index shifting

Figure 8. Impact of Index Shifting on performance of FixedSum

5.2. Impact of Index Shifting on performance of FixedSum

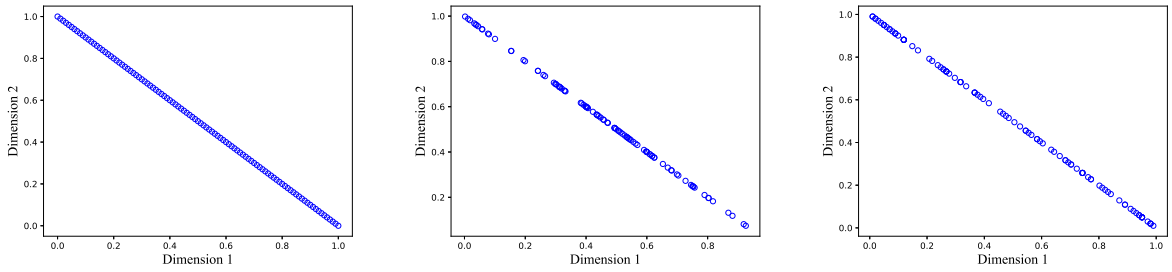
The proposed algorithm generates WVs using index shifting, in which starting index of $irnum_i$ is shifted for each next weight vector using modulo operator. Here the impact of this operation upon diversity characteristic of FixedSum is examined. Fig 8a) shows the PCP plot obtained for WVs without index shifting whereas fig 8b) shows the plot with index shifting for N=200 and m=10. This clearly demonstrates that, without index shifting, weight vectors produced are biased i.e. the values of weights obtained for first two dimensions are in a wider range than the values for remaining dimensions. However with index shifting, as shown in fig 8b), the WVs obtained are much more uniformly spread and the distribution of WVs among the boundary as well as inner regions of the m-dimensional space is much better.

5.3. Comparative analysis

In this section, we compare the performance of our proposed method with the results of other approaches for different dimensions of weight vectors. The results are shown graphically to demonstrate the real difference of our method with the other approaches.

5.3.1. Weight Vectors of dimension 2

Here we demonstrate the simplest case of generation of weight vectors of dimension 2. Fig 9 shows the WVs produced using the three approaches namely Das & Dennis method, RandomSum method and FixedSum method. Here dimension 1 of WVs is plotted on horizontal whereas dimension 2 is plotted on vertical axis. As fig 9a) illustrates, the WVs produced with DD method are uniformly spread ideally in the 2D space. However the spread of WVs obtained using the RandomSum method is the worst among the three approaches as it seems to produce different clusters of WVs with wider space between these clusters as compared to the FixedSum method. Finally the WVs obtained with FixedSum method are less uniformly spaced as compared to DD as demonstrated by some space in between the points, however, the spread in this case seems much better than RandomSum method.



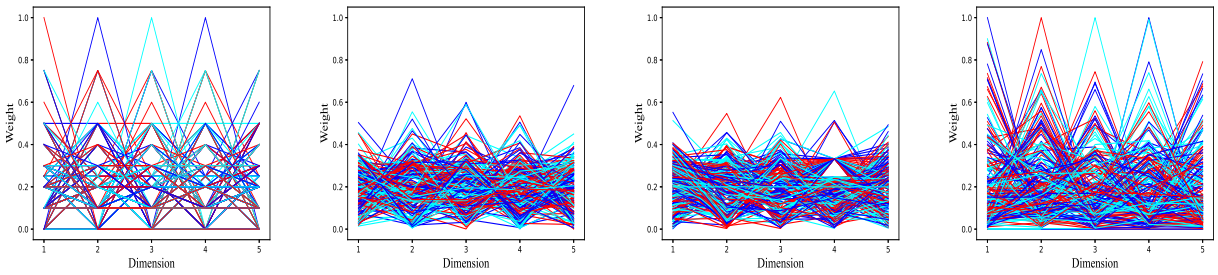
a). Das & Dennis method **b).** RandomSum method **c).** FixedSum method

Figure 9. Weight vectors of dimension 2

5.3.2. Weight Vectors of dimension 5

Generating and plotting the weight vectors

The number of weight vectors for dimension 5 are principally determined using two-layer method (sec 3.3) with different values of q for boundary and inner layers. Using q = 4, $N1 = \binom{4+5-1}{5-1} = 70$ weight vectors are chosen on the boundary layer whereas using q = 5, $N2 = \binom{5+5-1}{5-1} = 126$ weight vectors are chosen on the inner layer to get a total of $N = 196$ weight vectors. For comparison purpose, N is also set as 196 for FixedSum, RandomSum and uniform design methods. The δ for uniform design method is taken as 163 as recommended in [5] for M=5. As a result, the weight vectors of dimension 5, obtained using the four approaches, are shown in fig 10. A careful comparison of figures 10a) to 10d) shows that the WVs generated using FixedSum method are more densely and uniformly spread as compared to the other three approaches. The WVs generated by two-layer method, although seem to be uniformly spread, are sparsely located in space whereas the WVs produced by RandomSum and uniform design methods are not uniformly spread in the sense that the values obtained for each dimension are in a smaller range; for example in fig 10b) and 10c), very few WVs have a component with a value 0.6 or higher. Hence the RandomSum and uniform design methods do not cover WVs belonging to the boundary regions where few dimensions have a comparatively higher values of weights as compared to the remaining dimensions.



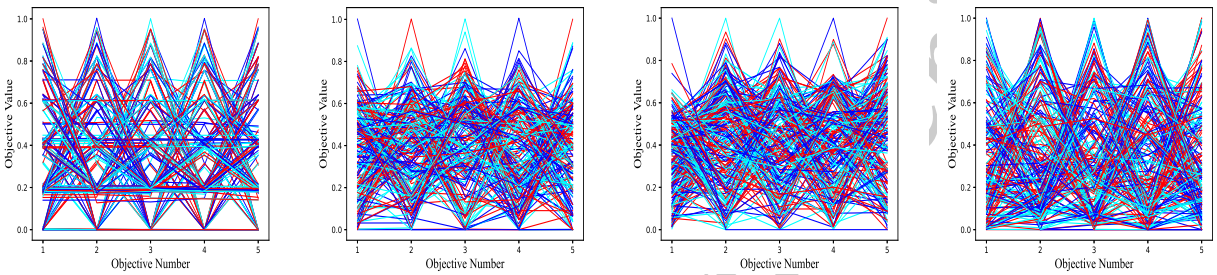
a). Two-layer method **b).** RandomSum method **c).** Uniform design method **d).** FixedSum method

Figure 10. Weight vectors of dimension 5

Plotting the results of DTLZ4

In order to demonstrate the superiority of our FixedSum algorithm in providing a better spread of WVs,

we have solved the DTLZ4 problem using MOEA/DD algorithm by utilizing the 5-D weight vectors obtained using the four methods namely two-layer method, RandomSum, uniform design and FixedSum. The resulting Pareto fronts are plotted using the PC plots in fig 11. A comparison of these plots in fig 11a) to 11d) clearly shows that the solutions obtained using WVs of proposed FixedSum algorithm are more uniformly spread over the entire Pareto front as compared to the solutions acquired through other WVs. As shown in fig 11a), the objective values for the solutions got through the two-layer method do not cover the entire respective domain (i.e. $f_i(x) \in [0, 1]$) and are missing certain ranges of values. However the solutions got through RandomSum and uniform design in fig 11b) and 11c) are more concentrated in the middle sections of Pareto front and the tradeoffs do not seem to be very good. Finally, the solutions got through FixedSum method (see fig 11d)) cover the entire range of objective values for different objectives.



a). Two-layer method **b).** RandomSum method **c).** Uniform Design method **d).** FixedSum method

Figure 11. Solutions of DTLZ4 problem using MOEA/DD

Quantitative comparison of different approaches

For comparing the performance of these four approaches on 5-D problems, we conducted an experiment for solving the four DTLZ benchmark problems (DTLZ1 to DTLZ4) [11] using MOEA/DD (with a population size of 196) and summarized the results using IGD+ indicator. The IGD+ (inverted generational distance) by Ishibuchi [15] gives combined estimate of convergence and diversity of solutions for an EMO algorithm. The lower value of this indicator gives better performance than higher value. The results of the experiment are shown in the tables 3 and 4.

Table 3. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-2L
DTLZ1	$4.19e-02_{1.2e-03}$	$4.57e-02_{1.9e-03}$	$4.80e-02_{1.1e-03}$	$4.28e-02_{8.9e-04}$
DTLZ2	$7.50e-02_{2.5e-04}$	$7.94e-02_{1.5e-03}$	$9.94e-02_{1.0e-03}$	$7.00e-02_{2.8e-04}$
DTLZ3	$7.88e-02_{2.5e-03}$	$9.03e-02_{5.0e-03}$	$1.04e-01_{3.0e-03}$	$7.63e-02_{3.1e-03}$
DTLZ4	$6.69e-02_{2.8e-04}$	$7.07e-02_{1.2e-03}$	$8.88e-02_{1.0e-03}$	$6.11e-02_{3.9e-04}$

Table 4. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-2L
DTLZ1	$4.16e-02_{1.4e-03}$	$4.52e-02_{1.3e-03}$	$4.79e-02_{1.6e-03}$	$4.28e-02_{9.9e-04}$
DTLZ2	$7.49e-02_{3.3e-04}$	$7.90e-02_{1.8e-03}$	$9.92e-02_{1.4e-03}$	$7.00e-02_{3.8e-04}$
DTLZ3	$7.86e-02_{3.8e-03}$	$8.96e-02_{6.1e-03}$	$1.04e-01_{4.4e-03}$	$7.53e-02_{4.4e-03}$
DTLZ4	$6.69e-02_{5.1e-04}$	$7.02e-02_{1.2e-03}$	$8.86e-02_{1.7e-03}$	$6.11e-02_{4.9e-04}$

The four columns MOEADD-FS, MOEADD-RS, MOEADD-UD and MOEADD-2L in these tables represent the versions of MOEA/DD algorithm using FixedSum, RandomSum, uniform design and two-layer approaches respectively for WV generation. The dark-grey background in these tables show that MOEA/DD with two-layer approach gives best performance in three DTLZ problems and MOEADD-FS is superior in only the DTLZ1 problem.

However the problem with two-layer method is that it cannot work on a population size of more than 196 on 5-D problems. At the same time, a big issue in many-objective optimization is the requirement of larger population size (sec 2.2). Hence, after excluding two-layer method, we conducted next experiment with remaining three approaches on 5-D problems with a larger population size of 300. The results are shown in tables 5 and 6 which show the best performance of proposed FixedSum over remaining approaches. Hence it is concluded that overall FixedSum approach for WV generation works best on larger population sizes than other three approaches.

Table 5. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$5.26e-02_{9.5e-04}$	$6.18e-02_{8.3e-04}$	$6.43e-02_{1.1e-03}$
DTLZ2	$6.14e-02_{1.4e-04}$	$8.26e-02_{8.9e-04}$	$8.54e-02_{2.2e-03}$
DTLZ3	$7.64e-02_{1.6e-03}$	$9.82e-02_{2.3e-03}$	$1.06e-01_{3.2e-03}$
DTLZ4	$5.29e-02_{2.1e-04}$	$7.41e-02_{1.1e-03}$	$7.41e-02_{1.5e-03}$

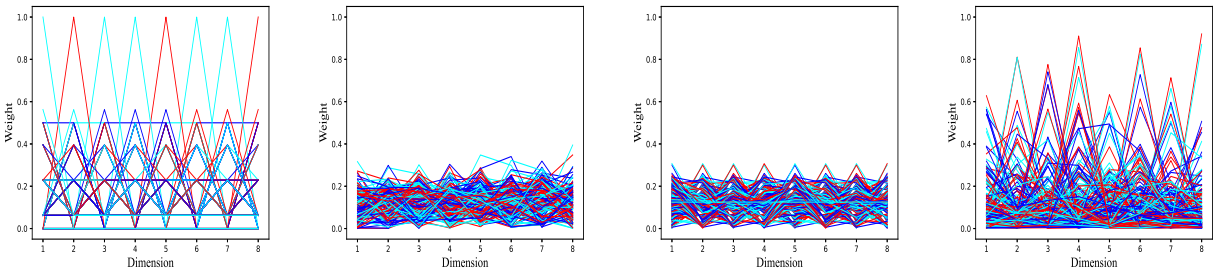
Table 6. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$5.22e-02_{1.4e-03}$	$6.19e-02_{1.1e-03}$	$6.40e-02_{1.8e-03}$
DTLZ2	$6.14e-02_{1.8e-04}$	$8.25e-02_{1.4e-03}$	$8.54e-02_{3.5e-03}$
DTLZ3	$7.61e-02_{2.4e-03}$	$9.78e-02_{3.4e-03}$	$1.05e-01_{4.0e-03}$
DTLZ4	$5.30e-02_{3.0e-04}$	$7.40e-02_{1.5e-03}$	$7.43e-02_{2.3e-03}$

5.3.3. Weight Vectors of dimension 8

Generating and plotting the weight vectors

The number of weight vectors for dimension 8 are again principally determined using two-layer method (sec 3.3) with different values of q for boundary and inner layers. Using q = 2, $N1 = \binom{2+8-1}{8-1} = 36$ weight vectors are chosen on the boundary layer whereas using q = 3, $N2 = \binom{3+8-1}{8-1} = 120$ weight vectors are chosen on the inner layer to get a total of $N = 156$ weight vectors. For comparison purpose, N is also set as 156 for RandomSum, FixedSum and uniform design methods. The δ for uniform design method is taken as 157 in this case. As a result, the WVs of dimension 8 obtained using the four approaches are shown in fig 12. A careful comparison of figures 12a) to 12d) shows that the WVs generated using FixedSum method are again more densely and uniformly spread (like D-5) as compared to the other three approaches. The WVs generated by two-layer method are sparsely located in space whereas the WVs produced by RandomSum and uniform design methods are not uniformly spread at all in the sense that the values obtained for each dimension are in a much smaller range as compared to the FixedSum method. Hence the RandomSum and uniform design methods do not cover WVs belonging to the boundary regions.

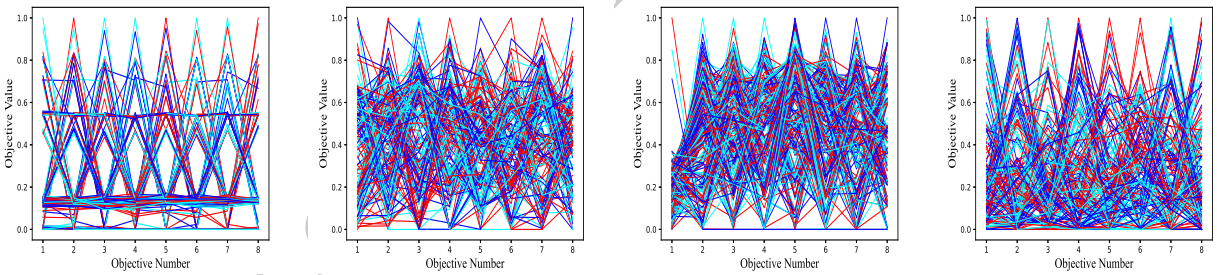


a). Two-layer method **b).** RandomSum method **c).** Uniform design method **d).** FixedSum method

Figure 12. Weight vectors of dimension 8

Plotting the results of DTLZ4

Like the 5-D case, we have solved the DTLZ4 problem with MOEA/DD algorithm by using the 8-D weight vectors obtained using the four methods. The resulting Pareto fronts are plotted using the PC plots in fig 13. A comparison of these plots in fig 13a) to 13d) clearly shows that the solutions obtained using WVs of proposed FixedSum algorithm are more uniformly spread over the entire Pareto front as compared to the solutions obtained through other WVs. As shown in fig 13a), the objective values for the solutions obtained through the two-layer method do not cover the entire domain and are missing certain ranges. However the solutions got through RandomSum in fig 13b) and uniform design in fig 13c) are more concentrated in the middle sections of Pareto front and the tradeoffs do not seem to be very wide. Finally, the solutions got through FixedSum method (fig 13d)) cover the entire range of objective values.



a). Two-layer method **b).** RandomSum method **c).** Uniform design method **d).** FixedSum method

Figure 13. Solutions of DTLZ4 problem using MOEA/DD

Quantitative comparison of different approaches

For comparing the performance of these four approaches on 8-D problems, we performed an experiment for solving the four DTLZ problems (DTLZ1 to DTLZ4) using MOEA/DD (with a population size of 156) and summarized the results using IGD+ indicator. The results of the experiment are shown in the tables 7 and 8.

The four columns MOEADD-FS, MOEADD-RS, MOEADD-UD and MOEADD-2L in these tables represent the versions of MOEA/DD algorithm using FixedSum, RandomSum, uniform design and two-layer approaches respectively for WV generation. The dark-grey background in these tables show that MOEA/DD with two-layer approach gives best performance in all four DTLZ problems as compared to

Table 7. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-2L
DTLZ1	$1.49e-03_{3.3e-04}$	$3.61e-03_{6.3e-04}$	$3.44e-01_{1.3e-01}$	$8.04e-04_{8.7e-05}$
DTLZ2	$1.04e-01_{6.8e-04}$	$1.72e-01_{2.5e-03}$	$3.07e-01_{1.1e-02}$	$9.31e-02_{1.2e-03}$
DTLZ3	$2.01e-03_{4.8e-04}$	$1.89e-02_{4.7e-03}$	$6.49e-01_{2.2e-01}$	$1.17e-03_{3.3e-04}$
DTLZ4	$8.46e-02_{1.6e-03}$	$1.84e-01_{2.8e-03}$	$2.54e-01_{1.2e-02}$	$6.29e-02_{4.1e-04}$

Table 8. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-2L
DTLZ1	$1.45e-03_{2.9e-04}$	$3.49e-03_{6.5e-04}$	$3.62e-01_{1.8e-01}$	$7.73e-04_{1.0e-04}$
DTLZ2	$1.04e-01_{8.2e-04}$	$1.72e-01_{3.9e-03}$	$3.05e-01_{1.2e-02}$	$9.31e-02_{1.5e-03}$
DTLZ3	$2.01e-03_{6.8e-04}$	$2.05e-02_{3.9e-03}$	$6.06e-01_{3.4e-01}$	$1.03e-03_{3.3e-04}$
DTLZ4	$8.44e-02_{2.7e-03}$	$1.84e-01_{4.1e-03}$	$2.55e-01_{1.5e-02}$	$6.29e-02_{5.4e-04}$

the other three approaches in achieving diversity and convergence of solutions whereas the FixedSum approach is runnerup in this case.

Again the problem with two-layer method is that it cannot work on a population size of more than 156 on 8-D problems. Owing to the requirement of large population size with many-objective optimization, we conducted next experiment, after excluding two-layer method, on 8-D problems with a population size of 300. The results are shown in tables 9 and 10 which show the best performance of proposed FixedSum over remaining approaches. Hence it is reiterated that overall FixedSum approach for WV generation works best on larger population sizes than other three approaches.

Table 9. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$1.41e-03_{5.2e-04}$	$4.94e-03_{2.8e-03}$	$2.57e-01_{8.5e-02}$
DTLZ2	$8.57e-02_{8.5e-04}$	$1.47e-01_{2.4e-03}$	$2.62e-01_{7.7e-03}$
DTLZ3	$9.62e-04_{2.7e-04}$	$4.70e-03_{4.4e-03}$	$3.79e-01_{1.3e-01}$
DTLZ4	$7.01e-02_{2.2e-03}$	$1.63e-01_{4.4e-03}$	$2.25e-01_{8.9e-03}$

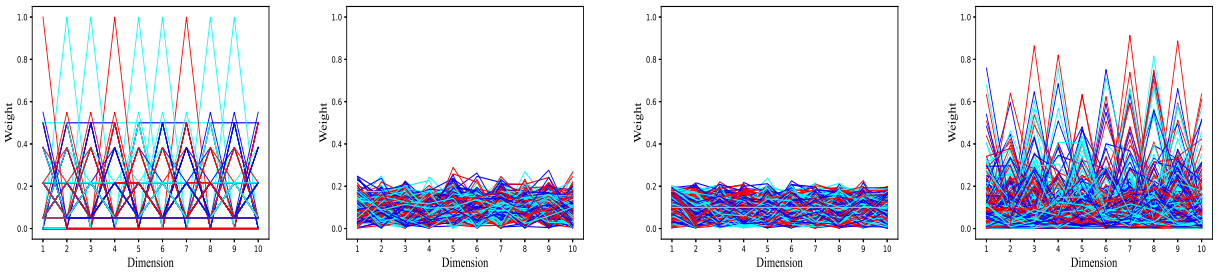
Table 10. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$1.31e-03_{6.8e-04}$	$4.31e-03_{4.1e-03}$	$2.57e-01_{1.4e-01}$
DTLZ2	$8.57e-02_{1.3e-03}$	$1.47e-01_{4.2e-03}$	$2.62e-01_{8.9e-03}$
DTLZ3	$8.66e-04_{4.0e-04}$	$2.82e-03_{7.4e-03}$	$3.93e-01_{2.1e-01}$
DTLZ4	$6.99e-02_{3.1e-03}$	$1.64e-01_{6.0e-03}$	$2.26e-01_{8.9e-03}$

5.3.4. Weight Vectors of dimension 10

Generating and plotting the weight vectors

The number of weight vectors for dimension 10 are again determined using two-layer method (sec 3.3) with different values of q for boundary and inner layers. Using q = 2, $N1 = \binom{2+10-1}{10-1} = 55$ WVs are chosen on the boundary layer whereas using q = 3, $N2 = \binom{3+10-1}{10-1} = 220$ WVs are chosen on the inner layer to get a total of $N = 275$ weight vectors. For comparison purpose, N is also set as 275 for FixedSum, RandomSum and uniform design methods. The δ for uniform design method is chosen as 71 in this case. As a result, the weight vectors of dimension 10 obtained using the four approaches are shown in fig 14.



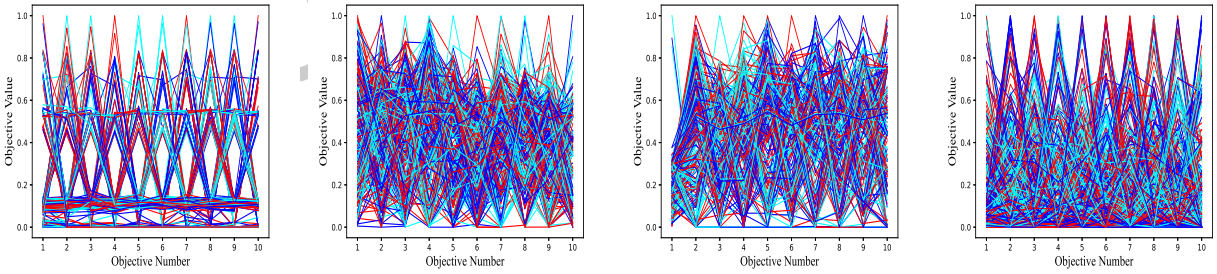
a). Two-layer method **b).** RandomSum method **c).** uniform design method **d).** FixedSum method

Figure 14. Weight vectors of dimension 10

A careful comparison of figures 14a) to 14d) shows that the WVs generated using FixedSum method are more densely and uniformly spread as compared to the other three approaches. The WVs generated by two-layer method are again sparsely located in space whereas the WVs produced by RandomSum and uniform design methods are not uniformly spread in the sense that the values obtained for each dimension are in a much smaller range as compared to the FixedSum method. Hence again the RandomSum and uniform design methods do not cover WVs belonging to the boundary regions.

Plotting the results of DTLZ4

Like the 5- and 8-D cases, we have solved the DTLZ4 problem with MOEA/DD by using the 10-D weight vectors obtained using the four methods. The resulting Pareto fronts are plotted using the PC plots in fig 15. A comparison of these plots in figures 15a) to 15d) clearly shows that the solutions obtained using WVs of proposed FixedSum algorithm are more uniformly spread over the entire Pareto front as compared to the other WVs. As shown in fig 15a), the objective values for the solutions obtained through the two-layer method again do not cover the entire range. However the solutions got through RandomSum in fig 15b) and uniform design method in fig 15c) are more concentrated in the middle regions of Pareto front. Finally, the solutions got through FixedSum method (see fig 15d)) cover the entire range of objective values.



a). Two-layer method **b).** RandomSum method **c).** Uniform design method **d).** FixedSum method

Figure 15. Solutions of DTLZ4 problem using MOEA/DD

Quantitative comparison of different approaches

For comparing the performance of the same four approaches on 10-D problems, we performed an experiment for solving the four DTLZ benchmark problems (DTLZ1 to DTLZ4) using MOEA/DD (with a population size of 275) and summarized the results using IGD+ indicator. The results of the experiment

are shown in the tables 11 and 12. It can be seen in these tables that MOEA/DD with two-layer approach gives best performance in three DTLZ problems whereas FixedSum works best for only DTLZ3 problem in achieving diversity and convergence of solutions.

Table 11. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-DD
DTLZ1	$1.02e-03_{1.3e-04}$	$6.73e-03_{1.4e-03}$	$1.44e-03_{1.0e-04}$	$1.00e-03_{1.1e-04}$
DTLZ2	$1.38e-01_{7.4e-04}$	$2.18e-01_{2.9e-03}$	$2.18e-01_{3.9e-03}$	$1.27e-01_{8.2e-04}$
DTLZ3	$1.37e-03_{3.0e-05}$	$3.11e-02_{8.5e-03}$	$2.16e-03_{5.4e-04}$	$1.39e-03_{8.3e-05}$
DTLZ4	$1.10e-01_{7.6e-04}$	$1.87e-01_{2.2e-03}$	$1.72e-01_{2.6e-03}$	$9.44e-02_{4.6e-04}$

Table 12. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD	MOEADD-DD
DTLZ1	$9.77e-04_{1.2e-04}$	$7.30e-03_{2.3e-03}$	$1.40e-03_{7.1e-05}$	$9.65e-04_{1.6e-04}$
DTLZ2	$1.38e-01_{1.1e-03}$	$2.18e-01_{3.0e-03}$	$2.19e-01_{6.2e-03}$	$1.27e-01_{1.0e-03}$
DTLZ3	$1.36e-03_{2.6e-05}$	$3.11e-02_{9.5e-03}$	$1.89e-03_{5.0e-04}$	$1.36e-03_{6.3e-05}$
DTLZ4	$1.09e-01_{1.1e-03}$	$1.87e-01_{3.0e-03}$	$1.73e-01_{4.1e-03}$	$9.44e-02_{7.3e-04}$

Recalling the inherent problem with two-layer method that it cannot work on a population size of more than 275 on 10-D problems. Hence, again after excluding two-layer method, we conducted next experiment with remaining three approaches on 10-D problems with a population size of 300. The results are shown in tables 13 and 14 which demonstrate the best performance of proposed FixedSum over remaining approaches on larger population sizes than the other approaches.

Table 13. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$1.56e-03_{2.7e-04}$	$1.42e-02_{1.7e-03}$	$2.62e-03_{2.2e-03}$
DTLZ2	$1.24e-01_{7.3e-04}$	$1.79e-01_{2.1e-03}$	$1.80e-01_{3.6e-03}$
DTLZ3	$1.00e-03_{1.1e-04}$	$6.25e-02_{1.5e-02}$	$2.70e-03_{2.9e-03}$
DTLZ4	$1.06e-01_{7.8e-04}$	$1.67e-01_{2.2e-03}$	$1.51e-01_{2.1e-03}$

Table 14. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$1.47e-03_{4.4e-04}$	$1.43e-02_{2.2e-03}$	$1.79e-03_{1.4e-04}$
DTLZ2	$1.24e-01_{9.5e-04}$	$1.80e-01_{2.8e-03}$	$1.81e-01_{5.0e-03}$
DTLZ3	$9.76e-04_{5.9e-05}$	$6.78e-02_{1.4e-02}$	$1.20e-03_{1.8e-03}$
DTLZ4	$1.06e-01_{9.9e-04}$	$1.66e-01_{2.6e-03}$	$1.51e-01_{2.8e-03}$

5.3.5. Weight Vectors of dimension 12

Generating and plotting the weight vectors

In this case, first of all, we demonstrate here that the number of weight vectors and hence the population size required for two-layer method is very large for m=12. Next we compare the FixedSum, RandomSum and uniform design methods to show the better results of the former over the later two methods.

Using the two-layer method, $N1 = \binom{2+12-1}{12-1} = 78$ weight vectors are chosen on the boundary layer with q = 2 whereas $N2 = \binom{3+12-1}{12-1} = 364$ weight vectors are chosen on the inner layer with q = 3 to get a total of $N = 442$ weight vectors. However, in this case, N=442 is a very large population size

resulting in the unbearably slow performance of any optimization algorithm for 12-D problem. Hence the use of two-layer method is ruled out for $m > 10$. Consequently, the three possible choices left are the FixedSum, RandomSum and uniform design methods.

Quantitative comparison of different approaches

We performed an experiment, for comparing these three approaches on 12-D problems, by solving the four DTLZ problems using MOEA/DD (with a population size of 300) and have summarized the results using IGD+ indicator. The results of the experiment are shown in the tables 15 and 16. It can be seen that MOEA/DD with FixedSum approach gives best performance in all four DTLZ problems as compared to MOEA/DD-RS and MOEA/DD-UD which proves the superiority of FixedSum method over other approaches.

Table 15. IGD+. Mean and Standard Deviation

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$9.75e-04_{1.7e-04}$	$5.65e-03_{7.1e-04}$	$3.93e-03_{1.7e-03}$
DTLZ2	$1.57e-01_{7.8e-04}$	$2.14e-01_{1.2e-03}$	$2.07e-01_{2.5e-03}$
DTLZ3	$1.08e-03_{1.1e-04}$	$7.58e-02_{5.0e-03}$	$3.77e-02_{2.7e-02}$
DTLZ4	$1.59e-01_{8.9e-04}$	$2.27e-01_{1.7e-03}$	$2.19e-01_{3.2e-03}$

Table 16. IGD+. Median and Interquartile Range

	MOEADD-FS	MOEADD-RS	MOEADD-UD
DTLZ1	$9.81e-04_{2.9e-04}$	$5.76e-03_{9.2e-04}$	$4.06e-03_{3.0e-03}$
DTLZ2	$1.57e-01_{1.1e-03}$	$2.14e-01_{1.5e-03}$	$2.07e-01_{4.0e-03}$
DTLZ3	$1.10e-03_{1.6e-04}$	$7.50e-02_{7.2e-03}$	$2.49e-02_{5.2e-02}$
DTLZ4	$1.59e-01_{1.3e-03}$	$2.27e-01_{2.7e-03}$	$2.19e-01_{4.3e-03}$

5.3.6. Comparison using the RODE algorithm

In this section, we have compared the proposed FixedSum approach with RandomSum and uniform design on a recently proposed many-objective algorithm RODE [34] on 5-D instances of DTLZ1 to DTLZ4 problems. The two-layer method is not considered due to the requirement of larger population size (i.e. 300 in this case). RODE is based on ranking dominance and employs weight vectors and opposition-based differential evolution (DE) to improve the diversity of solutions in a novel way. For conducting the experiment, we have chosen the $rand/1/bin$ strategy of DE with crossover ratio as 0.15 and scale factor taken as 1.5. The results are shown below in tables 17 and 18.

These results further substantiate the idea that FixedSum approach of WV generation is best for decomposition-based optimization algorithms especially in the many-objective scenario.

5.3.7. Statistical analysis of results

For further demonstrating the superiority of our results in sec 5.3.5, we have run two statistical tests on the collected experimental data.

The table 19 displays the outcomes of executing the Friedman test (for ranking of algorithms) on the results of tables 15 and 16 on the IGD+ metric. The lower the rank numeric value, the higher the rank. It demonstrates that the MOEADD-FS is ranked first in all experiments with DTLZ1 to DTLZ4 problems in sec 5.3.5 for 12-objective problems.

Table 17. IGD+. Mean and Standard Deviation

	RODE-FS	RODE-RS	RODE-UD
DTLZ1	$7.66e-02_{5.6e-03}$	$1.06e-01_{1.6e-01}$	$1.23e-01_{1.6e-01}$
DTLZ2	$6.80e-02_{8.9e-02}$	$1.83e-01_{1.2e-01}$	$2.43e-01_{8.5e-02}$
DTLZ3	$2.93e-01_{1.4e+00}$	$4.59e-01_{1.8e+00}$	$4.10e-01_{1.2e+00}$
DTLZ4	$6.39e-02_{8.3e-02}$	$1.50e-01_{1.1e-01}$	$1.88e-01_{1.1e-01}$

Table 18. IGD+. Median and Interquartile Range

	RODE-FS	RODE-RS	RODE-UD
DTLZ1	$7.90e-02_{9.5e-03}$	$8.06e-02_{8.4e-03}$	$8.14e-02_{6.9e-05}$
DTLZ2	$3.95e-04_{1.9e-01}$	$1.91e-01_{2.1e-01}$	$2.15e-01_{1.4e-01}$
DTLZ3	$8.73e-04_{2.0e-04}$	$1.91e-01_{1.9e-01}$	$1.91e-01_{9.5e-02}$
DTLZ4	$1.25e-05_{1.7e-01}$	$1.67e-01_{2.4e-01}$	$2.08e-01_{1.8e-01}$

Table 19. Average ranking of the algorithms

Algorithm	Ranking
MOEADD-FS	1.0
MOEADD-UD	2.0
MOEADD-RS	3.0

We have also applied the Wilcoxon rank-sum test at 5% level of significance to results of sec 5.3.5 which shows that MOEADD-FS is significantly better than MOEADD-RS and MOEADD-UD in all four DTLZ problems on IGD+ metric.

5.4. Effectiveness of FixedSum Method

In continuation of the discussion on experimentation about the proposed algorithm, we now sum up the performance of FixedSum. In a nutshell, the proposed strategy has been demonstrated to be effective in producing an arbitrary number of weight vectors which are distributed widely in m dimensional space. When used in the framework of a decomposition-based MOEA (sec 2.3), these WVs can be associated with each subproblem and each solution in order to produce a Pareto front with good convergence and diversity of solutions. Different scalarization techniques (e.g. weighted-sum, PBI) can be inserted into the frameworks to leverage the strength of WV method. Finally, the use of WVs generated by FixedSum will not only enhance the diversity but also improve the convergence of points because of the avoidance of premature convergence.

6. Conclusion

In this paper, we proposed a novel algorithm, FixedSum, for the generation of arbitrary number of WVs of any user-specified dimension and compared its results (graphically and analytically) with other methods on 2-, 5-, 8-, 10- and 12-D weight vectors. For further validation of our approach, we applied our weight vectors along with WVs of two other methods for solving the DTLZ problems using a popular decomposition-based algorithm MOEA/DD. All the results, including the statistical tests, demonstrate the improved spread ability (of weight vectors) of our approach as compared to the competing approaches. Also the complexity of FixedSum algorithm is $O(mn)$, which means, for a given m, complexity increases linearly with population size N. Especially for $m > 10$, the use of FixedSum approach

is proved to be indispensable because of poor performance of other approaches in this case. In a nutshell, the employment of FixedSum in decomposition-based multiobjective optimization will aid in improving the convergence and diversity of solutions in these approaches.

In future, we plan to fine tune our proposed algorithm for further uniform distribution of WVs and will study the impact of varying the parameter ϕ on the performance of FixedSum. Furthermore, we intend to extend our algorithm by introducing periodic WV adjustment for solving MOPs with irregular Pareto fronts [24, 27].

Declarations

"**Funding:** This work was supported by the MoST (Ministry of Science & Technology) endowment and NED University research grants."

Conflicts of interest/Competing interests: None

Availability of data and material: All data generated or analyzed during this study are included in this published article as different tables.

Code availability: All the code is developed in Java using jMetal framework and is available with the authors

Consent to participate: Not applicable

Ethics approval: Not applicable

Consent for publication: Not applicable

References

[1] Branke, J., Branke, J., Deb, K., Miettinen, K., and Slowiński, R. *Multiobjective optimization: Interactive and evolutionary approaches*, vol. 5252. Springer Science & Business Media, 2008.

[2] Chen, J., Ding, J., Tan, K. C., and Chen, Q. A decomposition-based evolutionary algorithm for scalable multi/many-objective optimization. *Memetic Computing 13* (2021), 413–432.

[3] Chen, X., Yin, J., Yu, D., and Fan, X. A decomposition-based many-objective evolutionary algorithm with adaptive weight vector strategy. *Applied Soft Computing 128* (2022), 109412.

[4] Coello, C. A. C., Lamont, G. B., Van Veldhuizen, D. A., et al. *Evolutionary algorithms for solving multi-objective problems*, vol. 5. Springer, 2007.

[5] Dai, C., and Wang, Y. A new decomposition based evolutionary algorithm with uniform designs for many-objective optimization. *Applied Soft Computing 30* (2015), 238–248.

[6] Das, I., and Dennis, J. E. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization 8*, 3 (1998), 631–657.

[7] Deb, K. *Multi-objective optimization using evolutionary algorithms*, vol. 16. John Wiley & Sons, 2001.

[8] Deb, K. *Multi-objective optimisation using evolutionary algorithms: an introduction*. Springer, 2011.

[9] Deb, K., and Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation 18*, 4 (2014), 577–601.

[10] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation 6*, 2 (2002), 182–197.

[11] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*. Springer, 2005, pp. 105–145.

[12] Fang, K.-T., and Wang, Y. *Number-theoretic methods in statistics*, vol. 51. CRC Press, 1993.

[13] He, L., Camacho, A., Nan, Y., Trivedi, A., Ishibuchi, H., and Srinivasan, D. Effects of corner weight vectors on the performance of decomposition-based multiobjective algorithms. *Swarm and Evolutionary Computation 79* (2023), 101305.

[14] Ishibuchi, H., Imada, R., Masuyama, N., and Nojima, Y. Two-layered weight vector specification in decomposition-based multi-objective algorithms for many-objective optimization problems. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (2019), IEEE, pp. 2434–2441.

[15] Ishibuchi, H., Masuda, H., Tanigaki, Y., and Nojima, Y. Modified distance calculation in generational distance and inverted generational distance. In *International conference on evolutionary multi-criterion optimization* (2015), Springer, pp. 110–125.

[16] Ishibuchi, H., and Murata, T. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 28*, 3 (1998), 392–403.

[17] Ishibuchi, H., Tsukamoto, N., and Nojima, Y. Evolutionary many-objective optimization: A short review. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on* (2008), IEEE, pp. 2419–2426.

[18] Ji, J.-Y., Tan, Z., Zeng, S., See-To, E. W., and Wong, M.-L. A surrogate-assisted evolutionary algorithm for seeking multiple solutions of expensive multimodal optimization problems. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2023).

[19] Ji, J.-Y., Tan, Z., Zeng, S., and Wong, M.-L. An ε-constrained multiobjective differential evolution with adaptive gradient-based repair method for real-world constrained optimization problems. *Applied Soft Computing 152* (2024), 111202.

[20] Ji, J.-Y., Yu, W.-J., Zhong, J., and Zhang, J. Density-enhanced multiobjective evolutionary approach for power economic dispatch problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems 51*, 4 (2019), 2054–2067.

[21] Ji, J.-Y., Zeng, S., and Wong, M. L. ε-constrained multiobjective differential evolution using linear population size expansion. *Information Sciences 609* (2022), 445–464.

[22] Kremmel, T., Kubalík, J., and Biffl, S. Software project portfolio optimization with advanced multiobjective evolutionary algorithms. *Applied Soft Computing 11*, 1 (2011), 1416–1426.

[23] Kukkonen, S., and Lampinen, J. Gde3: The third evolution step of generalized differential evolution. In *2005 IEEE congress on evolutionary computation* (2005), vol. 1, IEEE, pp. 443–450.

[24] Li, G., Wang, G.-G., and Xiao, R.-B. A novel adaptive weight algorithm based on decomposition and two-part update strategy for many-objective optimization. *Information Sciences 615* (2022), 323–347.

[25] Li, K., Deb, K., Zhang, Q., and Kwong, S. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation 19*, 5 (2014), 694–716.

[26] Li, X., Zhan, J., Pan, F., Lv, T., and Wang, S. A multi-objective optimization model of urban passenger transportation structure under low-carbon orientation considering participating subjects. *Environmental Science and Pollution Research 30*, 54 (2023), 115839–115854.

[27] Liu, Y., Hu, Y., Zhu, N., Li, K., Zou, J., and Li, M. A decomposition-based multiobjective evolutionary algorithm with weights updated adaptively. *Information Sciences 572* (2021), 343–377.

[28] López Jaimes, A., and Coello Coello, C. A. Some techniques to deal with many-objective problems. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers* (2009), pp. 2693–2696.

[29] Miettinen, K. *Nonlinear multiobjective optimization*, vol. 12. Springer Science & Business Media, 2012.

[30] Moustafa, R. E. Parallel coordinate and parallel coordinate density plots. *Wiley Interdisciplinary Reviews: Computational Statistics 3*, 2 (2011), 134–148.

[31] Ngatchou, P., Zarei, A., and El-Sharkawi, A. Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems* (2005), IEEE, pp. 84–91.

[32] Prajapati, A. A comparative study of many-objective optimizers on large-scale many-objective software clustering problems. *Complex & Intelligent Systems 7*, 2 (2021), 1061–1077.

[33] Qasim, S. Z., and Ismail, M. A. Research problems in search-based software engineering for many-objective optimization. In *Innovations in Electrical Engineering and Computational Technologies (ICIEECT), 2017 International Conference on* (2017), IEEE, pp. 1–6.

[34] Qasim, S. Z., and Ismail, M. A. Rode: Ranking-dominance-based algorithm for many-objective optimization with opposition-based differential evolution. *ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING* (2020).

[35] Qasim, S. Z., and Ismail, M. A. Docea/d: Dual-operator-based constrained many-objective evolutionary algorithm based on decomposition. *Cluster Computing* (2022), 1–19.

[36] Qasim, S. Z., and Ismail, M. A. Mosa/d: Multi-operator evolutionary many-objective algorithm with self-adaptation of parameters based on decomposition. *Evolutionary Intelligence* (2022), 1–23.

[37] Ramirez, A., Romero, J. R., and Ventura, S. A survey of many-objective optimisation in search-based software engineering. *Journal of Systems and Software 149* (2019), 382–395.

[38] Saxena, D. K., Mittal, S., Kapoor, S., and Deb, K. A localized high-fidelity-dominance based many-objective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation* (2022).

[39] Yacoubi, S., Manita, G., Chhabra, A., Korbaa, O., and Mirjalili, S. A multi-objective chaos game optimization algorithm based on decomposition and random learning mechanisms for numerical optimization. *Applied Soft Computing* (2023), 110525.

[40] Zhang, Q., and Li, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation 11*, 6 (2007), 712–731.

[41] Zheng, W., Tan, Y., Meng, L., and Zhang, H. An improved moea/d design for many-objective optimization problems. *Applied Intelligence 48*, 10 (2018), 3839–3861.

[42] Zitzler, E., Laumanns, M., and Thiele, L. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report 103* (2001).