What Weights Work for You? Adapting Weights for Any Pareto Front Shape in Decomposition-based Evolutionary Multi-Objective Optimisation

Article *in* Evolutionary Computation · September 2017

DOI: 10.1162/evco_a_00269

Miqing Li
University of Birmingham

What Weights Work for You? Adapting Weights for Any Pareto Front Shape in Decomposition-based Evolutionary Multi-Objective Optimisation

Miqing Li, and Xin Yao[1]

[1]CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U. K.
*Email: limitsing@gmail.com, x.yao@cs.bham.ac.uk

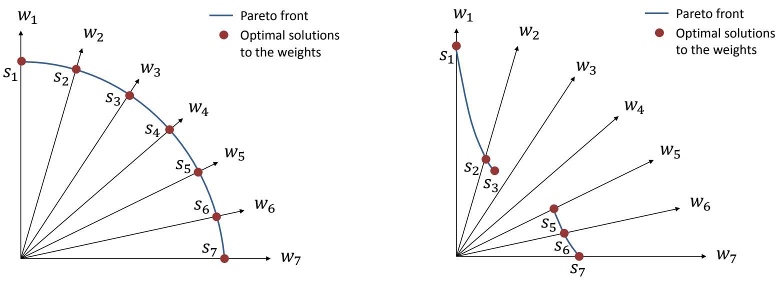
Abstract: The quality of solution sets generated by decomposition-based evolutionary multiobjective optimisation (EMO) algorithms depends heavily on the consistency between a given problem's Pareto front shape and the specified weights' distribution. A set of weights distributed uniformly in a simplex often lead to a set of well-distributed solutions on a Pareto front with a simplex-like shape, but may fail on other Pareto front shapes. It is an open problem on how to specify a set of appropriate weights without the information of the problem's Pareto front beforehand. In this paper, we propose an approach to adapt the weights during the evolutionary process (called AdaW). AdaW progressively seeks a suitable distribution of weights for the given problem by elaborating five parts in the weight adaptation — weight generation, weight addition, weight deletion, archive maintenance, and weight update frequency. Experimental results have shown the effectiveness of the proposed approach. AdaW works well for Pareto fronts with very different shapes: 1) the simplex-like, 2) the inverted simplex-like, 3) the highly nonlinear, 4) the disconnect, 5) the degenerated, 6) the badly-scaled, and 7) the high-dimensional.

Keywords: Multi-objective optimisation, many-objective optimisation, evolutionary algorithms, decomposition-based EMO, weight adaptation

1 Introduction

Decomposition-based evolutionary multiobjective optimisation (EMO) is a general-purpose algorithm framework. It decomposes a multi-objective optimisation problem (MOP) into a number of single-objective optimisation sub-problems on the basis of a set of weights (or called weight vectors) and then uses a search heuristic to optimise these sub-problems simultaneously and cooperatively. Compared with conventional Pareto-based EMO, decomposition-based EMO has clear strengths, e.g., providing high selection pressure toward the Pareto front [1], being easy to work with local search operators [2–4], owning high search ability for combinatorial MOPs [5–8], and being capable of dealing with MOPs with many objectives [9–12] and MOPs with a complicated Pareto set [13–16].

A key feature in decomposition-based EMO is that the diversity of the evolutionary population is controlled explicitly by a set of weight vectors (or a set of reference directions/points determined by this weight vector set). Each weight vector corresponds to one subproblem, ideally associated with one solution in the population; thus, diverse weight vectors may lead to different Pareto optimal solutions. In general, a well-distributed solution set can be obtained if the set of weight vectors and the



(a) On a concave Pareto front (b) On a disconnected, convex Pareto front

Figure 1: An example that uniformly distributed weights may lead to different distributions of optimal solutions. (a) Solutions s_1 to s_7 are the optimal solutions of weights w_1 to w_7, respectively. (b) Solutions s_1, s_2, s_3, s_6 and s_7 are the optimal solutions of w_1, w_2, w_3, w_6 and w_7, respectively, while solution s_5 is the optimal solution of w_4 and w_5.

Pareto front of a given problem share the same/similar distribution shape. In many existing studies, the weight vectors are predefined and distributed uniformly in a unit simplex. This specification can make decomposition-based algorithms well-suited to MOPs with a "regular" (i.e., simplex-like) Pareto front, e.g., a triangle plane or a sphere. Figure 1(a) shows such an example, where a set of uniformly-distributed weight vectors correspond to a set of uniformly-distributed Pareto optimal solutions.

However, when the shape of an MOP's Pareto front is far from the standard simplex, a set of uniformly distributed weight vectors may not result in a uniform distribution of Pareto optimal solutions. On MOPs with an "irregular" Pareto front (e.g., disconnected, degenerate, inverted simplex-like or badly-scaled), decomposition-based algorithms appear to struggle [17–20]. In such MOPs, some weight vectors may have no intersection with the Pareto front. This could lead to several weight vectors corresponding to one Pareto optimal solution. In addition, there may exist a big difference of distance between adjacent Pareto optimal solutions (obtained by adjacent weight vectors) in different parts of the Pareto front. This is due to the inconsistency between the shape of the Pareto front and the shape of the weight vector distribution. Overall, no intersection between some weight vectors and the Pareto front may cause the number of obtained Pareto optimal solutions to be smaller than that of weight vectors, while the distance difference regarding adjacent solutions in different parts of the Pareto front can result in a non-uniform distribution of solutions.

Figure 1(b) gives an example that a set of Pareto optimal solutions are obtained by a set of uniformly-distributed weight vectors on an "irregular" Pareto front. As can be seen, weight vectors w_3 and w_4 have no intersection with the Pareto front, and weight vectors w_4 and w_5 correspond to only one Pareto optimal solution (s_5). In addition, the obtained Pareto optimal solutions are far from being uniformly distributed, e.g., the distance between s_1 and s_2 being considerably farther than that between other adjacent solutions.

The above example illustrates the difficulties of predefining weight vectors in decomposition-based EMO. It could be very challenging (or even impossible) to find a set of optimal weight vectors beforehand for any MOP, especially in real-world scenarios where the information of a problem's Pareto front is often unknown.

A potential solution to this problem is to seek adaptation approaches that can progressively modify the weight vectors according to the evolutionary population during the optimisation process. Several interesting attempts have been made along this line [21]. A detail review of these adaptation approaches will be presented in next section.

Despite the potential advantages of these adaptation approaches for "irregular" Pareto fronts, the problem is far from being completely resolved. On one hand, varying the weight vectors which are pre-set and ideal for problems with "regular" Pareto fronts may compromise the performance of an algorithm on these problems themselves. On the other hand, varying the weight vectors materially changes the subproblems over the course of the optimisation, which could significantly deteriorate the convergence of the algorithm [22]. Overall, as pointed out in [19, 23], how to set the weight vectors is still an open question; the need for effective methods is pressing.

In this paper, we present an adaptation method (called AdaW) to progressively adjust the weight vectors during the evolutionary process. AdaW updates the weight vectors periodically based on the information produced by the evolving population itself, and then in turn guides the population by these weight vectors which are of a suitable distribution for the given problem. Specifically, an archive is used to find out potential undeveloped, but promising weight vectors. Then a weight vector deletion operation is used to remove existing unpromising weight vectors or/and weight vectors associated with the crowded solutions in the population.

The rest of the paper is organised as follows. Section II reviews related work. Section III is devoted to the proposed adaptation method, including the basic idea and the five key issues of this adaptation. Experimental results are presented in Section IV. Finally, Section V concludes the paper.

2 Related Work

A basic assumption in decomposition-based EMO is that the diversity of the weight vectors will result in the diversity of the Pareto optimal solutions. This motivates several studies on how to generate a set of uniformly distributed weight vectors [21], such as the simplex-lattice design [24], two-layer simplex lattice design [25], multi-layer simplex lattice design [26], uniform design [27], and a combination of the simplex-lattice design and uniform design [28]. A weakness of such systematic weight generators is that the number of generated weight vectors is not flexible, especially in a high-dimensional space. This contrasts with the uniform random sampling method [29] which can generate an arbitrary number of weight vectors for any dimension. In addition, some work has shown that if the geometry of the problem is known *in priori* then the optimal distribution of the weight vectors for a specific scalarizing function can be readily identified [30, 31].

The variety of weight generators gives us ample alternatives in initialising the weight vectors, each of which provides an explicit way of specifying a set of particular search directions in decomposition-based algorithms. However, the premise that these weight generators work well is the Pareto front of the problem sharing the simplex-like regular shape. An "irregular" Pareto front (e.g., disconnected, degenerate, inverted simplex-like or badly-scaled) may make these weight generators struggle, in which multiple weight vectors may correspond to one single point. This leads to a waste of computational resources, and further renders the algorithm's performance inferior.

An intuitive solution to this problem is to adaptively update the weight vectors during the optimisation process. Several interesting attempts have been made along this line. In early studies [5, 29, 32], researchers considered randomly-generated weight vectors (search directions) at each generation. This can make more computational cost allocated on the area around the nondominated solutions found so far [33]. Recently, Li et al. [33] presented that this strategy could be helpful in dealing with MOPs having an irregular Pareto front. They introduced an external population, which is used to store promising solutions by the ϵ dominance relation [34], to help generating the random weights.

The main trend of the weight adaptation has been on attempts to add/delete the weight vectors in sparse/dense regions in order to diversify solutions. For example, in [35] the authors periodically adjusted each weight to increase the distance of its corresponding solutions to others. In [36], Jiang et al. presented an adaptive weight adjustment method which samples the regression curve of the weight vectors on the basis of an external archive. Gu et al. [37] used the equidistant interpolation to periodically update the weight vectors on the estimated Pareto front. Later, they proposed a weight adaptation method by training an self-organising map according to the current solutions [38]. Jain and Deb [39] introduced an adaptive version of NSGA-III [25] (A-NSGA-III) for irregular Pareto fronts. In A-NSGA-III, an $(m-1)$-simplex of reference points (weight vectors) centred around a crowded reference point are added in the update process, and the reference points which are not associated with any of the solutions are deleted. Qi et al. [17] proposed an adaptive weight adjustment (MOEA/D-AWA) strategy for irregular Pareto fronts and integrated the strategy within MOEA/D-DRA [40]. MOEA/D-AWA has two phases: first a set of pre-set weight vectors are used until the algorithm is considered to approach the Pareto front, and then the weight vectors are adjusted periodically by removing the most crowded weight vectors and adding new weight vectors in sparse regions. Liu et al. [41, 42] proposed an adaptive weight update in MOEA/D-M2M [14] for degenerate Pareto fronts according to the angle among solutions (as a similar measure) in the evolutionary population. Cheng et al. [43,44] adapted the reference vectors (weight vectors) to guide the evolutionary search. In the presented reference vector guided evolutionary algorithm (RVEA), two reference vector adaptations are conducted to deal with badly-scaled Pareto fronts and irregular fronts respectively. Very recently, Zhang et al. [45] designed a weight vector adaptation via a linear interpolation for bi-objective optimisation problems with a discontinuous Pareto front. Wang et al. [46] considered both the ideal and nadir points to update the weight vectors in MOEA/D for a better distribution. Cai et al. [47] proposed two types of weight (direction) vector adjustments for many-objective problems, with one aiming at the number of the direction vectors and the other aiming at the positions of the direction vectors. Asafuddoula et al. [48] adapted the weight vectors on the basis of information collected over a "learning period", and stored the original weights which had been removed for the future use. In addition, some researchers introduced a set of weight vectors into non-decomposition-based EMO, and conducted the weight vector adaptation for Pareto-based search [49, 50] and indicator-based search [51]. And some other researchers adaptively adjusted the search directions according to the distribution of the evolutionary population, which can also be seen as an adaptation of weight vectors [52, 53].

In spite of the above progresses, challenges facing the weight adaptation remain:

- Adapting the weight vectors affects the convergence of the algorithm. Varying the weight vectors essentially changes the subproblems. After a change of the subproblems, their associated solutions need to readjust the search directions. This could lead to the solutions to *wander* in the objective space during the optimisation process [22].

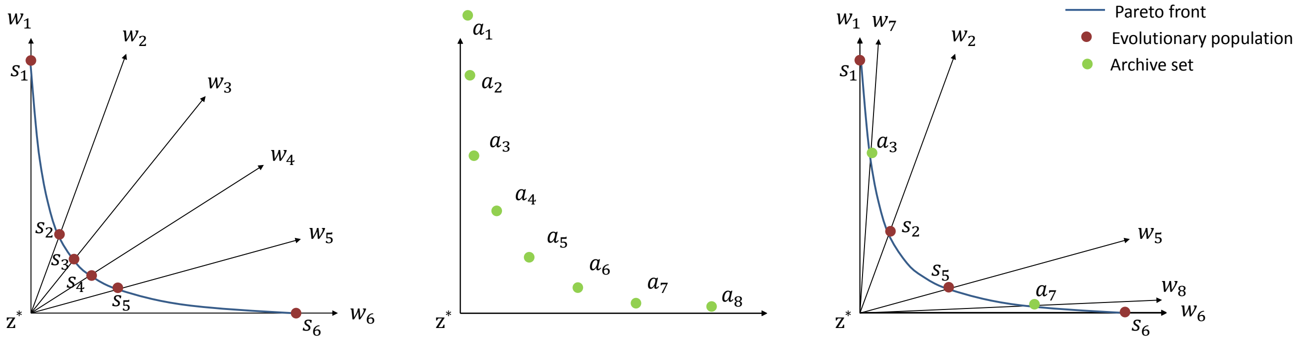
- Adapting the weight vectors may compromise the performance of the algorithm on regular Pareto fronts. Even for a regular Pareto front, there may exist multiple weight vectors corresponding to one single solution during the evolutionary process. As such, a change of the weight vectors which were already ideal for the considered regular Pareto front may lead solutions towards wrong search directions.

- It is difficult to adapt the weight vectors for different Pareto fronts. Many weight adaptation methods are designed or suitable for only certain types of Pareto fronts. The variety of Pareto fronts (disconnected, degenerate, inverted simplex-like, badly-scaled, highly-nonlinear, or/and high-dimensional) is a challenge to any adaptation method.

3 The Proposed Algorithm

3.1 Basic Idea

When optimising an MOP, the current nondominated solutions (i.e., the best solutions found so far) during the evolutionary process can indicate the evolutionary status [41, 54]. The nondominated solution set, with the progress of the evolution, gradually approximates the Pareto front, thus being likely to reflect the shape of the Pareto front provided that it is well maintained. Despite that such



(a) Before update (b) A well-maintained archive (c) After update

Figure 2: An illustration of updating the weight vectors of the population by the aid of a well-maintained archive set of nondominated solutions.

a set probably evolves slowly in comparison with the evolutionary population which is driven by the scalarizing function in decomposition-based algorithms, the set may be able to provide new search directions that are unexplored by the scalarizing function-driven population.

Figure 2 gives an illustration of updating the search directions (weight vectors) of the population by the aid of a well-maintained archive set of nondominated solutions. As can be seen, before the update a set of uniformly distributed weight vectors correspond to a poorly distributed population along the Pareto front. In the update, the two solutions from the archive (a_3 and a_7) whose areas are not explored well in the population are added (Figure 2(c)) and their corresponding weights are considered as new search directions to guide the evolution (w_7 and w_8). In addition, the weight vectors that are associated with crowded solutions (s_3 and s_4) in the population are deleted. Then, a new population is formed with poorly-distributed weight vectors but well-distributed solutions.

The above is the basic idea of the weight vector adaptation in our proposed work. However, materialising it requires a proper handling of several important issues. They are

- How to maintain the archive?
- Which solutions from the archive should enter the evolutionary population to generate new weight vectors?
- How to generate weight vectors on the basis of these newly-entered solutions?
- Which old weight vectors in the population should be deleted?
- What is the frequency of updating the weight vectors? i.e., how long should we allow the population to evolve by the current weight vectors?

In next five subsections, we will describe in sequence how we handle these issues, followed by the main framework of the algorithm.

3.2 Archive Maintenance

In AdaW, an archive with a pre-set capacity is to store the nondominated solutions produced during the evolutionary process. When the number of solutions in the archive exceeds the capacity, a maintenance mechanism is used to remove some solutions with poor distribution. Here, we consider the population maintenance method in [18]. This method which iteratively deletes the solution having the biggest crowding degree in the set can preserve a set of representative nondominated solutions [18]. The crowding degree of a solution is estimated by considering both the number and location of its neighbors in a niche. Formally, the crowding degree of a solution p in the set A is defined as

$$D(p) = 1 - \prod_{q \in A, q \neq p} R(p, q) \tag{1}$$

$$R(p, q) = \begin{cases} d(p, q)/r \ , & \text{if } d(p, q) \leq r \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

where $d(p, q)$ denotes the Euclidean distance between solutions p and q, and r is the radius of the niche, set to be the median of the distances from all the solutions to their kth nearest solution in the set. Note that similar to [18] all the objectives are normalised with respect to their minimum and maximum in the considered set in AdaW.

It is worth mentioning that there are two slight differences of the settings from that in [18]. First, parameter k for the kth nearest neighbor was set to 3 in [18], while here k is set to the number of objectives of the problem. This leads to less parameters required by the algorithm. Second, the median, instead of the average in [18], of the distances from all the solutions to their kth nearest neighbor is considered. This could alleviate the effect of the dominance resistant solutions (DRS), i.e., the solutions with a quite poor value in some objectives but with (near) optimal values in some other objectives [55].

3.3 Weight Addition

In AdaW, we aim to add weight vectors (into the evolutionary population) whose search directions/areas are *undeveloped* and *promising*. Both criteria can be measured by contrasting the evolutionary population with the archive set. For the former, if the niche in which a solution of the archive is located in has no solution in the evolutionary population, it is likely that niche is undeveloped. For the latter, if a solution of the archive performs better on its search direction (weight vector) than any solution of the evolutionary population, it is likely that the niche of that solution is promising.

Specifically, to find out the solutions whose niche is undeveloped by the evolutionary population, we consider the niche size which is determined by the archive itself. That is, the radius of the niche is set to the median of the distances from all the solutions to their closest solution in the archive. After finding out these candidate solutions whose niche are not developed by the evolutionary population, we then consider whether they are promising or not. First, we obtain their corresponding weight vectors (which will be detailed in next section). Then for each of these weight vectors, we find its neighbouring weight vectors[1] in the evolutionary population, and further determine the solutions associated with the neighboring weight vectors. Finally we compare these solutions with the candidate solution on the basis of the candidate solution's weight vector. Formally, let q be one of the candidate solutions in the archive and w_q be its corresponding weight vector. Let w_p be one of the neighbouring weight vectors of w_q in the evolutionary population, and p be the solution associated with w_p in the evolutionary population. We define that q outperforms p on the basis of w_q, if

$$g(q, w_q) < g(p, w_q) \tag{3}$$

where $g()$ is the considered scalarizing function, or

$$g(q, w_q) = g(p, w_q) \ \text{ and } \ \sum_{i=1}^{m} f_i(q) < \sum_{i=1}^{m} f_i(p) \tag{4}$$

where $f_i(q)$ denotes the objective value of q in the ith objective and m is the number of objectives. If q outperforms all of the neighboring solutions on the basis of its weight vector, then q will enter

[1]The definition of neighbouring weight vectors is based on that in decomposition-based EMO

the evolutionary population, along with its search direction (weight vector). After that, the neighbouring information of q's weight vector in the evolutionary population (i.e., the solutions that the neighbouring weight vectors corresponding to) is updated by q.

3.4 Weight Generation

Given a reference point, the optimal weight vector to a solution (e.g., w_7 to a_3 in Figure 2(c)) with respect to the Tchebycheff scalarizing function can be easily generated. This is already a frequently used approach in the weight vector adaptation [17, 37].

Formally, let $z^* = (z_1^*, z_2^*, ..., z_m^*)$ be the reference point[2] and $w = (\lambda_1, \lambda_2, ..., \lambda_m)$ be the optimal weight vector to a solution q. Then it holds that

$$\frac{f_1(q) - z_1^*}{\lambda_1} = \frac{f_2(q) - z_2^*}{\lambda_2} = \cdots = \frac{f_m(q) - z_m^*}{\lambda_m} \tag{5}$$

Since $\lambda_1 + \lambda_2 + ... + \lambda_m = 1$, we have

$$w = (\lambda_1, ..., \lambda_m) = (\frac{f_1(q) - z_1^*}{\sum_{i=1}^m f_i(q) - z_i^*}, ..., \frac{f_m(q) - z_m^*}{\sum_{i=1}^m f_i(q) - z_i^*}) \tag{6}$$

3.5 Weight Deletion

After the weight addition operation, AdaW needs to delete some weight vectors in the evolutionary population to keep the number of the weight vectors unchanged (i.e., back to the predefined population size N). In view of that ideally each weight vector is associated with one distinct solution in decomposition-based EMO, we consider the situation that multiple weight vectors share one solution (e.g., weight vectors w_4 and w_5 sharing solution s_5 in Figure 1(b)). Specifically, we find out the solution who is shared by the most weight vectors in the population. In these weight vectors, we delete the one whose scalarizing function value is the worst. Formally, let p be the solution shared by the most weight vectors $w_1, w_2, ..., w_n$ out of the population. Then the weight vector to be deleted is

$$\underset{1 \leq i \leq n}{\text{argmax}} \; g(p, w_i) \tag{7}$$

In addition, there may exist several solutions in the population having the same largest number of weight vectors. For this case, we compare their worst weight vectors — the weight vector having the highest (worst) scalarizing function value will be deleted.

The above deletion operation is repeated until the number of the weight vectors restores (i.e., back to N). However, there may exist one situation that even when every solution in the population corresponds to only one weight vector, the number of the weight vectors still exceeds N. In this situation, we use the same diversity maintenance method of Section III-B to iteratively delete the most crowded solution (along with its weight vector) in the population until the number of the weight vectors reduces to N.

3.6 Weight Update Frequency

The timing and frequency of updating the weight vectors of the evolutionary population play an important role in weight vector adaptation methods. Since varying the weight vectors essentially changes the subproblems to be optimised, a frequent change can significantly affect the convergence of the algorithm [22]. In AdaW, the weight update operation is conducted every 5% of the total

[2] The reference point in decomposition-based algorithms is often set to be equal to or slightly smaller than the best value found so far [17, 56]; here we set it to 10^{-4} smaller than the best value, following the suggestions in [57].

Algorithm 1: The Algorithm AdaW

Require: N (size of the evolutionary population P, i.e., size of the weight vector set W), N_A (size of the archive set A), T (neighbourhood size), Gen_{max} (maximum generations in the evolution).

1: Initialise the population P and a set of weight vectors W.
2: Calculate the reference point according to P.
3: Determine the neighbours of each weight vector of W.
4: Associate the weight vectors with the solutions in the population randomly.
5: Put the nondominated solutions of P into the archive A.
6: $Gen \leftarrow 1$.
7: **while** $Gen < Gen_{max}$ **do**
8:   **for** each subproblem (weight vector) $w \in W$ **do**
9:     Determine the mating pool by selecting the solutions associated with the neighbours of w or from the whole population in a probability.
10:     Generate a new solution p by using the variation operators on the solutions in the mating pool.
11:     Update the reference point by p.
12:     Update by p the solutions associated with the neighbours of w or from the whole population in a probability.
13:     **if** $\nexists q \in A, q \prec p$ **then**
14:       $A \leftarrow A \cup p$
15:       $A \leftarrow A / \{q \in A \mid p \prec q\}$
16:     **end if**
17:   **end for**
18:   **if** $|A| > N_A$ **then**
19:     Maintain the archive A (Section III-B).
20:   **end if**
21:   **if** $Gen = Gen_{max} \times 5\% \wedge Gen < Gen_{max} \times 90\%$ **then**
22:     Generate and find the promising, undeveloped weight vectors, and add them into W (Section III-C and Section III-D).
23:     Delete the poorly-performed weight vectors and/or the weight vectors associated with the crowded solutions until the size of W is reduced to N (Section III-E).
24:     Update the neighbours of each weight vector of W.
25:   **end if**
26:   $Gen \leftarrow Gen + 1$.
27: **end while**
28: **return** P

generations/evaluations. In addition, when the algorithm approaches the end of the optimisation process, a change of the weight vectors may lead to the solutions evolving insufficiently along those specified search directions (weight vectors). Therefore, AdaW does not change the weight vectors during the last 10% generations/evaluations.

3.7 Algorithm Framework

Algorithm 1 gives the main procedure of AdaW. As can be seen, apart from the weight vector update (Steps 21–25) and archive operations (Steps 5, 13–16 and 18–20), the remaining steps are the common steps in a generic decomposition-based algorithm. Here, we implemented them by a widely-used MOEA/D version in [13]. Next, we analyse the time complexity of the proposed algorithm.

Additional computational costs of AdaW (in comparison with the basic MOEA/D) are from the archive operations and weight vector update. In one generation of AdaW, updating the archive (Steps 13–16) requires $O(mNN_A)$ comparisons, where m is the number of the problem's objectives, N is the population size, and N_A is the archive size. Maintaining the archive (Steps 18–20) requires $O(mN_A^2)$ comparisons [18]. The computational cost of the weight vector update is governed by three operations, weight vector addition (Step 22), weight vector deletion (Step 23), and neighbouring weight vector update (Step 24). In the weight vector addition, undeveloped solutions are first determined. This includes calculating the niche radius and finding out undeveloped solutions, which require $O(mN_A^2)$ and $O(mNN_A)$ comparisons, respectively. After L undeveloped solutions are found, we check if they are promising by comparing them with the solutions that their neighbouring weight vectors corresponding to. The computational complexity of finding the neighbours of the L weight vectors is bounded by $O(mLN)$ or $O(TLN)$ (T denotes the neighbourhood size), whichever is greater. Then, checking if these L solutions are promising requires $O(mTL)$ comparisons. In the weight vector deletion, considering the situation that one solution shared by multiple weight vectors requires $O(LN)$ comparisons and removing the weight vectors which are associated with crowded solutions requires $O(m(L+N)^2)$ comparisons [18]. Finally, after the weight vector deletion completes, updating the neighbours of each weight vector in the population requires $O(mN^2)$ or $O(TN^2)$ comparisons, whichever is greater.

To sum up, since $O(N) = O(N_A)$ and $0 \leqslant L \leqslant N_A$, the additional computational cost of AdaW is bounded by $O(mN^2)$ or $O(TN^2)$ whichever is greater, where m is the number of objectives and T is the neighbourhood size. This governs the proposed algorithm given a lower time complexity ($O(mTN)$) required in the basic MOEA/D [1].

4 Results

Three state-of-the-art weight vector adaptation approaches, A-NSGA-III [39], RVEA [44] and MOEA/D-AWA [17], along with the baseline MOEA/D [13], were considered as peer algorithms[3] to evaluate the proposed AdaW. These adaptations had been demonstrated to be competitive on MOPs with various Pareto fronts. In MOEA/D, the Tchebycheff scalarizing function was used in which "multiplying the weight vector" was replaced with "dividing the weight vector" in order to obtain more uniform solutions [17, 25].

In view of the goal of the proposed work, we selected 17 test problems with a variety of representative Pareto fronts from the existing problem suites [25, 39, 59–63]. According to the properties of their Pareto fronts, we categorised the problems into seven groups to challenge the algorithms in balancing the convergence and diversity of solutions. They are

1. problems with a simple-like Pareto front: DTLZ1, DTLZ2 and convex DTLZ2 (CDTLZ2).
2. problems with an inverted simple-like Pareto front: inverted DTLZ1 (IDTLZ1) and inverted DTLZ2 (IDTLZ2).
3. problems with a highly nonlinear Pareto front: SCH1 and FON.
4. problems with a disconnect Pareto front: ZDT3 and DTLZ7.
5. problems with a degenerate Pareto front: DTLZ5 and VNT2.
6. problems with a badly-scaled Pareto front: scaled DTLZ1 (SDTLZ1), scaled DTLZ2 (SDTLZ2) and SCH2.
7. problems with a high-dimensional Pareto front: 10-objective DTLZ2 (DTLZ2-10), 10-objective inverted DTLZ1 (IDTLZ1-10) and DTLZ5(2,10).

All the problems were configured as described in their original papers [25, 39, 59–63].

To compare the performance of the algorithms, the inverted generational distance (IGD) [1, 64] was used. IGD, which measures the average distance from uniformly distributed points along the Pareto

[3]The codes of all the peer algorithms were from `http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html` [58].

front to their closest solution in a set, can provide a comprehensive assessment of the convergence and diversity of the set. In addition, for a visual understanding of the search behaviour of the five algorithms, we also plotted their final solution set in a single run on all the test problems. This particular run was associated with the solution set which obtained the median of the IGD values out of all the runs.

All the algorithms were given real-valued variables. Simulated binary crossover (SBX) [65] and polynomial mutation (PM) [66] (with the distribution indexes 20) were used to perform the variation. The crossover probability was set to $p_c = 1.0$ and mutation probability to $p_m = 1/d$, where d is the number of variables in the decision space.

In decomposition-based EMO, the population size which correlates with the number of the weight vectors cannot be set arbitrarily. For a set of uniformly-distributed weight vectors in a simplex, we set 100, 105 and 220 for the 2-, 3- and 10-objective problems, respectively. Like many existing studies, the number of function evaluations was set to 25,000, 30,000 and 100,000 for 2-, 3- and 10-objective problems, respectively. Each algorithm was executed 30 independent runs on each problem.

Parameters of the peer algorithms were set as specified/recommended in their original papers. In MOEA/D, the neighbourhood size, the probability of parent solutions selected from the neighbours, and the maximum number of replaced solutions were set to 10% of the population size, 0.9, and 1% of the population size, respectively. In RVEA, the rate of changing the penalty function and the frequency to conduct the reference vector adaptation were set to 2 and 0.1, respectively. In MOEA/D-AWA, the maximal number of adjusting subproblems and the computational resources for the weight vector adaptation were set to $0.05N$ and 20%, respectively. In addition, the size of the external population in MOEA/D-AWA was set to $1.5N$.

Several specific parameters are required in the proposed AdaW. As stated in Section III-F, the time of updating the weight vectors and the time of not allowing the update were every 5% of the total generations and the last 10% generations, respectively. Finally, the maximum capacity of the archive was set to $2N$.

Tables 1 gives the IGD results (mean and standard deviation) of the five algorithms on all the 17 problems. The better mean for each problem was highlighted in boldface. To have statistically sound conclusions, the Wilcoxon's rank sum test [67] at a 0.05 significance level was used to test the significance of the differences between the results obtained by AdaW and the four peer algorithms.

4.1 On Simplex-like Pareto Fronts

On MOPs with a simplex-like Pareto front, decomposition-based algorithms are expected to perform well. Figures 3–5 plot the final solution set of the five algorithms on DTLZ1, DTLZ2 and CDTLZ2, respectively. As can be seen, MOEA/D, RVEA, MOEA/D-AWA and AdaW can all obtain a well-distributed solution set, despite the set of AdaW not being so "regular" as that of the other three algorithms. An interesting observation is that A-NSGA-III (adapting the weight vectors in NSGA-III) appears to struggle in maintaining the uniformity of the solutions, especially for DTLZ1 and CDTLZ2. This indicates that adapting the weight vectors may compromise the performance of decomposition-based approach on simplex-like Pareto fronts, as NSGA-III had been demonstrated to work very well on these three MOPs [25]. In addition, it is worth mentioning that on the convex DTLZ2 there is an interval between the outer and inner solutions in the solution sets of MOEA/D, RVEA and MOEA/D-AWA. In contrast, the proposed AdaW has no such interval, thereby returning a better IGD result as shown in Table 1.

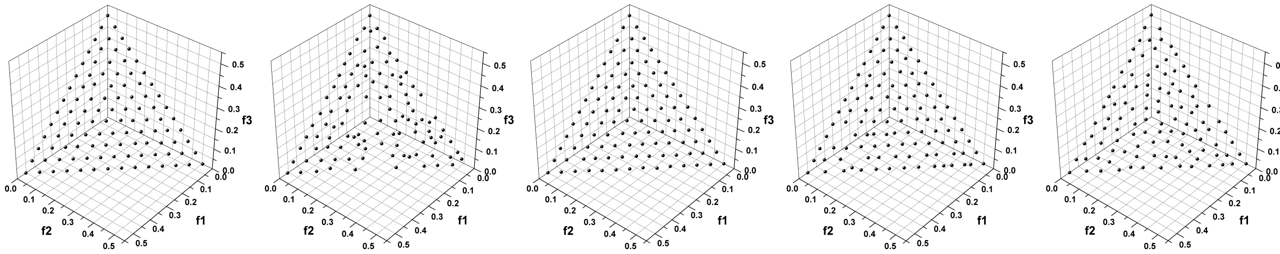
4.2 On Inverted Simplex-like Pareto Fronts

The proposed AdaW has shown a clear advantage over its competitors on this group. Figures 6–7 plot the final solution set of the five algorithms on IDTLZ1 and IDTLZ2, respectively. As shown,

Table 1: IGD results (mean and SD) of the five algorithms. The better mean for each case is highlighted in boldface.

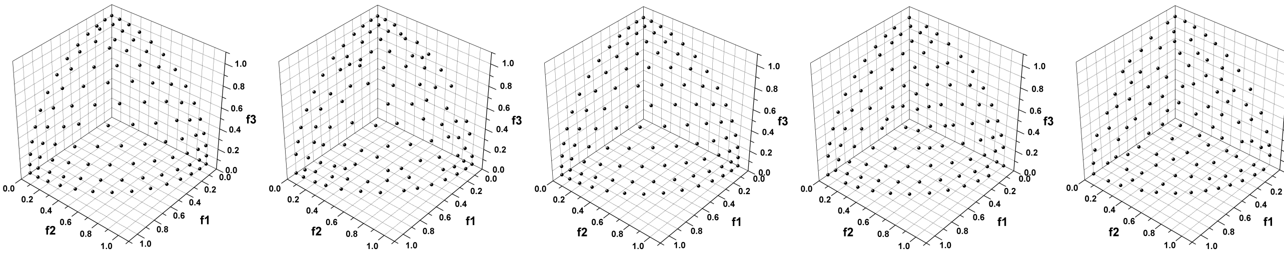
Property	Problem	MOEA/D	A-NSGA-III	RVEA	MOEA/D-AWA	AdaW
Simplex-like	DTLZ1	**1.909E–02(3.1E–04)**†	2.463E–02(8.0E–03)†	1.974E–02(2.2E–03)	1.941E–02(6.1E–04)	1.944E–02(3.1E–04)
	DTLZ2	5.124E–02(4.6E–04)	5.222E–02(1.4E–03)†	**5.020E–02(7.3E–05)**†	5.070E–02(3.8E–04)†	5.126E–02(6.0E–04)
	CDTLZ2	4.388E–02(1.0E–04)†	8.766E–02(2.8E–02)†	4.198E–02(1.4E–03)†	3.879E–02(3.2E–03)†	**2.852E–02(5.9E–04)**
Inverted simplex-like	IDTLZ1	3.175E–02(7.9E–04)†	2.091E–02(1.5E–03)†	6.404E–02(4.6E–02)†	2.698E–02(6.2E–04)†	**1.961E–02(4.8E–04)**
	IDTLZ2	9.010E–02(1.5E–04)†	7.200E–02(6.7E–03)†	7.736E–02(1.7E–03)†	7.166E–02(5.2E–03)†	**5.037E–02(6.2E–04)**
Highly nonlinear	SCH1	4.835E–02(1.7E–03)†	5.411E–02(9.7E–03)†	4.643E–02(4.1E–03)†	2.604E–02(3.6E–03)†	**1.703E–02(1.5E–04)**
	FON	**4.596E–03(1.6E–05)**†	5.333E–03(4.5E–04)†	5.161E–03(1.8E–04)†	4.739E–03(5.2E–05)†	4.632E–03(8.3E–05)
Disconnect	ZDT3	1.107E–02(5.1E–04)†	3.735E–02(4.1E–02)†	9.128E–02(4.2E–02)†	3.125E–02(5.1E–02)†	**4.840E–03(5.6E–04)**
	DTLZ7	1.297E–01(1.1E–03)†	7.079E–02(2.3E–03)†	1.012E–01(4.6E–03)†	1.318E–01(9.0E–02)†	**5.275E–02(6.0E–04)**
Degenerate	DTLZ5	1.811E–02(1.0E–05)†	9.759E–03(1.2E–03)†	6.816E–02(5.3E–03)†	9.584E–03(2.9E–04)†	**3.976E–03(2.4E–04)**
	VNT2	4.651E–02(2.7E–04)†	2.143E–02(3.2E–03)†	3.492E–02(4.6E–03)†	1.961E–02(7.4E–04)†	**1.155E–02(2.3E–04)**
Badly scaled	SDTLZ1	5.584E+00(2.0E+00)†	7.426E–01(4.2E–02)†	1.522E+00(2.0E+00)†	2.988E+00(5.0E–01)†	**6.571E–01(6.0E–02)**
	SDTLZ2	6.071E+00(2.0E+00)†	1.357E+00(4.7E–02)†	1.295E+00(1.7E–02)†	4.176E+00(5.7E–01)†	**1.244E+00(5.2E–02)**
	SCH2	1.049E–01(2.6E–04)†	5.109E–02(4.4E–02)†	4.488E–02(3.6E–04)†	5.538E–02(3.0E–03)†	**2.097E–02(3.1E–04)**
Many objectives	DTLZ2-10	5.172E–01(1.4E–02)	5.314E–01(6.2E–02)†	**4.924E–01(2.6E–05)**†	5.234E–01(3.1E–02)	5.202E–01(1.4E–02)
	IDTLZ1-10	2.721E–01(7.7E–03)†	1.507E–01(6.5E–03)†	2.461E–01(9.0E–03)†	2.421E–01(9.0E–03)†	**1.071E–01(3.3E–03)**
	DTLZ5(2,10)	1.708E–01(1.6E–03)†	4.431E–01(1.0E–01)†	1.520E–01(2.3E–02)†	3.830E–02(1.3E–02)†	**2.150E–03(1.8E–05)**

"†" indicates that the result of the peer algorithm is significantly different from that of AdaW at a 0.05 level by the Wilcoxon's rank sum test.



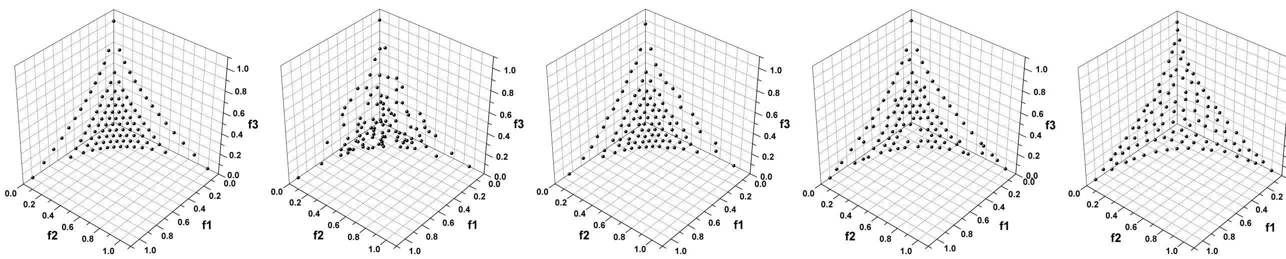
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 3: The final solution set of the five algorithms on DTLZ1.



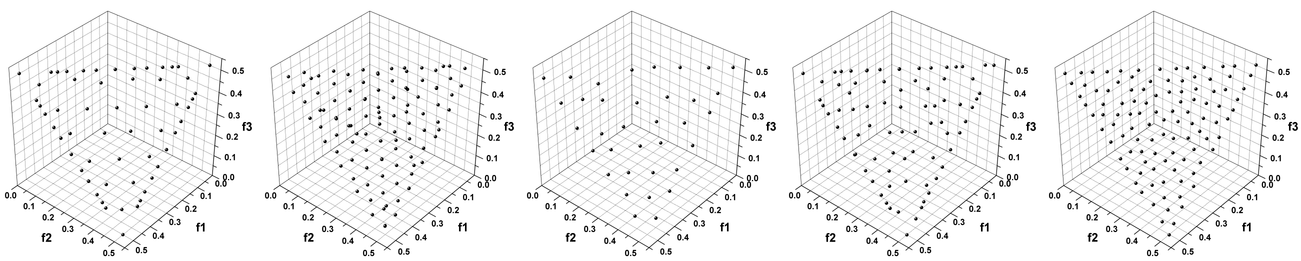
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 4: The final solution set of the five algorithms on DTLZ2.



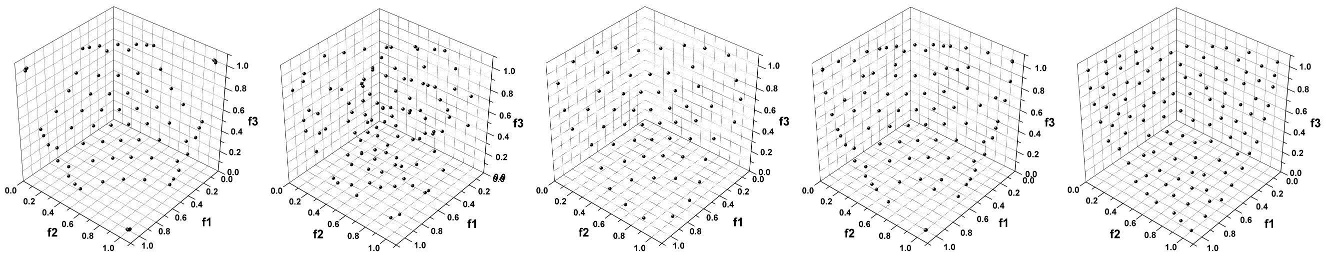
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 5: The final solution set of the five algorithms on the convex DTLZ1.



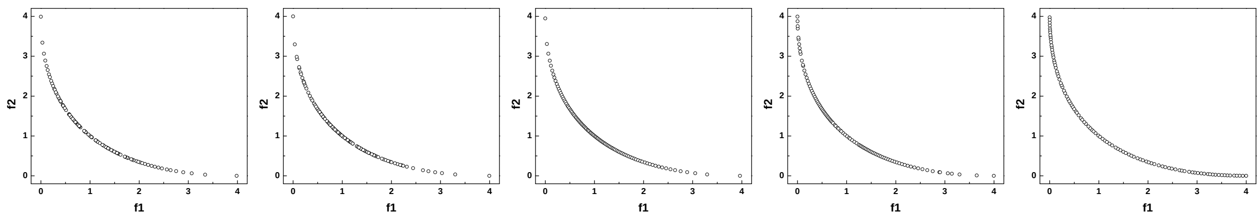
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 6: The final solution set of the five algorithms on the inverted DTLZ1.



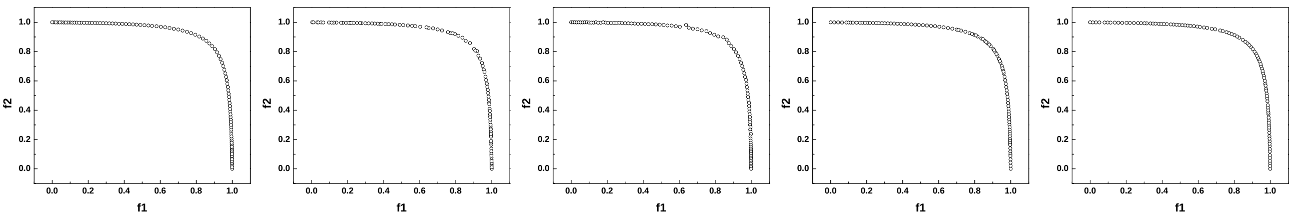
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 7: The final solution set of the five algorithms on the inverted DTLZ2.



(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 8: The final solution set of the five algorithms on SCH1.



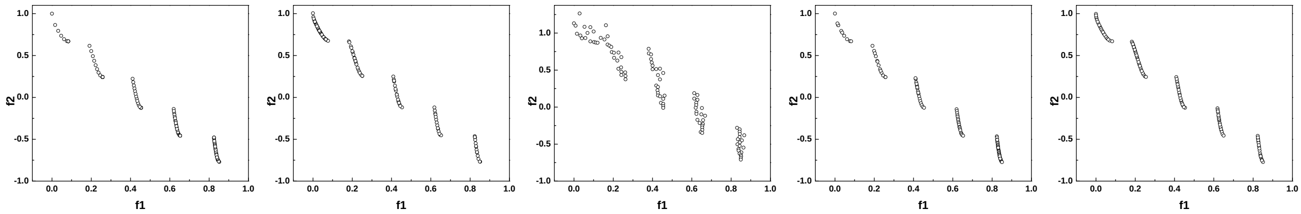
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 9: The final solution set of the five algorithms on FON.

many solutions of MOEA/D and MOEA/D-AWA concentrate on the boundary of the Pareto front. The solutions of A-NSGA-III have a good coverage but are not distributed very uniformly, while the solutions of RVEA are distributed uniformly but their number is apparently less than the population size. For AdaW, an inverted simple-like Pareto front has no effect on the algorithm's performance, and the obtained solution set has a good coverage and uniformity over the whole front.

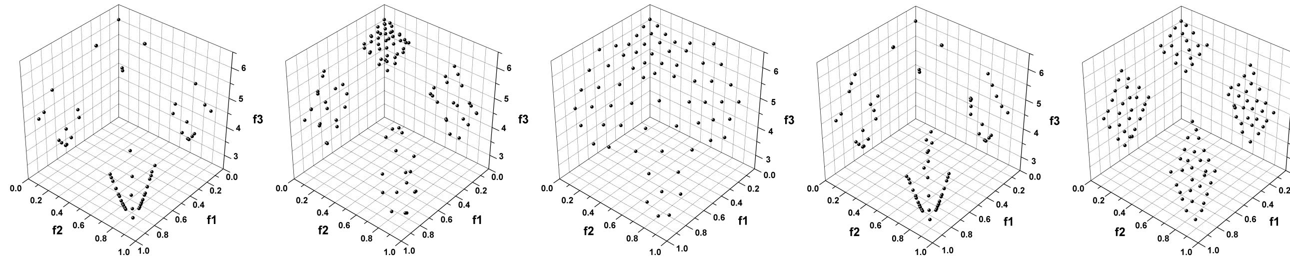
4.3 On Highly Nonlinear Pareto Fronts

The peer algorithms perform differently on the two instances of this group. On the problem with a concave Pareto front (i.e., FON), all the algorithms work well (Figure 9), despite A-NSGA-II and RVEA performing slightly worse than the other three. In contrast, on the problem with a convex Pareto front (i.e., SCH1), only the proposed AdaW can obtain a well-distributed solution set, and the others fail to extend their solutions to the boundary of the Pareto front (Figure 8). This indicates that the convex Pareto front still poses a challenge to decomposition-based approach even if some weight vector adaptations are introduced.



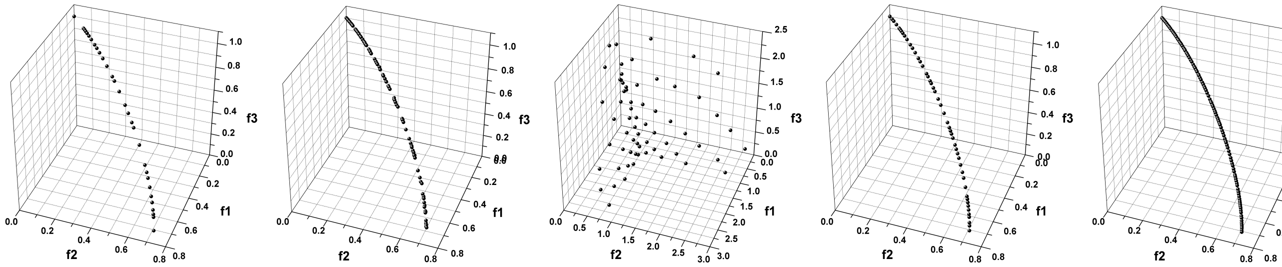
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 10: The final solution set of the five algorithms on ZDT3.



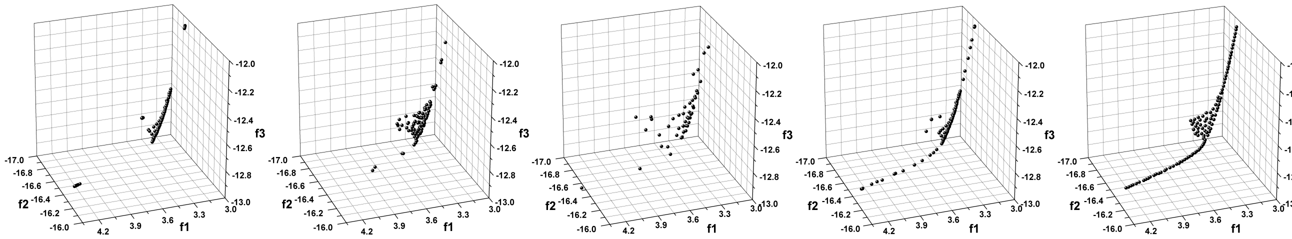
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 11: The final solution set of the five algorithms on DTLZ7.



(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 12: The final solution set of the five algorithms on DTLZ5.



(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 13: The final solution set of the five algorithms on VNT2.

4.4 On Disconnected Pareto Fronts

Figures 10 and 11 plot the final solution set of the five algorithms on ZDT3 and DTLZ7, respectively. On ZDT3, only AdaW and A-NSGA-III can maintain a good distribution of the solution set. MOEA/D and MOEA/D-AWA show a similar pattern, with their solutions distributed sparsely on the upper-left part of the Pareto front. The set obtained by RVEA has many dominated solutions. On DTLZ7, only the proposed algorithm works well. The peer algorithms either fail to lead their solutions to cover the Pareto front (MOEA/D and MOEA/D-AWA), struggle to maintain the uniformity (A-NSGA-III), or produce some dominated solutions (RVEA).

4.5 On Degenerate Pareto Fronts

Problems with a degenerate Pareto front poses a big challenge to decomposition-based approaches since the ideal weight vector set is located in a lower-dimensional manifold than its initial setting [20]. On this group of problems, the proposed algorithm has shown a significant advantage over its competitors (see Figures 12 and 13). It is worth noting that VNT2 has a mixed Pareto front, with both ends degenerating into two curves and the middle part being a triangle-like plane. As can be seen from Figure 13, the solution set of AdaW has a good distribution over the whole Pareto front.

4.6 On Badly-Scaled Pareto Fronts

Figures 14–16 plot the final solution set of the five algorithms on SDTLZ1, SDTLZ2 and SCH2, respectively. For the first two problems, AdaW, A-NSGA-III and RVEA work fairly well, but the solutions obtained by RVEA are not so uniform as those obtained by the other two algorithms on SDTLZ1. For SCH2 which also has a disconnected Pareto front, AdaW significantly outperforms its competitors, with the solution set being uniformly distributed over the two parts of the Pareto front.

4.7 On Many-Objective Problems

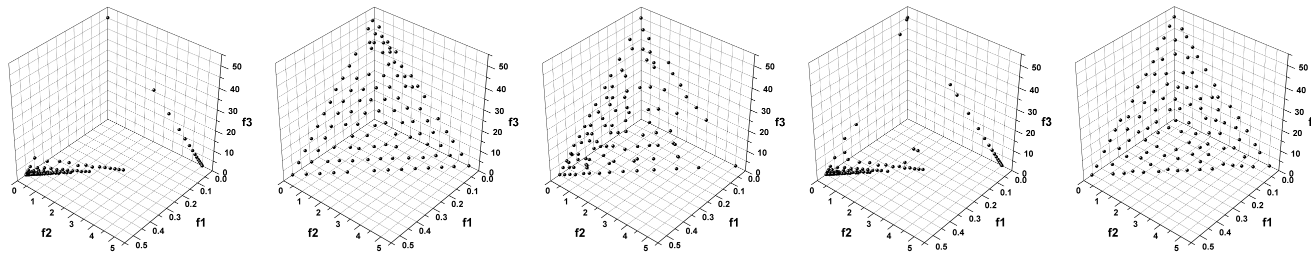
This section evaluates the performance of the proposed AdaW on many-objective problems by considering three instances, the 10-objective DTLZ2, 10-objective IDTLZ1, and DTLZ5(2,10) where the number of objectives is 10 and the true Pareto front's dimensionality is 2.

For the 10-objective DTLZ2 which has a simplex-like Pareto front, all the five algorithms appear to work well (Figure 17) despite that there exists one solution of AdaW not converging into the Pareto front. We may not be able to conclude the distribution difference of the algorithms by the parallel coordinates plots [68], but all the algorithms seem to perform similarly according to the IGD results in Table 1.

For the many-objective problems whose Pareto front is far from the standard simplex, a clear advantage of AdaW over its competitors is shown (Figures 18 and 19). The peer algorithms either fail to cover the whole Pareto front (i.e., MOEA/D, A-NSGA-III and MOEA/D-AWA on the 10-objective IDTLZ1 and MOEA/D and MOEA/D-AWA on DTLZ5(2,10)), or struggle to converge into the front (i.e., RVEA on the 10-objective IDTLZ1 and A-NSGA-III and RVEA on DTLZ5(2,10)). In contrast, the proposed AdaW has shown its ability in dealing with irregular Pareto fronts in the high-dimensional space, by which a spread of solutions over the whole Pareto front is obtained.

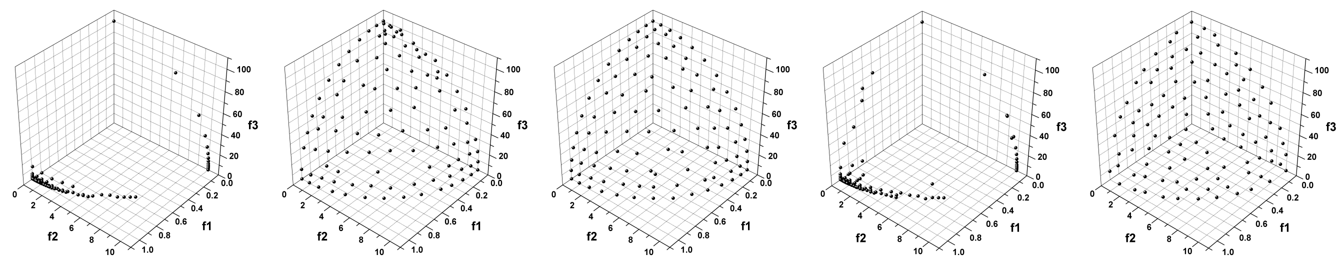
5 Conclusions

Adaptation of the weight vectors during the optimisation process provides a viable approach to enhance existing decomposition-based EMO. This paper proposed an adaptation method to periodically update the weight vectors by contrasting the current evolutionary population with a well-maintained



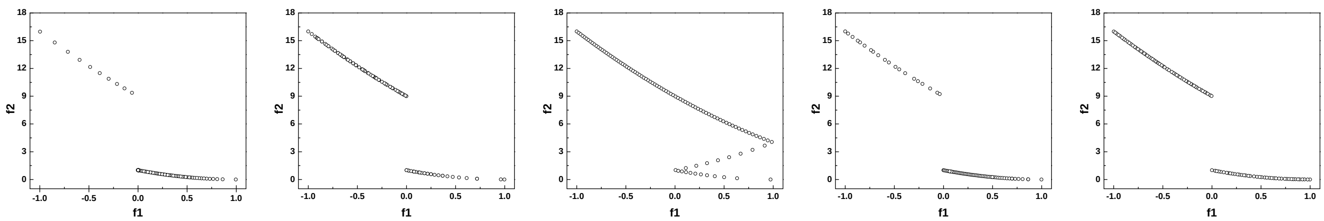
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 14: The final solution set of the five algorithms on the scaled DTLZ1.



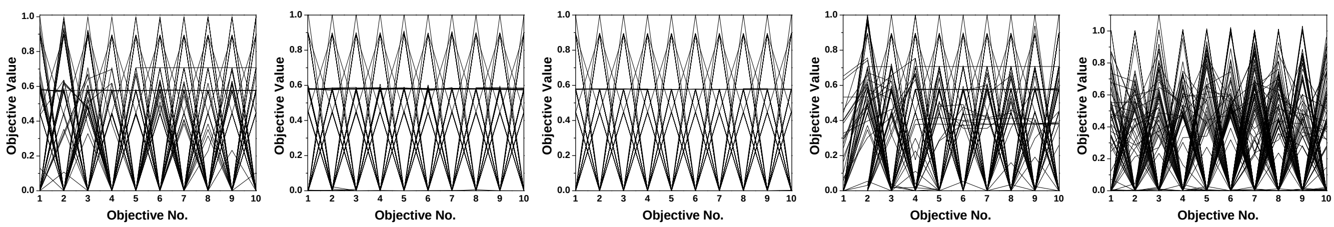
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 15: The final solution set of the five algorithms on the scaled DTLZ2.



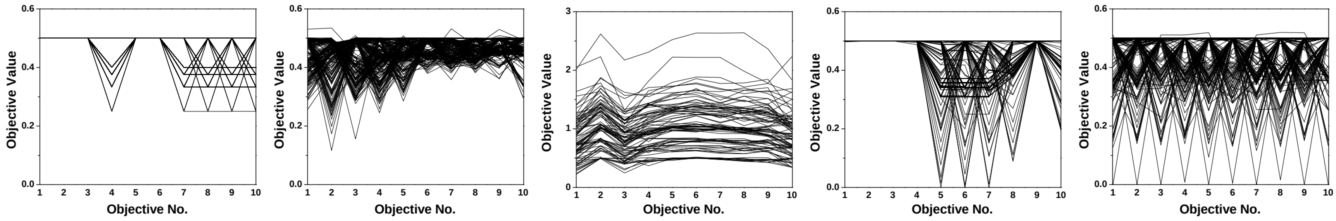
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 16: The final solution set of the five algorithms on SCH2.



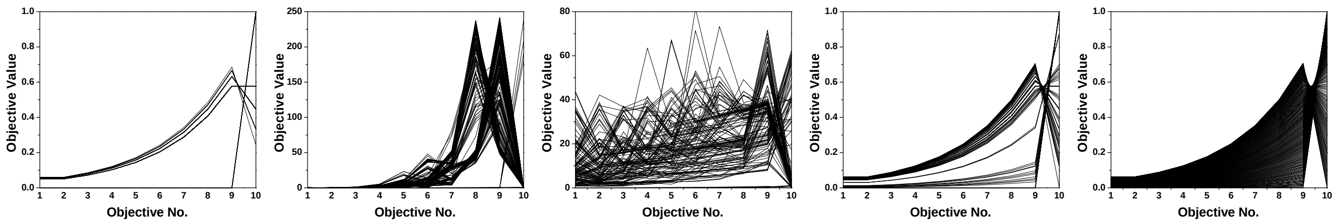
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 17: The final solution set of the five algorithms on the 10-objective DTLZ2.



(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 18: The final solution set of the five algorithms on the 10-objective inverted DTLZ1.



(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

Figure 19: The final solution set of the five algorithms on the DTLZ5(2,10).

archive set. From experimental studies on seven categories of problems with various properties, the proposed algorithm has shown its high performance over a wide variety of different Pareto fronts.

However, it is worth noting that the proposed algorithm needs more computational resources than the basic MOEA/D. The time complexity of AdaW is bounded by $O(mN^2)$ or $O(TN^2)$ whichever is greater (where m is the number of objectives and T is the neighbourhood size), in contrast to $O(mTN)$ of MOEA/D. In addition, AdaW also incorporates several parameters, such as the maximum capacity of the archive and the time of updating the weight vectors. Although these parameters were fixed on all test problems in our study, customised settings for specific problems may lead to better performance. For example, a longer duration allowing the weight vectors evolving along the constant weight vectors is expected to achieve better convergence on problems with many objectives.

References

[1] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[2] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[3] S. Z. Martínez and C. A. C. Coello, "A direct local search mechanism for decomposition-based multi-objective evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1–8.

[4] B. Derbel, A. Liefooghe, Q. Zhang, H. Aguirre, and K. Tanaka, "Multi-objective local search based on decomposition," in *International Conference on Parallel Problem Solving from Nature*, 2016, pp. 431–441.

[5] H. Ishibuchi and T. Murata, "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)*, vol. 28, no. 3, pp. 392–403, 1998.

[6] Y. Mei, K. Tang, and X. Yao, "A memetic algorithm for periodic capacitated arc routing problem," *IEEE Transactions on Systems Man and Cybernetics-Part B-Cybernetics*, vol. 41, no. 6, pp. 1654–1667, 2011.

[7] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 682–691, 2012.

[8] C. P. Almeida, R. A. Gonçalves, E. F. Goldbarg, M. C. Goldbarg, and M. R. Delgado, "An experimental analysis of evolutionary heuristics for the biobjective traveling purchaser problem," *Annals of Operations Research*, vol. 199, no. 1, pp. 305–341, 2012.

[9] E. J. Hughes, "Evolutionary many-objective optimisation: Many once or one many?" in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 1, 2005, pp. 222–227.

[10] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition based evolutionary algorithm for many objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 445–460, 2015.

[11] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[12] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[13] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[14] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2014.

[15] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching based selection in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2014.

[16] A. Zhou and Q. Zhang, "Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 52–64, 2016.

[17] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, 2014.

[18] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: Bi-criterion evolution in multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 645–665, 2016.

[19] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 169–190, 2017.

[20] M. Li, C. Grosan, S. Yang, X. Liu, and X. Yao, "Multi-line distance minimization: A visualized many-objective test problem suite," *IEEE Transactions on Evolutionary Computation*, 2017, DOI: 10.1109/TEVC.2017.2655451.

[21] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, 2016.

[22] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Towards understanding the cost of adaptation in decomposition-based optimization algorithms," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 615–620.

[23] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–35, 2015.

[24] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[25] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[26] S. Jiang and S. Yang, "A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, 2017.

[27] Y. Tan, Y. Jiao, H. Li, and X. Wang, "MOEA/D + uniform design: A new version of MOEA/D for optimization problems with many objectives," *Computers & Operations Research*, vol. 40, no. 6, pp. 1648–1660, 2013.

[28] X. Ma, Y. Qi, L. Li, F. Liu, L. Jiao, and J. Wu, "MOEA/D with uniform decomposition measurement for many-objective problems," *Soft Computing*, vol. 18, no. 12, pp. 2541–2564, 2014.

[29] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[30] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Generalized decomposition," in *Evolutionary Multi-Criterion Optimization (EMO)*, 2013, pp. 428–442.

[31] R. Wang, Q. Zhang, and T. Zhang, "Decomposition-based algorithms using Pareto adaptive scalarizing methods," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 821–837, 2016.

[32] Y. Jin, T. Okabe, and B. Sendhoff, "Adapting weighted aggregation for multiobjective evolution strategies," in *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 96–110.

[33] H. Li, M. Ding, J. Deng, and Q. Zhang, "On the use of random weights in MOEA/D," in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2015, pp. 978–985.

[34] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[35] H. Li and D. Landa-Silva, "An adaptive evolutionary multi-objective approach based on simulated annealing," *Evolutionary computation*, vol. 19, no. 4, pp. 561–595, 2011.

[36] S. Jiang, Z. Cai, J. Zhang, and Y. S. Ong, "Multiobjective optimization by decomposition with Pareto-adaptive weight vectors," in *2011 International Conference on Natural Computation (ICNC)*, vol. 3, 2011, pp. 1260–1264.

[37] F. Gu, H. Liu, and K. C. Tan, "A multiobjective evolutionary algorithm using dynamic weight design method," *International Journal of Innovative Computing Information and Control*, vol. 8, no. 5, pp. 3677–3688, 2012.

[38] F. Gu and Y.-M. Cheung, "Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, 2017, DOI: 10.1109/TEVC.2017.2695579.

[39] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, Aug 2014.

[40] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances." in *IEEE Congress on Evolutionary Computation (CEC)*, 2009, pp. 203–208.

[41] H.-L. Liu, L. Chen, Q. Zhang, and K. Deb, "An evolutionary many-objective optimisation algorithm with adaptive region decomposition," in *IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 4763–4769.

[42] ——, "Adaptively allocating search effort in challenging many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, 2017, in press.

[43] R. Cheng, Y. Jin, and K. Narukawa, "Adaptive reference vector generation for inverse model based evolutionary multiobjective optimization with degenerate and disconnected Pareto fronts," in *International Conference on Evolutionary Multi-Criterion Optimization*, 2015, pp. 127–140.

[44] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[45] C. Zhang, K. C. Tan, L. H. Lee, and L. Gao, "Adjust weight vectors in moea/d for bi-objective optimization problems with discontinuous pareto fronts," *Soft Computing*, pp. 1–16, 2017.

[46] Z. Wang, Q. Zhang, H. Li, H. Ishibuchi, and L. Jiao, "On the use of two reference points in decomposition based multiobjective evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 34, pp. 89–102, 2017.

[47] X. Cai, Z. Mei, and Z. Fan, "A decomposition-based many-objective evolutionary algorithm with two types of adjustments for direction vectors," *IEEE Transactions on Cybernetics*, 2017.

[48] M. Asafuddoula, H. K. Singh, and T. Ray, "An enhanced decomposition-based evolutionary algorithm with adaptive reference vectors," *IEEE Transactions on Cybernetics*, 2017, in press.

[49] R. Wang, R. C. Purshouse, and P. J. Fleming, “On finding well-spread Pareto optimal solutions by preference-inspired co-evolutionary algorithm,” in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)*, 2013, pp. 695–702.

[50] ——, “Preference-inspired co-evolutionary algorithms using weight vectors,” *European Journal of Operational Research*, vol. 243, no. 2, pp. 423–441, 2015.

[51] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, “An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility,” *IEEE Transactions on Evolutionary Computation*, 2017, in press.

[52] Y. Xiang, Z. Yuren, M. Li, and Z. Chen, “A vector angle based evolutionary algorithm for unconstrained many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 131–152, 2017.

[53] Y. Xiang, J. Peng, Y. Zhou, M. Li, and Z. Chen, “An angle based constrained many-objective evolutionary algorithm,” *Applied Intelligence*, 2017, in press.

[54] M. Li, S. Yang, K. Li, and X. Liu, “Evolutionary algorithms with segment-based search for multiobjective optimization problems,” *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1295–1313, 2014.

[55] K. Ikeda, H. Kita, and S. Kobayashi, “Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal?” in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, vol. 2, 2001, pp. 957–962.

[56] L. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao, “Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm,” *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 475–480, 2016.

[57] R. Wang, J. Xiong, H. Ishibuchi, G. Wu, and T. Zhang, “On the effect of reference point in moea/d for multi-objective optimization,” *Applied Soft Computing*, vol. 58, pp. 25–34, 2017.

[58] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “Platemo: A matlab platform for evolutionary multi-objective optimization,” *arXiv preprint arXiv:1701.00879*, 2017.

[59] D. A. Van Veldhuizen, “Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations,” Ph.D. dissertation, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.

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

[61] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable test problems for evolutionary multi-objective optimization,” in *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. Berlin, Germany: Springer, 2005, pp. 105–145.

[62] K. Deb and D. K. Saxena, “On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems,” *Kangal report*, vol. 2005011, 2005.

[63] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, and X. Yao, “A benchmark test suite for evolutionary many-objective optimization,” *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 67–81, 2017.

[64] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *Mexican International Conference on Artificial Intelligence (MICAI)*, 2004, pp. 688–697.

[65] R. B. Agrawal, K. Deb, and R. Agrawal, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[66] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms.* New York: John Wiley, 2001.

[67] E. Zitzler, J. Knowles, and L. Thiele, "Quality assessment of Pareto set approximations," in *Multiobjective Optimization*, J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Eds. Springer Berlin / Heidelberg, 2008, vol. 5252, pp. 373–404.

[68] M. Li, L. Zhen, and X. Yao, "How to read many-objective solution sets in parallel coordinates," *IEEE Computational Intelligence Magazine*, 2017, in press.