Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

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

Miqing Li limitsing@gmail.com
CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U. K.

Xin Yao x.yao@cs.bham.ac.uk
Department of Computer Science, Southern University of Science and Technology, Shenzhen, China; CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U. K.

Abstract

The quality of solution sets generated by decomposition-based evolutionary multi-objective 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 weights during the evolutionary process (called AdaW). AdaW progressively seeks a suitable distribution of weights for the given problem by elaborating several key parts in weight adaptation — weight generation, weight addition, weight deletion, 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 degenerate, 6) the scaled, and 7) the high-dimensional.

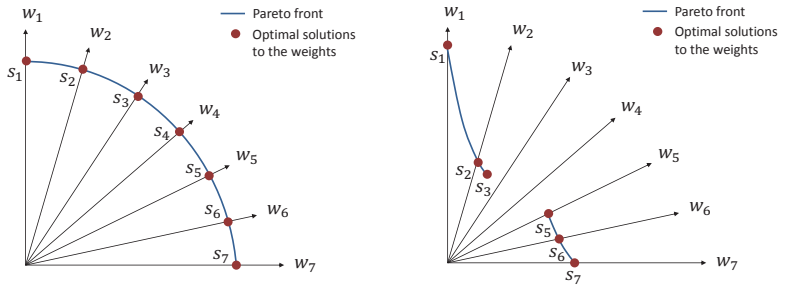
Keywords

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

1 Introduction

Decomposition-based evolutionary multi-objective optimisation (EMO), crystallised in Zhang and Li (2007), is a general-purpose algorithm framework (termed as MOEA/D). It decomposes a multi-objective optimisation problem (MOP) into a number of single-objective (or multi-objective (Liu et al., 2014)) 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

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



(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.

conventional Pareto-based EMO, decomposition-based EMO has clear strengths, e.g., providing high selection pressure toward the Pareto front (Hughes, 2005; Li et al., 2014a), being easy to work with local search operators (Ishibuchi et al., 2003; Martínez and Coello, 2012; Derbel et al., 2016), owning high search ability for combinatorial MOPs (Ishibuchi and Murata, 1998; Mei et al., 2011; Shim et al., 2012; Almeida et al., 2012; Cai et al., 2015), and being capable of dealing with MOPs with many objectives (Asafuddoula et al., 2015b; Li et al., 2015c; Yuan et al., 2016) and MOPs with a complicated Pareto set (Li and Zhang, 2009; Liu et al., 2014; Zhou and Zhang, 2016).

A key feature in MOEA/D is that the diversity of the evolutionary population is controlled explicitly by a set of weights (or a set of reference directions/points determined by this weight set). Each weight corresponds to one subproblem, ideally associated with one solution in the population; thus, diverse weights may lead to different Pareto optimal solutions. In general, a well-distributed solution set can be obtained if the set of weights and the Pareto front of a given problem share the same/similar distribution shape. In many existing studies, the weights 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 weights 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 weights 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 scaled), decomposition-based algorithms appear to struggle (Qi et al., 2014; Li et al., 2016; Ishibuchi et al., 2017b; Li et al., 2018). In such MOPs, some weights may have no intersection with the Pareto front. This could lead to several weights corresponding to one Pareto optimal solution. In addition, there may exist a big difference of distance between adjacent Pareto optimal solutions (obtained by adjacent weights) 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 distri-

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

bution. Overall, no intersection between some weights and the Pareto front may cause the number of obtained Pareto optimal solutions to be smaller than that of weights, while the distance difference with 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 weights on an "irregular" Pareto front. As can be seen, weights w_3 and w_4 have no intersection with the Pareto front, and weights 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 greater than that between other adjacent solutions.

The above example illustrates the difficulties of predefining weights in MOEA/D. It could be very challenging (or even impossible) to find a set of optimal weights 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 weights during the evolutionary process. Several interesting attempts have been made along this line (Trivedi et al., 2017). 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 fully resolved. On one hand, varying the weights 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 weights materially changes the subproblems over the course of the optimisation, which could significantly deteriorate the convergence of the algorithm (Giagkiozis et al., 2013b). Overall, as pointed out in Li et al. (2015a); Ishibuchi et al. (2017b), how to set the weights 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 weights during the evolutionary process. AdaW updates the weights periodically based on the information produced by the evolving population itself, and then in turn guides the population by these weights which are of a suitable distribution for the given problem. AdaW focuses on several key parts in the weight adaptation process, namely, weight generation, weight addition, weight deletion, and weight update frequency. The main contributions of this work can be summarised as follows.

- An approach to find out potential undeveloped, but promising weights is presented, with the aid of a well-maintained archive set.
- An approach to delete unpromising weights is presented, taking into account both the number of solutions associated with the weight and the crowding degree in the space.
- A design to make several parts in the weight adaptation process cohere as a whole is developed, enabling the algorithm to strike a balance between convergence and diversity on various problems.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 is devoted to the proposed adaptation method, including the basic idea and the

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

five key issues of this adaptation. Experimental results are presented in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

A basic assumption in MOEA/D is that the diversity of the weights will result in the diversity of the Pareto optimal solutions. This motivates several studies on how to generate a set of uniformly distributed weights (see Trivedi et al. (2017)), such as the simplex-lattice design (Das and Dennis, 1998), two-layer simplex lattice design (Deb and Jain, 2014), multi-layer simplex lattice design (Jiang and Yang, 2017), uniform design (Tan et al., 2013), and a combination of the simplex-lattice design and uniform design (Ma et al., 2014). A weakness of such systematic weight generators is that the number of generated weights is not flexible. This contrasts with the uniform random sampling method (Murata et al., 2001; Jaszkiewicz, 2002) which can generate an arbitrary number of weights 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 weights for a specific scalarising function can be readily identified (Giagkiozis et al., 2013a; Wang et al., 2016b).

The variety of weight generators gives us ample options in initialising the weights, each of which provides an explicit way of specifying a set of particular search directions in decomposition-based optimisation. However, the precondition 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 and scaled) may make these weight generators struggle, in which multiple weights may correspond to one single point. This leads to a waste of computational resources, and more importantly renders the algorithm's performance inferior.

An intuitive solution to this problem is to adaptively update the weights during the optimisation process. Several interesting attempts have been made along this line. Table 1, on the basis of some previous studies (Asafuddoula et al., 2017), summarises existing works of considering weight adaptation in decomposition-based EMO. These works have represented significant progress made in weight adaptation, with various features incorporated (see Table 1) and clear advantage exhibited (over the weight presetting method) on many irregular Pareto fronts. In addition, it is worth mentioning that some researchers also introduced weights into non-decomposition-based EMO, and conducted weight adaptation for Pareto-based search (Wang et al., 2013, 2015) and indicator-based search (Tian et al., 2017a). And some other researchers adaptively adjusted the search directions according to the distribution of the evolutionary population, which in a sense can also be seen as a weight adaptation (Xiang et al., 2017b,a).

However, it is still far from weight adaptation being a mature method in the sense that it is able to deal with a wide variety of MOPs as effectively as the weight presetting method dealing with simplex-like Pareto fronts. Some challenges/limitations facing are outlined as follows.

- Difficulties in obtaining/generating new weights; i.e., where to find promising weights. Many adaptation methods introduce new weights from a set of systematically-generated weights, such as EMOSA (Li and Landa-Silva, 2011), A-

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Table 1: Summary of existing works that consider weight adaptation in MOEA/D.

Method	Features/characteristics, strengths	Weaknesses/limitations
Random weights (Ishibuchi and Murata, 1998; Jin et al., 2001; Jaszkiewicz, 2002)	More computational cost allocated around the nondominated solutions; helpful for irregular Pareto fronts (Li et al., 2015b).	Unable to maintain solutions' uniformity; poor convergence.
EMOSA (Li and Landa-Silva, 2011)	Adapting the weights to diversify the population towards the unexplored parts of the Pareto front.	Systematic weight generation for adaptation, thus likely to struggle on irregular Pareto fronts.
$pa\lambda$-MOEA/D (Jiang et al., 2011)	Sampling the regression curve of the weights on the basis of an external archive.	Assuming symmetry and continuity of the Pareto front (Asafuddoula et al., 2017).
DMOEA/D (Gu et al., 2012)	Equidistant interpolation to update the weights; working well on bi-objective problems.	Hard to maintain uniformity of solutions on problems with more than two objectives.
A-NSGA-III (Jain and Deb, 2014)	An $(m-1)$-simplex of reference points centred around a crowded reference point being added.	Systematic weight generation for adaptation; struggling on disconnected and degenerate Pareto fronts.
MOEA/D-AWA (Qi et al., 2014)	Generating new weights by particular solutions in the archive; able to tackle problems with "sharp peak" and "low tail".	Updating weights in the ending stage of evolution; not performing very well in uniformity maintenance.
RVEA (Cheng et al., 2015, 2016)	Two weight adaptations being conducted to deal with scaled Pareto fronts and irregular fronts respectively.	Failing to maintain diversity of solutions on highly nonlinear Pareto fronts.
MOEA/D-AM2M (Liu et al., 2016, 2017)	An adaptive weight update for MOEA/D-M2M (Liu et al., 2014) for degenerate Pareto fronts according to the angle among solutions (as a similarity measure).	Less global competition in finding multiple regions on the Pareto front (Liu et al., 2017).
SOM for weight adaptation (Gu and Cheung, 2018)	Updating weights via training a self-organising map (SOM); effective for degenerate Pareto fronts.	Still difficult to maintain a good uniformity of solutions through limited training vectors.
MOEA/D-ABD (Zhang et al., 2017)	Linear interpolation to update weights for discontinuous Pareto fronts.	Restricted to bi-objective problems.
MOEA/D-MR (Wang et al., 2017b)	Updating weights via considering both ideal and nadir points.	Potentially struggling on problems with irregular Pareto fronts, e.g., degenerate and discontinuous.
MaOEA/D-2ADV (Cai et al., 2017)	Two types of weight adjustments: one for the number of weights and the other for the position of weights; achieving a good balance between convergence and diversity.	Designed for many-objective problems; only being evaluated on the DTLZ test problems most of which have regular Pareto fronts.
g-DBEA (Asafuddoula et al., 2017)	An adaptive weight update for DBEA (Asafuddoula et al., 2015a) via a "learning period"; storing the removed weights for the future use; capable of obtaining diversified solutions over the front.	Not performing very well in maintaining uniformity of solutions.

NSGA-III (Jain and Deb, 2014) and RVEA (Cheng et al., 2016). However, such weights may still have no intersection with the problem's Pareto front, thus failing to guarantee the uniformity of the final solution set. Determining new weights by promising solutions produced previously during the evolutionary search may alleviate this issue. Yet this requires an additional archive to store these solutions, and how to maintain the archive is of high importance as it essentially determines the search directions of decomposition-based evolution.

- Difficulties of deleting/adding weights. It is not easy to find current unpromising weights. Even when each weight is solely associated with one individual, the population may still not have good uniformity, e.g., in a highly convex Pareto front. In addition, where to add weights is also a trick question, as we do not know if the sparsely-distributed weights represent undeveloped regions that need to be explored or unpromising regions that have no intersection with the Pareto front.

- Difficulties of setting weight update frequency. Varying the weights essentially changes the subproblems. If the update is not frequent enough, there are lots of

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

computational resources wasted on unpromising search directions (Asafuddoula et al., 2017). If the update is too frequent, individuals are likely to *wander* around the search space, which significantly affects the convergence of the algorithm (Giagkiozis et al., 2013b; Qi et al., 2014). Especially when the algorithm approaches the end of the search process, a change of weights may lead to the algorithm ending up returning a well-diversified but poorly-converged population.

- Challenges to adapt weights for different Pareto fronts. Many adaptation methods are designed or suitable for certain types of Pareto fronts; for example, $pa\lambda$-MOEA/D (Jiang et al., 2011) for connected Pareto fronts, DMOEA/D (Gu et al., 2012) and MOEA/D-ABD (Zhang et al., 2017) for bi-objective MOPs, and MOEA/D-AM2M (Liu et al., 2017) and SOM-based weight adaptation (Gu and Cheung, 2018) for degenerate Pareto fronts. In addition, some methods designed for many-objective problems may not perform very well on problems with two or three objectives, especially in maintaining uniformity of solutions (Cai et al., 2017; Asafuddoula et al., 2017).

The above discussions motivate our work. In this paper, we propose a weight adaptation method via elaborating several key parts in weight adaptation — weight generation, weight deletion, weight addition and weight update frequency. Our goal is to present a decomposition-based algorithm which is able to handle various Pareto fronts, the regular and irregular.

3 The Proposed Algorithm

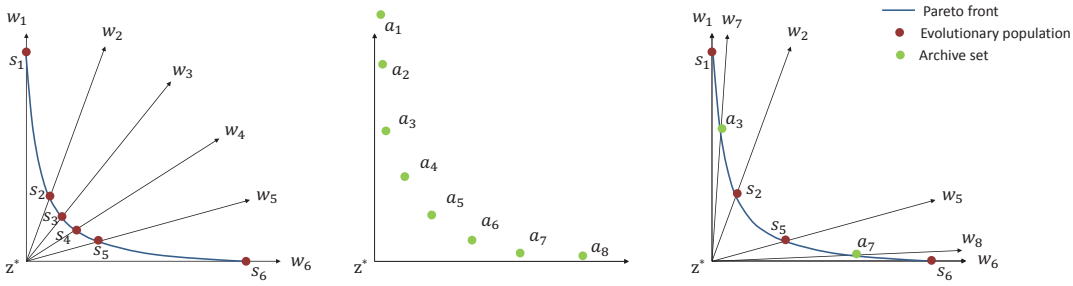
3.1 Basic Idea

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 (Li et al., 2014b; Liu et al., 2016, 2017). 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 when it is well maintained. Despite that such a set probably evolves slowly in comparison with the evolutionary population which is driven by the scalarising function in decomposition-based evolution, the set may be able to provide new search directions that are unexplored by the scalarising function-driven population.

Figure 2 gives an illustration of updating the search directions (weights) 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 weights correspond to a poorly distributed population along the Pareto front. After 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 contrast, the weights that are associated with crowded solutions (s_3 and s_4) in the population are deleted. Then, a new population is formed with unevenly-distributed weights but well-distributed solutions.

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

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



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

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

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

In next several 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 only 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 Li et al. (2016) due to its efficiency and effectiveness. This method is able to maintain a set of representative individuals for various problems, i.e., being independent of the problem properties (e.g., the number of objectives and the shape of the Pareto front) (Li et al., 2016). The method iteratively deletes the solution having the biggest crowding degree in the set. The crowding degree of a solution is estimated by considering both the number and location of its neighbours 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

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

kth nearest solution in the set. Note that similar to Li et al. (2016) 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 Li et al. (2016). First, parameter k for the kth nearest neighbour was set to 3 in Li et al. (2016), while here k is set to the number of objectives of the problem. There are two reasons for this change. The first is that as shown in Li et al. (2016), k is not very sensitive to the performance of the maintenance operation; it works fairly well within a big range like $[2, 10]$. The second reason is that a larger k may be more suitable for many-objective optimisation as it can lead to more emphases put on boundary solutions, which are important points to tell problem characteristics. The second difference is that the median, instead of the average in Li et al. (2016), of the distances from all the solutions to their kth nearest neighbour 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 (Ikeda et al., 2001).

3.3 Weight Addition

In AdaW, we aim to add weights (into the evolutionary population) whose search directions/areas are *undeveloped* and *promising*. Both criteria are 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 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) 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 is undeveloped by the evolutionary population, we then consider whether they are promising or not. First, we obtain their corresponding weights (which will be detailed in next section). Then for each of these weights, we find its neighbouring weights[1] in the evolutionary population, and further determine the solutions associated with the neighbouring weights. Finally we compare these solutions with the candidate solution on the basis of the candidate solution's weight. Formally, let q be a candidate solution in the archive and w_q be its corresponding weight. Let w_p be one of the neighbouring weights 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

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

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 $g()$ denotes the considered scalarising function, $f_i()$ denotes the value of the ith

[1]The definition of neighbouring weights is based on that in MOEA/D (i.e., the Euclidean distance between weights).

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

objective, and m is the number of objectives. If q outperforms all of the neighbouring solutions on the basis of its weight, then q will enter the evolutionary population, along with its search direction (weight). After that, the neighbouring information of q's weight in the evolutionary population (i.e., the solutions that the neighbouring weights corresponding to) is updated by q. Note that there is no restriction for the number of neighbouring solutions that can be replaced by q.

3.4 Weight Generation

Given a reference point, the optimal weight to a solution (e.g., w_7 to a_3 in Figure 2(c)) with respect to the Tchebycheff scalarising function can be easily generated. This is already a frequently used approach in the weight adaptation (Gu et al., 2012; Qi et al., 2014).

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 to a solution q in the Tchebycheff scalarising function. 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 weights in the evolutionary population to keep the number of the weights unchanged (i.e., back to the predefined population size N). In view of that ideally each weight is associated with one distinct solution in decomposition-based EMO, we deal with weights that share one solution (e.g., weights 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 weights in the population. In these weights, we delete the one whose scalarising function value is the worst. Formally, let p be the solution shared by the most weights $w_1, w_2, ..., w_n$ out of the population. Then the weight to be deleted is

$$\arg\max_{1 \le i \le n} \ g(p, w_i) \tag{7}$$

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

The above deletion operation is repeated until the number of the weights 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, the number of the weights still exceeds N. In this situation, we use the same diversity maintenance method of Section 3.2 to iteratively delete the most crowded solution (along with its weight) in the population until the number of the weights reduces to N.

[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 (Wang et al., 2016a; Qi et al., 2014); here we set it to 10^{-4} smaller than the best value throughout the method, following the suggestions in Wang et al. (2017a). This setting is also adopted in the calculation of the scalarising function.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

3.6 Weight Update Frequency

The timing and frequency of updating the weights of the evolutionary population play an important role in weight adaptation methods. Since varying the weights essentially changes the subproblems to be optimised, a frequent change can significantly affect the convergence of the algorithm (Giagkiozis et al., 2013b). In AdaW, the weight update operation is conducted every 5% of the total generations/evaluations. In addition, when the algorithm approaches the end of the optimisation process, a change of the weights may lead to the solutions evolving insufficiently along those specified search directions (weights). Therefore, AdaW does not change the weights 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 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 Li and Zhang (2009), and the Tchebycheff scalarising function was used despite the fact that AdaW can be implemented by other scalarising functions with respect to its weight addition and deletion. That is, the steps of the initialisation (Steps 1–4), mating selection (Step 9), variation operation (Step 10), reference point update (Step 11) and population update (Step 12) follow the practice in Li and Zhang (2009).

Additional computational costs of AdaW (in comparison with the basic MOEA/D) are from the archiving operations and weight 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 (Li et al., 2016). The computational cost of the weight update is governed by three operations, weight addition (Step 22), weight deletion (Step 23), and neighbouring weight update (Step 24). In the weight 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 weights corresponding to. The computational complexity of finding the neighbours of the L weights 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 deletion, considering the situation that one solution shared by multiple weights requires $O(LN)$ comparisons and removing the weights which are associated with crowded solutions requires $O(m(L+N)^2)$ comparisons (Li et al., 2016). Finally, after the weight deletion completes, updating the neighbours of each weight 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 (Zhang and Li, 2007).

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Algorithm 1 The Algorithm AdaW

Require: N (size of the evolutionary population P, i.e., size of the weight 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 weights W.
2: Calculate the reference point according to P.
3: Determine the neighbours of each weight of W.
4: Associate the weights with solutions in the population randomly.
5: Place the nondominated solutions of P into the archive A.
6: $Gen \leftarrow 1$.
7: **while** $Gen < Gen_{max}$ **do**
8: **for** each subproblem (weight) $w \in W$ **do**
9: Mating selection for w.
10: Generate a new solution p by using variation operators on the solutions in the mating pool.
11: Reference point update by p.
12: Population update by p.
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 3.2).
20: **end if**
21: **if** $Gen = Gen_{max} \times 5\% \wedge Gen < Gen_{max} \times 90\%$ **then**
22: Generate and find the promising, undeveloped weights, and add them into W (Section 3.3 and Section 3.4).
23: Delete the poorly-performed weights until the size of W is reduced to N (Section 3.5).
24: Update the neighbours of each weight of W.
25: **end if**
26: $Gen \leftarrow Gen + 1$.
27: **end while**
28: **return** P

4 Results

Three state-of-the-art weight adaptation approaches, A-NSGA-III (Jain and Deb, 2014), RVEA (Cheng et al., 2016) and MOEA/D-AWA (Qi et al., 2014), along with the baseline MOEA/D (Li and Zhang, 2009), 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 scalarising function was used in which "multiplying the weight" was replaced with "dividing the weight" in order to obtain more uniform solutions (Qi et al., 2014; Deb and Jain, 2014).

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 (Van Veldhuizen, 1999; Zitzler et al., 2000; Deb et al., 2005; Deb and Saxena, 2005; Deb and Jain, 2014; Jain and Deb, 2014; Cheng et al., 2017). 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

[3]The codes of all the peer algorithms were from `http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html` (Tian et al., 2017b).

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Table 2: Settings and properties of test problems. m and d denote the number of objectives and decision variables, respectively

Problem	m	d	Properties	Problem	m	d	Properties
DTLZ1	3	7	Simplex-like, Linear, Multimodals	DTLZ5	3	12	Degenerate, Concave
DTLZ2	3	12	Simplex-like, Concave	VNT2	3	2	Degenerate, Convex
CDTLZ2	3	12	Simplex-like, Convex	SDTLZ1	3	7	Scaled, Simplex-like, Linear, Multimodals
IDTLZ1	3	7	Inverted simplex-like, Linear, Multimodals	SDTLZ2	3	12	Scaled, Simplex-like, Concave
IDTLZ2	3	12	Inverted simplex-like, Concave	SCH2	2	1	Scaled, Discontinuous, Convex
SCH1	2	1	Highly nonlinear, Convex	DTLZ2-10	10	19	Many-objective, Simplex-like, Concave
FON	2	2	Highly nonlinear, Concave	IDTLZ1-10	10	19	Many-objective, Inverted simplex-like, Linear, Multimodals
ZDT3	2	30	Disconnected, Mixed	DTLZ5(2,10)	10	19	Many-objective, Degenerate, Concave
DTLZ7	3	22	Disconnected, Mixed, Multimodal				

(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 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 (Van Veldhuizen, 1999; Zitzler et al., 2000; Deb et al., 2005; Deb and Saxena, 2005; Deb and Jain, 2014; Jain and Deb, 2014; Cheng et al., 2017).

To compare the performance of the algorithms, the inverted generational distance (IGD) (Coello and Sierra, 2004) and hypervolume (Zitzler and Thiele, 1999) were used. IGD, which measures the average distance from uniformly distributed points along the Pareto front to their closest solution in a set, can provide a comprehensive assessment of the convergence and diversity of the set. To calculate IGD, we need a reference set that well represents the problem's Pareto front, and the assessment result may heavily depend on the specification of the reference set (Ishibuchi et al., 2018b). In most of the test problems used in our study, their Pareto fronts are known (e.g., the ZDT and DTLZ suites and the variants of the DTLZ problems). For them, we considered around 10,000 evenly-distributed points along the Pareto front as the reference set. For remaining test problems, their reference sets were obtained at the website `http://delta.cs.cinvestav.mx/~ccoello/EMOO/`.

However, IGD is not Pareto-compliant in the sense that it does not certainly prefer a Pareto-dominating set to a Pareto-dominated set. So we also used the Pareto-complaint indicator hypervolume. Hypervolume measures the volume of the objective space enclosed by a solution set and a reference point. Following the practice in Li et al. (2014c), the reference point of DTLZ1, DTLZ2, SCH1, FON, ZDT3, DTLZ7, DTLZ5, VNT2 and SCH2 was set to $(1,1,1)$, $(2,2,2)$, $(5,5)$, $(2,2)$, $(2,2,7)$, $(2,2,2)$, $(5,16,12)$ and $(2,17)$, respectively. For the remaining problems, we considered common settings,

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

i.e., $(2, 2, 2)$ for CDTLZ2 and IDTLZ2, $(1, 1, 1)$ for IDTLZ1, $(2, 2, ..., 2)$ for DTLZ2-10 and DTLZ5(2,10), $(1, 1, ..., 1)$ for IDLTZ1, $(0.55, 5.5, 55)$ for SDTLZ1, and $(1.1, 11, 110)$ for SDTLZ2. Note that it is not necessary to normalise the solution set when measuring the hypervolume value for scaled problems, provided that the range of the Pareto front is taken into account in setting the reference point (Li and Yao, 2018).

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) (Agrawal et al., 1995) and polynomial mutation (PM) (Deb, 2001) (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 weights cannot be set arbitrarily. For a set of uniformly-distributed weights 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 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 3.6, the time of updating the weights 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 3 and 4 give the IGD and hypervolume results (mean and standard deviation) of the five algorithms on all the 17 problems, respectively. The better mean for each problem was highlighted in boldface. To have statistically sound conclusions, the Wilcoxon's rank sum test (Zitzler et al., 2008) 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

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Table 3: IGD results (mean and SD) of the five algorithms. The best 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)**
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)**

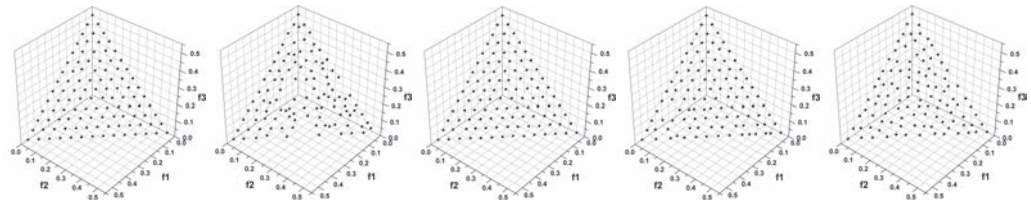
"†" indicates that the result of the peer algorithm is significantly different from that of AdaW at a 0.05 level by the Wilcoxon's rank sum test.

Table 4: Hypervolume results (mean and SD) of the five algorithms. The best mean for each case is highlighted in boldface.

Property	Problem	MOEA/D	A-NSGA-III	RVEA	MOEA/D-AWA	AdaW
Simplex-like	DTLZ1	**9.738E-01(1.8E-04)**†	9.710E-01(3.5E-03)†	9.733E-01(1.1E-03)	9.734E-01(7.2E-04)	9.735E-01(2.7E-04)
	DTLZ2	7.418E+00(1.2E-04)†	7.412E+00(4.6E-03)	7.418E+00(5.1E-04)†	**7.420E+00(1.1E-03)**†	7.412E+00(6.7E-03)
	CDTLZ2	7.947E+00(7.9E-05)†	7.937E+00(7.0E-03)†	7.944E+00(1.6E-03)†	7.949E+00(4.5E-04)†	**7.952E+00(1.7E-04)**
Inverted simplex-like	IDTLZ1	6.808E-01(1.4E-03)†	6.646E-01(4.1E-03)†	6.159E-01(5.7E-02)†	**6.844E-01(1.1E-03)**	6.839E-01(2.1E-03)
	IDTLZ2	6.557E+00(2.1E-03)†	6.617E+00(3.4E-02)†	6.615E+00(7.0E-03)†	6.687E+00(1.4E-02)†	**6.728E+00(3.7E-03)**
Highly nonlinear	SCH1	2.224E+01(2.7E-03)†	2.223E+01(1.9E-02)†	2.225E+01(1.4E-02)†	2.227E+01(3.5E-03)	**2.227E+01(8.2E-04)**
	FON	3.062E+00(2.1E-04)†	3.057E+00(7.3E-03)†	3.060E+00(3.4E-04)†	**3.062E+00(8.4E-04)**	3.061E+00(3.2E-03)
Disconnect	ZDT3	4.808E+00(4.4E-03)†	4.517E+00(3.2E-01)†	4.299E+00(3.3E-01)†	4.632E+00(3.3E-01)†	**4.812E+00(5.0E-03)**
	DTLZ7	1.341E+01(1.4E-03)†	1.328E+01(6.3E-02)†	1.310E+01(5.5E-02)†	1.303E+01(1.1E+00)†	**1.347E+01(2.7E-02)**
Degenerate	DTLZ5	6.076E+00(1.8E-05)†	6.056E+00(3.4E-02)†	5.936E+00(2.4E-02)†	6.091E+00(8.9E-04)†	**6.102E+00(6.0E-03)**
	VNT2	1.878E+00(2.0E-04)†	1.905E+00(3.0E-03)†	1.882E+00(5.9E-03)†	1.912E+00(6.7E-04)†	**1.916E+00(2.8E-04)**
Scaled	SDTLZ1	1.084E+02(8.2E+00)†	1.392E+02(6.5E-01)	1.283E+02(2.4E+01)†	1.216E+02(3.0E+00)†	**1.393E+02(1.1E+00)**
	SDTLZ2	5.736E+02(1.6E+01)†	7.400E+02(1.8E+00)†	7.437E+02(2.6E+00)†	6.353E+02(2.6E+01)†	**7.483E+02(1.2E+00)**
	SCH2	3.794E+01(1.1E-03)†	3.815E+01(1.8E-01)†	3.814E+01(6.9E-03)†	3.813E+01(1.1E-02)†	**3.825E+01(4.2E-03)**
Many objectives	DTLZ2-10	1.024E+03(6.5E-02)	1.023E+03(8.9E-01)†	**1.024E+03(1.2E-02)**†	1.023E+03(7.4E-01)†	1.024E+03(2.1E-02)
	IDTLZ1-10	6.351E-03(1.8E-04)†	1.159E-02(7.9E-04)†	4.505E-03(3.6E-04)†	9.189E-03(5.6E-04)†	**2.631E-02(1.2E-03)**
	DTLZ5(2,10)	6.365E+02(5.9E+00)†	5.444E+02(7.8E+01)†	6.362E+02(5.9E+01)†	6.800E+02(1.1E+01)†	**7.067E+02(4.6E-01)**

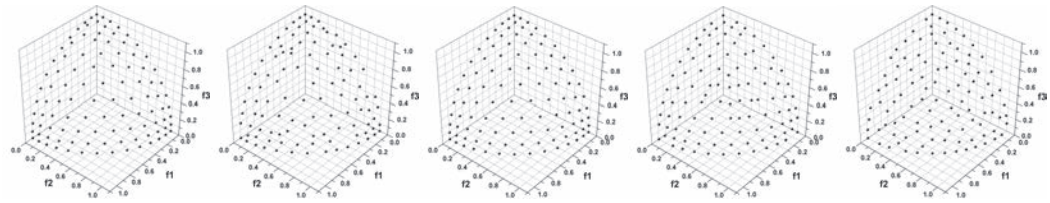
"†" indicates that the result of the peer algorithm is significantly different from that of AdaW at a 0.05 level by the Wilcoxon's rank sum test.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



(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

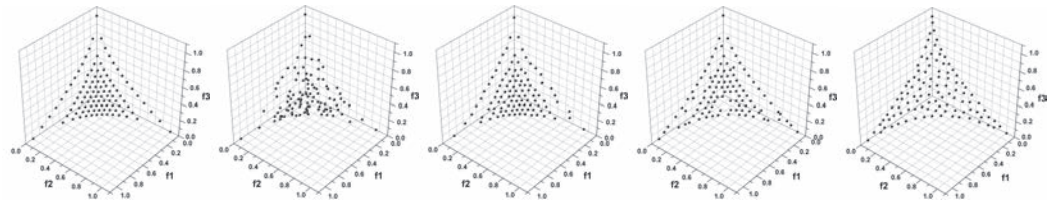
Figure 4: The final solution set of the five algorithms on DTLZ2.

set of AdaW not being so "regular" as that of the other three algorithms. An interesting observation is that A-NSGA-III (adapting the weights in NSGA-III) appears to struggle in maintaining the uniformity of the solutions, especially for DTLZ1 and CDTLZ2. This indicates that adapting the weights 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 (Deb and Jain, 2014). 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 better IGD and hypervolume results as shown in Tables 3 and 4. Another note is on the preservation of the extreme solutions (e.g., $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$ for DTLZ2) in AdaW. Preserving the extreme solutions is not an trivial task in a weight adaptation process, as their corresponding extreme weights can be easily discarded, particularly after the normalisation of the weights (Ishibuchi et al., 2017a). Interestingly, as shown in the three figures, AdaW is doing well in preserve the extreme solutions. This occurrence is due to the fact that the extreme weights can be seen to be located in relatively sparse regions (as not many weights around them), and thus they are unlikely to be eliminated during the weight deletion process. But they are not certainly preserved — they do be missing in some situation.

4.2 On Inverted Simplex-like Pareto Fronts

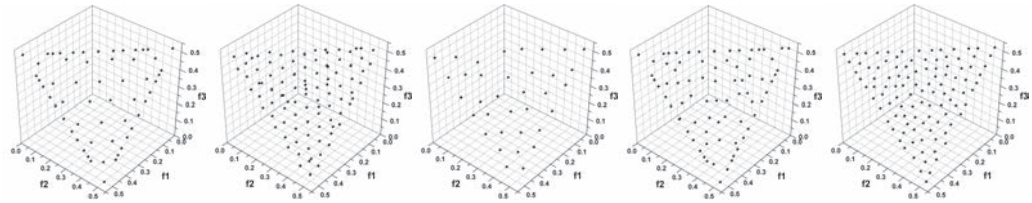
The proposed AdaW has shown a clear advantage over its competitors on this group. Figures 6–7 plot the final solution set of the five algorithms on IDTLZ1 and IDTLZ2, respectively. As shown, many solutions of MOEA/D and MOEA/D-AWA concentrate on the boundary of the Pareto front. The solutions of A-NSGA-III 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

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



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

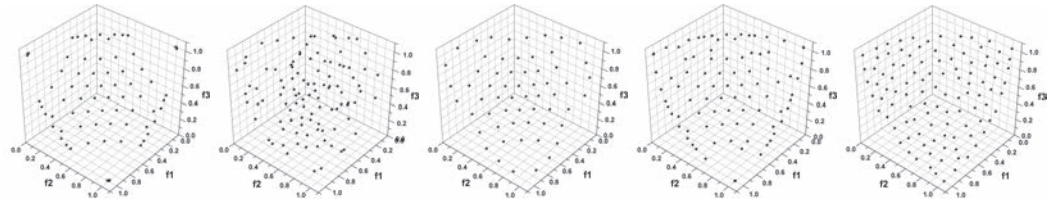
Figure 5: The final solution set of the five algorithms on the convex DTLZ1.



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

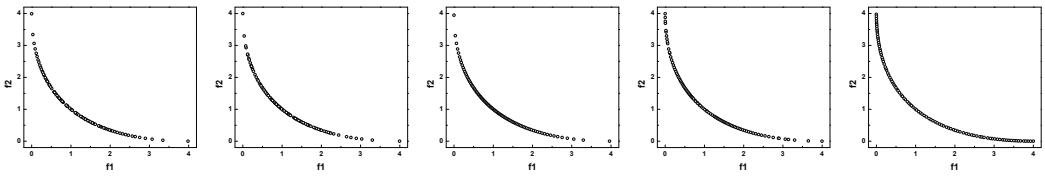
Figure 6: The final solution set of the five algorithms on the inverted DTLZ1.

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. However, an interesting observation is that when looking at the hypervolume results on IDTLZ1 in Table 4, MOEA/D-AWA is preferred to AdaW. This is because the optimal distribution of solutions for hypervolume maximisation may not be even, as shown in Ishibuchi et al. (2018a); Li et al. (2015d).



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

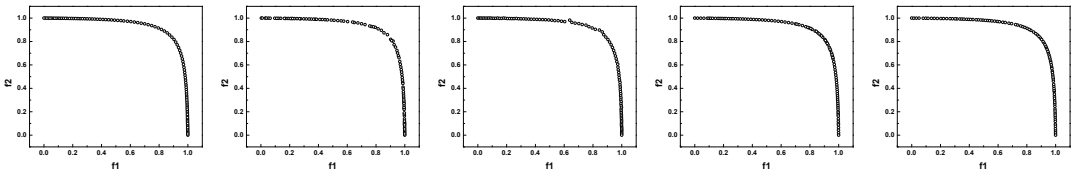
Figure 7: The final solution set of the five algorithms on the inverted DTLZ2.



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

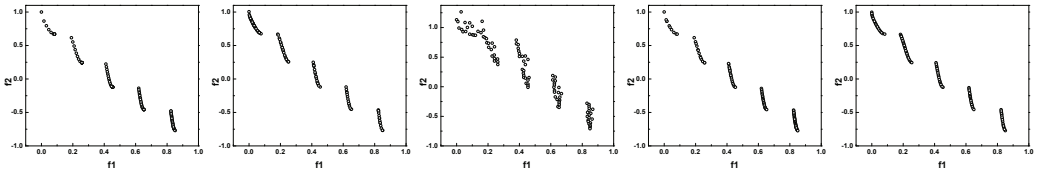
Figure 8: The final solution set of the five algorithms on SCH1.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



(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.



(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.

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 adaptations are introduced.

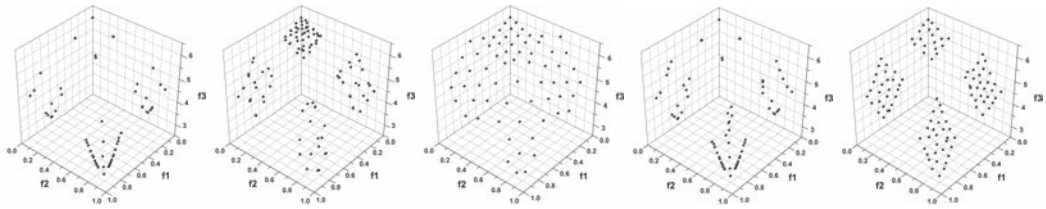
4.4 On Disconnected Pareto Fronts

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), or struggle to maintain the uniformity (A-NSGA-III), or produce some dominated solutions (RVEA).

4.5 On Degenerate Pareto Fronts

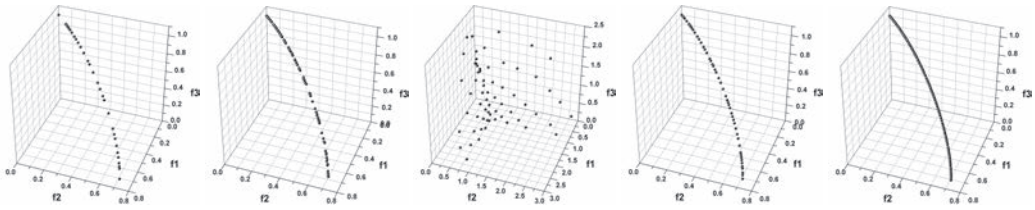
Problems with a degenerate Pareto front poses a big challenge to decomposition-based approaches since the ideal weight set is located in a lower-dimensional manifold than its initial setting (Li et al., 2018). On this group of problems, the proposed algorithm has shown a significant advantage over its 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,

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



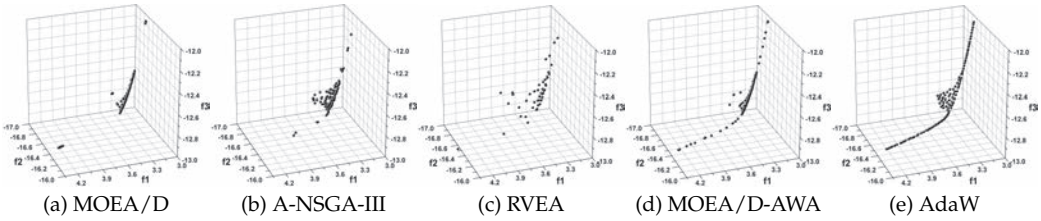
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

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

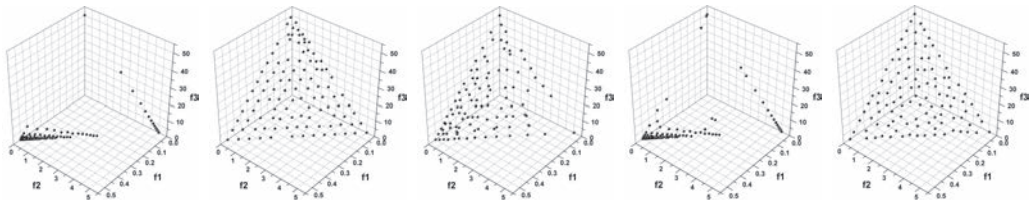
Figure 13: The final solution set of the five algorithms on VNT2.

the solution set of AdaW has a good distribution over the whole Pareto front.

4.6 On Scaled Pareto Fronts

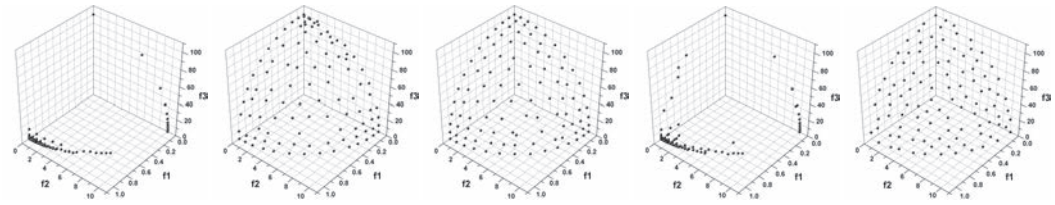
Figures 14–16 plot the final solution set of the five algorithms on SDTLZ1, SDTLZ2 and SCH2, respectively. For the first two problems, AdaW, A-NSGA-III and RVEA work fairly well, but the solutions obtained by RVEA are not so uniform as those obtained by the other two algorithms on SDTLZ1. For SCH2 which 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. In fact, all the competitors, except the original MOEA/D, use the normalisation operation in their calculation. However, as pointed out in Ishibuchi et al. (2017a), the normalisation in decomposition-based algorithms may cause the degradation of the diversity of solutions in the population. So, the normalisation may not work on all scaled problems. Interestingly, our algorithm performs well on all the scaled problems. One probable explanation is that AdaW not only considers the normalisation of the current population, but also the normalisation of the archive which stores a set of well-distributed nondominated solutions, and then use the archive to guide the weight update (via a

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



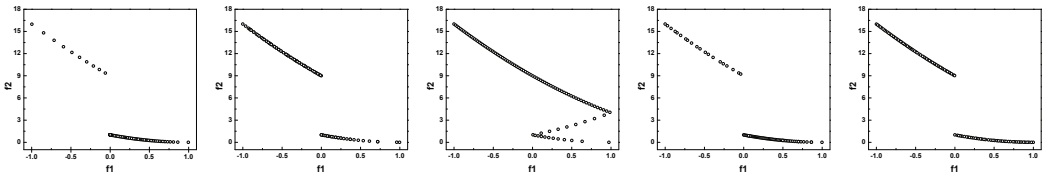
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

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.

comparison between the population and archive). This could avoid the diversity loss of solutions (and their associated weights) during the normalisation process.

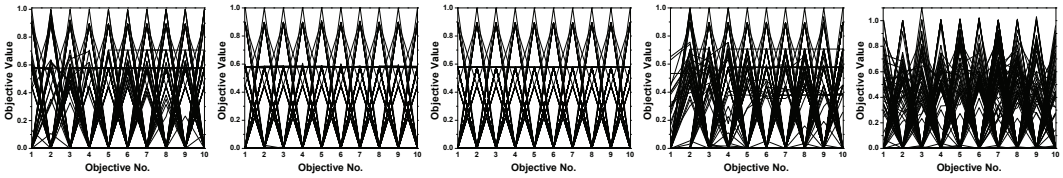
4.7 On Many-Objective Problems

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 exist several solutions of AdaW not fully converging into the Pareto front. We may not be able to conclude the distribution difference of the algorithms by the parallel coordinates plots (Li et al., 2017), but all the algorithms seem to perform similarly according to the IGD and hypervolume results in Tables 3 and 4.

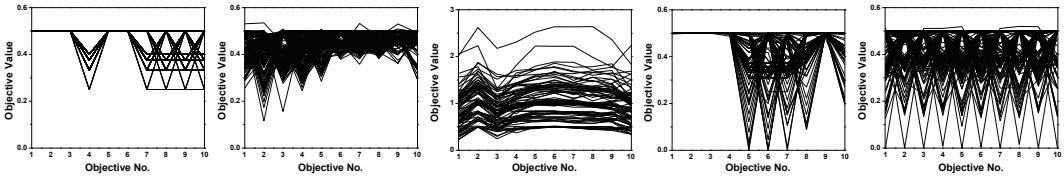
For the many-objective problems whose Pareto front is far from the standard simplex, a clear advantage of AdaW over its competitors is shown (Figures 18 and 19). The

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269



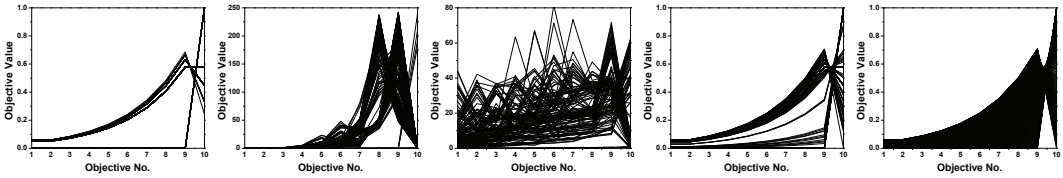
(a) MOEA/D (b) A-NSGA-III (c) RVEA (d) MOEA/D-AWA (e) AdaW

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).

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.

4.8 Discussions

Methods involving weight update need a parameter of controlling the update frequency, except those which change weights every generation, e.g., A-NSGA-III (Jain and Deb, 2014). However, a frequent weight change may lead to the solutions to wander around the search space (Giagkiozis et al., 2013b). Like in MOEA/D-AWA (Qi et al., 2014), we used a percentage value (5%) of the maximum generations/evaluations as the update frequency. And also the algorithm does not allow to change the weights in the last 10% generations/evaluations. In a situation where the maximum generation number is not applicable (i.e., the total number of generations being not as the termination condition of the algorithm, for example, in particular real-world scenarios), we recommend to update the weights every $5 \times m$ generations, where m denotes the number of

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

objectives.

The test problems considered in our experimental studies all are unconstrained. However, constrained MOPs are widely seen in real-world scenarios. There exist several constraint-handling techniques used in decomposition-based EMO, such as the feasibility-first scheme (Fonseca and Fleming, 1998) used in NSGA-III (Deb and Jain, 2014) and RVEA (Cheng et al., 2016), and the epsilon level comparison (Asafuddoula et al., 2012) used in DBEA (Asafuddoula et al., 2015a) and g-DBEA (Asafuddoula et al., 2017). The proposed AdaW can easily incorporate these constraint-handling techniques via slightly modifying the selection operation of the algorithm.

Finally, it is worth pointing out that since AdaW adopts a Pareto nondominated archive set for weight generation, the algorithm may struggle if the archive set cannot well represent the whole Pareto front. The problems proposed in Liu et al. (2014) challenge EMO algorithms in such a way. The nondominated solutions in those problems lie on a very small area of the search space. Once the extremities of the Pareto front are achieved by the algorithm, all other solutions will be dominated. This makes it very difficult to obtain the central part of the front. This difficulty applies to all algorithms that use Pareto dominance as the primary selection criterion (e.g., Pareto-based algorithms) or use a criterion providing higher selection pressure than Pareto dominance (e.g., most indicator-based and decomposition-based algorithms), We leave addressing this issue as an important topic of our future study.

5 Conclusions

Adaptation of the weights during the optimisation process provides a viable approach to enhance existing decomposition-based EMO. This paper proposed an adaptation method to periodically update the weights by contrasting the current evolutionary population with a well-maintained 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 weights. 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 weights evolving along the constant weights is expected to achieve better convergence on problems with many objectives. Another potential improvement is from the weight deletion operation, where one may consider to delete the weight that has the biggest angle from the solution (instead of the one who has the worst scalarising function value), as the ideal case is to see solutions exactly lie on the search directions determined by the weights.

6 Acknowledgement

The authors would like to thank Dr. Liangli Zhen for his help in the experimental study. This work has been supported by the Science and Technology Innovation

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Committee Foundation of Shenzhen (ZDSYS201703031748284), Shenzhen Peacock Plan (KQTD2016112514355531), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), and EPSRC (EP/J017515/1 and EP/P005578/1).

References

Agrawal, R. B., Deb, K., and Agrawal, R. (1995). Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148.

Almeida, C. P., Gonçalves, R. A., Goldbarg, E. F., Goldbarg, M. C., and Delgado, M. R. (2012). An experimental analysis of evolutionary heuristics for the biobjective traveling purchaser problem. *Annals of Operations Research*, 199(1):305–341.

Asafuddoula, M., Ray, T., and Sarker, R. (2015a). A decomposition based evolutionary algorithm for many objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(3):445–460.

Asafuddoula, M., Ray, T., Sarker, R., and Alam, K. (2012). An adaptive constraint handling approach embedded MOEA/D. In *Proceedings of the IEEE Congress Evolutionary Computation (CEC)*, pages 1–8.

Asafuddoula, M., Singh, H. K., and Ray, T. (2015b). Six-sigma robust design optimization using a many-objective decomposition based evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 19(4):490–507.

Asafuddoula, M., Singh, H. K., and Ray, T. (2017). An enhanced decomposition-based evolutionary algorithm with adaptive reference vectors. *IEEE Transactions on Cybernetics*, published online.

Cai, X., Li, Y., Fan, Z., and Zhang, Q. (2015). An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 19(4):508–523.

Cai, X., Mei, Z., and Fan, Z. (2017). A decomposition-based many-objective evolutionary algorithm with two types of adjustments for direction vectors. *IEEE Transactions on Cybernetics*, published online.

Cheng, R., Jin, Y., and Narukawa, K. (2015). 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*, pages 127–140.

Cheng, R., Jin, Y., Olhofer, M., and Sendhoff, B. (2016). A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):773–791.

Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., and Yao, X. (2017). A benchmark test suite for evolutionary many-objective optimization. *Complex & Intelligent Systems*, 3(1):67–81.

Coello, C. A. C. and Sierra, M. R. (2004). A study of the parallelization of a coevolutionary multiobjective evolutionary algorithm. In *Mexican International Conference on Artificial Intelligence (MICAI)*, pages 688–697.

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

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley, New York.

Deb, K. and Jain, H. (2014). 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*, 18(4):577–601.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Deb, K. and Saxena, D. K. (2005). On finding Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. *Kangal report*, 2005011.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In Abraham, A., Jain, L., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Berlin, Germany: Springer.

Derbel, B., Liefooghe, A., Zhang, Q., Aguirre, H., and Tanaka, K. (2016). Multi-objective local search based on decomposition. In *International Conference on Parallel Problem Solving from Nature*, pages 431–441.

Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(1):26–37.

Giagkiozis, I., Purshouse, R. C., and Fleming, P. J. (2013a). Generalized decomposition. In *Evolutionary Multi-Criterion Optimization (EMO)*, pages 428–442.

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

Gu, F. and Cheung, Y.-M. (2018). Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 22(2):211–225.

Gu, F., Liu, H., and Tan, K. C. (2012). A multiobjective evolutionary algorithm using dynamic weight design method. *International Journal of Innovative Computing Information and Control*, 8(5):3677–3688.

Hughes, E. J. (2005). Evolutionary many-objective optimisation: Many once or one many? In *IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 222–227.

Ikeda, K., Kita, H., and Kobayashi, S. (2001). Failure of Pareto-based MOEAs: does non-dominated really mean near to optimal? In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 957–962.

Ishibuchi, H., Doi, K., and Nojima, Y. (2017a). On the effect of normalization in MOEA/D for multi-objective and many-objective optimization. *Complex & Intelligent Systems*, 3(4):279–294.

Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. (2018a). How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary Computation*, 26(3):411–440.

Ishibuchi, H., Imada, R., Setoguchi, Y., and Nojima, Y. (2018b). Reference point specification in inverted generational distance for triangular linear pareto front. *IEEE Transactions on Evolutionary Computation*.

Ishibuchi, H. and Murata, T. (1998). 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):392–403.

Ishibuchi, H., Setoguchi, Y., Masuda, H., and Nojima, Y. (2017b). Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Jain, H. and Deb, K. (2014). 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*, 18(4):602–622.

Jaszkiewicz, A. (2002). On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412.

Jiang, S., Cai, Z., Zhang, J., and Ong, Y. S. (2011). Multiobjective optimization by decomposition with Pareto-adaptive weight vectors. In *2011 International Conference on Natural Computation (ICNC)*, volume 3, pages 1260–1264.

Jiang, S. and Yang, S. (2017). A strength Pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 21(3):329–346.

Jin, Y., Okabe, T., and Sendhoff, B. (2001). Adapting weighted aggregation for multiobjective evolution strategies. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 96–110.

Li, B., Li, J., Tang, K., and Yao, X. (2015a). Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48(1):1–35.

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

Li, H. and Landa-Silva, D. (2011). An adaptive evolutionary multi-objective approach based on simulated annealing. *Evolutionary computation*, 19(4):561–595.

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

Li, K., Deb, K., Zhang, Q., and Kwong, S. (2015c). An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716.

Li, K., Zhang, Q., Kwong, S., Li, M., and Wang, R. (2014a). Stable matching based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 18(6):909–923.

Li, M., Grosan, C., Yang, S., Liu, X., and Yao, X. (2018). Multi-line distance minimization: A visualized many-objective test problem suite. *IEEE Transactions on Evolutionary Computation*, 22(1):61–78.

Li, M., Yang, S., Li, K., and Liu, X. (2014b). Evolutionary algorithms with segment-based search for multiobjective optimization problems. *IEEE Transactions on Cybernetics*, 44(8):1295–1313.

Li, M., Yang, S., and Liu, X. (2015d). A performance comparison indicator for Pareto front approximations in many-objective optimization. In *Proceedings of the 17th Confernece on Genetic and Evolutionary Computation (GECCO)*, pages 703–710.

Li, M., Yang, S., and Liu, X. (2016). Pareto or non-Pareto: Bi-criterion evolution in multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(5):645–665.

Li, M., Yang, S., Liu, X., and Zheng, J. (2014c). ETEA: A Euclidean minimum spanning tree-based evolutionary algorithm for multiobjective optimization. *Evolutionary Computation*, 22(2):189–230.

Li, M. and Yao, X. (2018). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys*, under review.

Li, M., Zhen, L., and Yao, X. (2017). How to read many-objective solution sets in parallel coordinates. *IEEE Computational Intelligence Magazine*, 12(4):88–97.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

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

Liu, H.-L., Chen, L., Zhang, Q., and Deb, K. (2017). Adaptively allocating search effort in challenging many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*, published online.

Liu, H.-L., Gu, F., and Zhang, Q. (2014). Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation*, 18(3):450–455.

Ma, X., Qi, Y., Li, L., Liu, F., Jiao, L., and Wu, J. (2014). MOEA/D with uniform decomposition measurement for many-objective problems. *Soft Computing*, 18(12):2541–2564.

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

Mei, Y., Tang, K., and Yao, X. (2011). A memetic algorithm for periodic capacitated arc routing problem. *IEEE Transactions on Systems Man and Cybernetics-Part B-Cybernetics*, 41(6):1654–1667.

Murata, T., Ishibuchi, H., and Gen, M. (2001). Specification of genetic search directions in cellular multi-objective genetic algorithms. In *Evolutionary Multi-Criterion optimization*, pages 82–95. Springer.

Qi, Y., Ma, X., Liu, F., Jiao, L., Sun, J., and Wu, J. (2014). MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264.

Shim, V. A., Tan, K. C., and Cheong, C. Y. (2012). 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)*, 42(5):682–691.

Tan, Y., Jiao, Y., Li, H., and Wang, X. (2013). MOEA/D + uniform design: A new version of MOEA/D for optimization problems with many objectives. *Computers & Operations Research*, 40(6):1648–1660.

Tian, Y., Cheng, R., Zhang, X., Cheng, F., and Jin, Y. (2017a). An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility. *IEEE Transactions on Evolutionary Computation*, published online.

Tian, Y., Cheng, R., Zhang, X., and Jin, Y. (2017b). Platemo: A matlab platform for evolutionary multi-objective optimization. *arXiv preprint arXiv:1701.00879*.

Trivedi, A., Srinivasan, D., Sanyal, K., and Ghosh, A. (2017). A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462.

Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Wang, L., Zhang, Q., Zhou, A., Gong, M., and Jiao, L. (2016a). Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*, 20(3):475–480.

Wang, R., Purshouse, R. C., and Fleming, P. J. (2013). 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)*, pages 695–702.

Wang, R., Purshouse, R. C., and Fleming, P. J. (2015). Preference-inspired co-evolutionary algorithms using weight vectors. *European Journal of Operational Research*, 243(2):423–441.

Evolutionary Computation Just Accepted MS.
https://doi.org/10.1162/evco_a_00269

Wang, R., Xiong, J., Ishibuchi, H., Wu, G., and Zhang, T. (2017a). On the effect of reference point in moea/d for multi-objective optimization. *Applied Soft Computing*, 58:25–34.

Wang, R., Zhang, Q., and Zhang, T. (2016b). Decomposition-based algorithms using Pareto adaptive scalarizing methods. *IEEE Transactions on Evolutionary Computation*, 20(6):821–837.

Wang, Z., Zhang, Q., Li, H., Ishibuchi, H., and Jiao, L. (2017b). On the use of two reference points in decomposition based multiobjective evolutionary algorithms. *Swarm and Evolutionary Computation*, 34:89–102.

Xiang, Y., Peng, J., Zhou, Y., Li, M., and Chen, Z. (2017a). An angle based constrained many-objective evolutionary algorithm. *Applied Intelligence*, 47(3):705–720.

Xiang, Y., Yuren, Z., Li, M., and Chen, Z. (2017b). A vector angle based evolutionary algorithm for unconstrained many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 21(1):131–152.

Yuan, Y., Xu, H., Wang, B., and Yao, X. (2016). A new dominance relation based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37.

Zhang, C., Tan, K. C., Lee, L. H., and Gao, L. (2017). Adjust weight vectors in moea/d for bi-objective optimization problems with discontinuous pareto fronts. *Soft Computing*, published online.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhou, A. and Zhang, Q. (2016). Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(1):52–64.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

Zitzler, E., Knowles, J., and Thiele, L. (2008). Quality assessment of Pareto set approximations. In Branke, J., Deb, K., Miettinen, K., and Slowinski, R., editors, *Multiobjective Optimization*, volume 5252, pages 373–404. Springer Berlin / Heidelberg.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.