Reference Point Based Multi-objective Optimization Through Decomposition

Asad Mohammadi
School of Computer Science and IT
RMIT University
Melbourne, Australia
Email: asadmoh1985@gmail.com

Mohammad Nabi Omidvar
School of Computer Science and IT
RMIT University
Melbourne, Australia
Email: mohammad.omidvar@rmit.edu.au

Xiaodong Li
School of Computer Science and IT
RMIT University
Melbourne, Australia
Email: xiaodong.li@rmit.edu.au

***Abstract*—In this paper we propose a user-preference based evolutionary algorithm that relies on decomposition strategies to convert a multi-objective problem into a set of single-objective problems. The use of a reference point allows the algorithm to focus the search on more preferred regions which can potentially save considerable amount of computational resources. The algorithm that we proposed, dynamically adapts the weight vectors and is able to converge close to the preferred regions. Combining decomposition strategies with reference point approaches paves the way for more effective optimization of many-objective problems. The use of a decomposition method alleviates the selection pressure problem associated with dominance-based approaches while a reference point allows a more focused search. The experimental results show that the proposed algorithm is capable of finding solutions close to the reference points specified by a decision maker. Moreover, our results show that high quality solutions can be obtained using less computational effort as compared to a state-of-the-art decomposition based evolutionary multi-objective algorithm.**

I. Introduction

Evolutionary Multi-objective Optimization (EMO) [1], [2] techniques have been successfully applied to many real-world applications in engineering design, production and manufacturing [3]. In recent years there has been a growing demand for tackling problems with many conflicting objectives [4], however, the performance of evolutionary multi-objective approaches degrades rapidly as the number of objectives increases [5], [6].

More specifically, there are three major challenges faced by EMO algorithms when dealing with many-objective problems [7]. Visualizing the Pareto-front when there are more than three objectives is virtually impossible. This makes it difficult for the decision maker (DM) to get a visual sense of the solutions to be able to select a preferred one. The second difficulty is that the number of solutions required to approximately generate the full Pareto-optimal front increases exponentially with respect to the number of objectives. Finally, when the number of objectives increases, most of the solutions even in the initial randomly generated population are non-dominated to each other, hence there will not be enough selection pressure to propel the solutions towards the Pareto-optimal front [6], [5].

It has been shown that the dominance-based approaches such as NSGA-II [8], SPEA [9] and MOGA [10] suffer the most from the selection pressure problem when dealing with high dimensional objective spaces [11], [5], [6]. By applying decomposition strategies borrowed from multi-criterion decision making [12] to convert a multi-objective problem into a single-objective problem, we can alleviate the selection pressure problem imposed on dominance-based evolutionary algorithms. This eliminates the need for a dominance operator which is the prime source of low selection pressure. Two major techniques, which are widely used to convert a multi-objective problem into a single-objective problem, are the weighted-sum and the Tchebycheff approaches [12]. These strategies mainly rely on assigning weight values to the objective functions in order to obtain a solution on the Pareto-optimal front. In particular, for each set of weight values often a solution can be found on the Pareto-optimal front. Thus, by using various sets of weight values, different solutions can be obtained on the Pareto-optimal front. The well-known evolutionary approaches that rely on a decomposition strategy to generate a set of single-objective problems from a multi-objective problem are MOGLS [13] and MOEA/D [14].

Although the decomposition based approaches such as MOGLS and MOEA/D suffer less than the dominance-based approaches from the selection pressure problem and can potentially scale better to higher objective spaces, the number of single-objective sub-problems that they have to solve in order to approximate the entire Pareto-optimal front grows exponentially as the number of objectives increases. In order to reduce this computational cost, one can focus the search on a preferred region of the Pareto-optimal front rather than spending a considerable amount of computational resources to generate the entire front. User preference based approaches [15] are ideal for such scenarios where the DM provides, as input, a reference point in the objective space as a representative for his/her preferred region. By having access to such a reference point, the algorithm can use the computational budget more effectively by searching in the more desired regions. The amount of computational resources that can be potentially saved using user-preference based approaches is magnified when dealing with many-objective problems.

In this paper, we propose a multi-objective optimization algorithm that integrates the user preference with decomposition based EMO algorithms in order to provide a mechanism

for tackling many-objective problems more efficiently. In particular, the proposed algorithm has the following major advantages:

- Less susceptible to the selection pressure problem resulting from the use of dominance comparisons, by using a decomposition based method;
- More computationally effective by searching the regions which are preferred by the decision maker;
- Faster convergence to the Pareto-optimal front;
- Better scalability to higher objective spaces.

The organization of the rest of this paper is as follows. In Section II we explain some preliminaries. Section III contains a brief description of some related studies. The proposed algorithm is described in Section IV. The experimental results are presented and analyzed in Section V and finally, Section VI concludes the paper.

II. Background

A. Multi-objective Optimization

A multi-objective optimization problem is defined as follows:

$$F(\vec{x}) = \langle f_i(\vec{x}), \ldots, f_m(\vec{x}) \rangle \,, \tag{1}$$

where $f_i(\vec{x})$ is the i-th objective from a set of m objective functions and $\vec{x} = \langle x_1, \ldots, x_n \rangle$ is the decision vector and n is the dimensionality of the decision space. Since two solutions in a multi-objective problem can be compared with respect to various objectives, the concept of *dominance* has been introduced for deciding which solutions should be preferred to others. A solution $\vec{x}$ is said to dominate solution $\vec{x}'$ if it is as good as $\vec{x}'$ with respect to all of the objectives and strictly better on at least one. This is shown as $\vec{x} \prec \vec{x}'$ and is more formally defined as follows (minimization of the objectives has been assumed):

$$\vec{x} \prec \vec{x}' \iff \forall i,\ f_i(\vec{x}) \leq f_i(\vec{x}') \ \wedge \exists j,\ f_j(\vec{x}) < f_j(\vec{x}').$$

A non-dominated set is a set of solutions where all of its members do not dominate each other. A Pareto-optimal set is a non-dominated set where its members dominate all other possible solutions. The mapping of all possible Pareto-optimal solutions into the objective space form a curve (surface) which is commonly referred to as Pareto-optimal front. A Pareto-optimal front is said to be convex if and only if the connecting line between any two points on the Pareto-optimal front lies above it and non-convex otherwise.

B. Weighted-Sum Approach

The weighted-sum method is one of the simplest and well-known strategies to convert a multi-objective problem into a single-objective problem [12]. Although this approach is simple and easy to apply, choosing a weight vector that results in finding a solution near the user reference point is not a straightforward task. Choosing a weight value for each of the objectives depends on their relative importance in the context of the actual problem which is being solved. Moreover, in order to have a fair scaling of the objectives, they first need to be normalized [1]. A compound objective function is the sum of the weighted normalized objectives which is defined as follows:

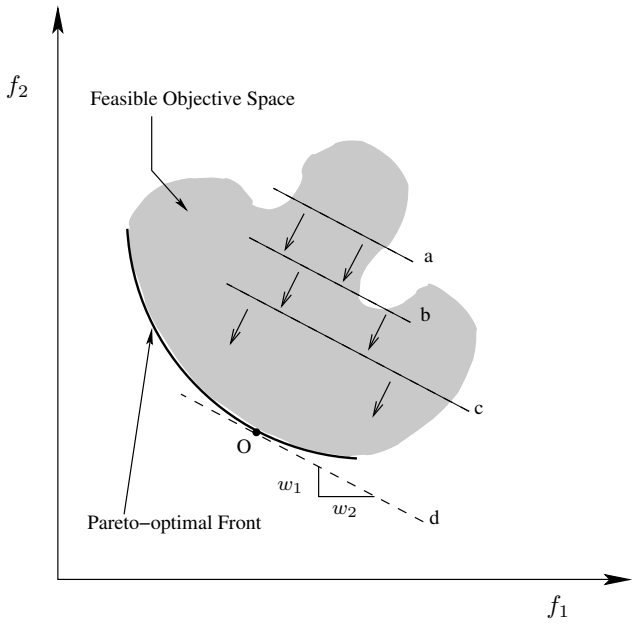


Fig. 1. This figure shows the effect of updating the weight vectors in moving the solutions closer to the desired region.

$$\text{minimize } g^{\text{ws}} = \sum_{i=1}^{m} w_i f_i(\vec{x}) \,, \tag{2}$$

where $0 \leq w_i \leq 1$, m is the number of objectives and w_i is the weight value for for the i-th objective function. It is customary to choose weights such that they add up to one. It has been proved that for any Pareto-optimal solution $\vec{x^\star}$ of a *convex* multi-objective problem, there exists a positive weight vector such that $\vec{x}^\star$ is a solution to Equation (2) [12]. For any given set of weight values, Equations (2) will form a hyperplane in $\mathbb{R}^m$ for which the location is identified by the objective values which are subsequently dependent on the input vector $\vec{x}$ and the orientation of the plane is determined by the weight values w_i. In the special case of having two objectives, the g^{ws} will take the form of a straight line for which the slope is determined by the weight vector as depicted in Figure 1. The effect of minimizing g^{ws} is to push this line as close as possible to the Pareto-optimal front until a unique solution is obtained. For example in Figure 1 the solutions lie on the line 'a' and as they improve during the evolution, they move in the feasible region towards the Pareto-optimal front until an optimum solution ('O') is obtained. Lines 'a' through 'd' show how the improvement of solutions will result in the movement on line (hyperplane in higher dimensions) until it becomes tangential to the Pareto-optimal front at point 'O'. A major advantage of the weighted-sum approach is its simplicity and effectiveness, however, it is less effective when dealing with non-convex Pareto-optimal fronts [1].

C. Tchebycheff Approach

Another decomposition method, Tchebycheff [12], converts a multi-objective problem into a single-objective problem using the following equation:

$$\text{minimize } g^{\text{te}}(\vec{x}) = \max\{w_i|f_i(\vec{x}) - z_i^\star|\}, \qquad (3)$$

where $i \in \{1, \ldots, m\}$, m is the number of objectives, $\vec{z}^\star = \langle z_1^\star, \ldots, z_m^\star \rangle$ is the ideal point such that $z_i^\star = \min\{f_i(\vec{x})\}$ for all possible decision vectors $\vec{x}$, and w_i is a weight value. Similar to the weighted-sum approach, by minimizing $g^{\text{te}}(\vec{x})$ for any given weight vector $\vec{w}$ a solution can be found on the Pareto-optimal front. Therefore the complete Pareto-optimal front can be generated by minimizing $g^{\text{te}}(\vec{x})$ with various weight vectors. A major advantage of the Tchebycheff approach is that – unlike the weighted-sum approach – it can be used to obtain a Pareto-optimal solution on problems with a non-convex Pareto-optimal front.

D. Decomposition Based EMO Methods

Decomposition based EMO algorithms rely on a decomposition strategy such as weighted-sum or Tchebycheff to convert a multi-objective problem into a single-objective problem which is then optimized using any Evolutionary Algorithm. Among decomposition based EMO approaches, MOGLS [13] and MOEA/D [14] enjoyed more popularity than others. Since the algorithm that we propose in this paper is mainly based on MOEA/D, here we provide a brief description of its algorithm.

In MOEA/D, each individual is associated with a unique weight vector which is used for the decomposition of the problem. In addition to the weight vector, a neighborhood is formed for each individual which contains the indices of T closest neighbors based on the closeness of their corresponding weight vectors. In the mating process, the individuals from the same neighborhood are combined using suitable genetic operators to generate new offspring. Since MOEA/D relies on the individuals' neighborhood rather than the whole population to generate new offspring, it benefits from a lower computational cost compared to its counterparts such as MOGLS [13] and NSGA-II [8].

III. Related Works

The idea of a user-preference based optimization in the context of classical multi-criterion decision making was proposed by Wierzbicki [16]. Such approaches allow the user to specify their desired region through a reference point. Since the reference point allows a more focused search, considerable amount of computational resources can be saved. These inherent advantages prompted the researchers to apply user-preference based approaches to the field of Evolutionary Multi-objective Optimization (EMO) [1], [2], [17].

Deb [18] made the first attempt to apply EMO approaches to classical goal programming [19]. The ability of evolutionary algorithms in finding multiple solutions made it possible to simultaneously minimize the deviations from individual goals which eliminated the need for a user-defined weight vector. The effectiveness of this evolutionary approach to goal programming has been verified empirically using several test problems as well as a real-world engineering design problem [18]. In this approach, the emphasis was mainly on goal satisfaction and the algorithm does not try to find Pareto-optimal solutions close to the supplied goal.

In another study, Deb and Sundar [7] proposed a method which combined a reference point strategy with NSGA-II [8] in order to simultaneously find multiple solutions on the Pareto-optimal front close to each of the user-supplied reference points. This new algorithm which was called R-NSGA-II used an extra parameter called ϵ in order to control the crowding of solutions near the provided reference points. In the same study, a predator-prey approach was also investigated to find solutions close to a desired region. However, it has been shown that the predator-prey approach does not scale well as the number of objectives increases [7]. Although R-NSGA-II has been applied successfully to problems with up to ten objectives, it can potentially suffer from the selection pressure problem common to all dominance-based approaches when applied to many-objective problems.

In the context of decomposition based approaches, Cvetkovic and Parmee [20] used a modified dominance operator to account for the importance weights associated to the objectives. Since the dominance operator does not specify the extent to which one solution is better than the other, this algorithm has been reported to be hard to control and only provides a very coarse control over the desired region [7]. Jin and Sendhoff [21] used dynamic weighted aggregation method to decompose a multi-objective problem into a single-objective problem. They also dynamically updated the weight values to obtain desired solutions. In this approach, the users cannot provide a reference point and the desired region can only be coarsely specified by specifying the relative importance of the objectives.

Branke *et al.* [22] proposed Guided Multi-Objective Evolutionary Algorithm (G-MOEA) which guides the search to regions on Pareto-optimal front which are of interest to the decision maker. Since it is difficult for the decision makers to determine the weight of each objective directly, instead, they provide a maximum acceptable trade-off between each objective pair. The definition of dominance is modified accordingly in order to favor the solutions closer to the region of interest. Although this approach allows a faster convergence, it is very difficult for the user to provide a set of pair-wise trade-off values for the objectives when dealing with many-objective problems.

A biased fitness sharing technique has been proposed by Deb [23] where the search was focused on a desired region by an importance weight vector that specifies the relative importance of the objectives. The drawback of this technique is that it cannot obtain a set of solutions anywhere on the Pareto-optimal front in a controlled manner [23].

IV. Proposed Approach

In this section we describe the details of a reference point based evolutionary multi-objective optimization prob-

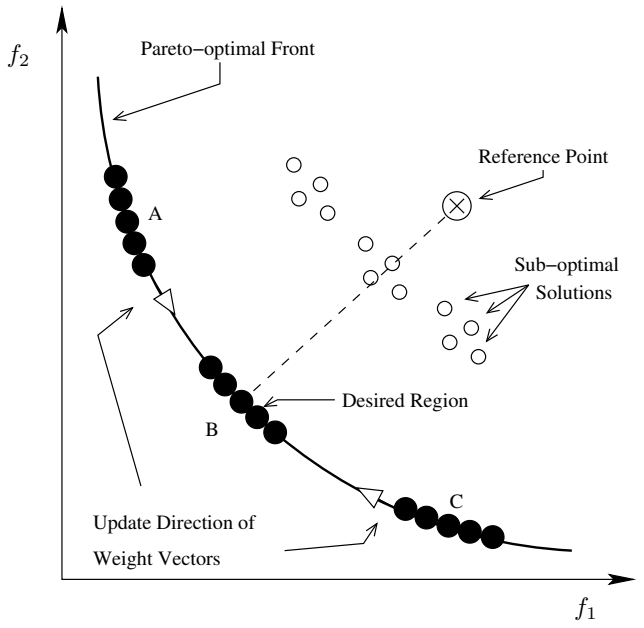


Fig. 2. This figure shows the effect of updating the weight vectors in moving the solutions closer to the desired region.

lem through decomposition which we refer to as the R-MEAD algorithm hereafter. As it was mentioned earlier in Section I, we aimed at designing an algorithm that scales better when the number of objectives increases. Therefore, instead of tackling a multi-objective problem directly, we first decompose it into a set of single-objective problems using a decomposition strategy such as weighted-sum or Tchebycheff. However, unlike decomposition methods such as MOGLS [13] and MOEA/D [14] where a set of weight vectors is generated to cover the *entire* Pareto-optimal front, we generate a smaller set of weight vectors just to find a number of solutions close to the user-supplied reference. Theoretically, if we manage to find a suitable weight vector, it is possible to find a solution close to the reference point. However, finding an optimum weight vector that results in a solution close to the reference point is not trivial. In order to solve this problem we initially pick a sub-optimum weight vector and adapt it in the course of evolution until it results in a solution as close as possible to the user-provided reference point as well as the Pareto-optimal front.

Figure 2 shows how updating the weight vector might result in solutions closer to the user's desired region. The region B on Figure 2 shows the desired region close to the reference point that we wish to find. Regions A and C show several solutions which are obtained on the Pareto-optimal front far from the desired region (B). The white solid arrows show the effect of updating the weight vectors. Note that the optimization of the population and the weight vectors happen simultaneously and as a result, the sub-optimal solutions which are shown in white circle on Figure 2 might move directly towards the desired region on the Pareto-optimal front. However, if the individuals converge on any other location on the Pareto-optimal curve, the adaptation of the weight vector will move them closer to the desired region.

Algorithm 1: R-MEAD

Inputs:

$size1$: the initial population size
$size2$: number of solutions to be generated close to the reference point
$radius$: determines the size of the solution region close to the reference point
m : number of objectives
n : number of dimensions
$nref$: number of provided reference points
$\mathcal{RF}$: A $nref \times m$ dimensional matrix of reference points with each row representing a single reference point.
dm : the decomposition method (weighted-sum or Tchebycheff)
$F(\vec{x})$: the objective function

Variables:

$\mathcal{IW}$: A $size1 \times m$ dimensional matrix of the initial weight vectors
$\mathcal{W}^i$: A $size2 \times m$ dimensional matrix of weight vectors for the i^{th} reference point.
$\mathcal{BW}$: A $nref \times m$ dimensional matrix of best weight vectors for reference points

1. $\mathcal{P} \leftarrow$ `rand(`*lbounds, ubounds, size1, n*`)`
2. $\mathcal{IW} \leftarrow$ `init_weight(`*size1*`)`
3. `evolve(`$\mathcal{P}$, $F(\vec{x})$, $\mathcal{IW}$, dm`)`
4. $\mathcal{PF} \leftarrow$ `evaluate(`$\mathcal{P}$, $F(\vec{x})$`)`
5. $step \leftarrow radius/size2$
6. **for** $i \leftarrow 1$ **to** $nref$ **do**
7. $\quad ind \leftarrow$ `min_ind(euclid_dist(`$\mathcal{PF}$, $\mathcal{RF}_{[i,:]}$`))`
8. $\quad \mathcal{BW}_{[i,:]} \leftarrow \mathcal{IW}_{[ind,:]}$
9. $\quad \mathcal{W}^i \leftarrow$ `init_weight(`$size2$, $radius$, $\mathcal{BW}_{[i,:]}$`)`
10. **end for**
11. $\mathcal{P} \leftarrow$ `rand(`*lbounds, ubounds,* $size2 \times nref$, n`)`
12. **while** stop criteria not met **do**
13. $\quad$ `evolve(`$\mathcal{P}$, $F(\vec{x})$`)`
14. $\quad \mathcal{PF} \leftarrow$ `evaluate(`$\mathcal{P}$, $\mathcal{W}$, dm`)`
15. $\quad$ **for** $j \leftarrow 1$ **to** $nref$ **do**
16. $\quad\quad \vec{dist} \leftarrow$ `euclid_dist(`$\mathcal{PF}$, $\mathcal{RF}_{[i,:]}$`)`
17. $\quad\quad best\_index \leftarrow$ `min_ind(`$\vec{dist}$`)`
18. $\quad\quad worst\_index \leftarrow$ `max_ind(`$\vec{dist}$`)`
19. $\quad\quad \vec{direction} \leftarrow \mathcal{W}^j_{[best\_index,:]} - \mathcal{W}^j_{[worst\_index,:]}$
20. $\quad\quad$ `update_weight(`$\mathcal{W}^j$, $direction$, $step$`)`
21. $\quad$ **end for**
22. **end while**

Algorithm 1 shows the details of this process. The major stages of the algorithm is outlined below:

Step 1 - Initialization: An initial population is randomly initialized within the lower and upper boundaries and evaluated for a limited number of iterations of an evolutionary algorithm with a predetermined decomposition method. Once the population is evolved, all of the objectives are evaluated and the results are stored in $\mathcal{PF}$ matrix (Algorithm 1 lines 1-4). It should be noted that each individual in the population is associated with only one weight vector which is later used to decompose the problem. The size of this initial population is determined by $size1$.

Step 2 - Finding the base weight vectors: At this stage for each of the reference points provided by the user, the closest point in the objective space ($\mathcal{PF}$) is found using Euclidean distance. The corresponding weight vector of the closest point to each reference point is chosen as the base weight vector around which a set of new weight vectors are generated. The number and the spread of weight vectors are determined by the $size2$ and $radius$ variables respectively (Algorithm 1 lines 6-10). Since the number of individuals and the weight vectors are equal, the variable $size2$ is practically the population size.

Step 3 - Evolving the population: Here the population is evolved and the weight vectors associated to each reference point are updated in a round-robin fashion until a termination criteria is met (Algorithm 1 lines 12-22).

Step 3.1 - Updating weights: At this stage the Euclidean distance of each reference point to its corresponding solutions in the population is calculated. In order to find the direction in which the weights should be updated, the best and the worst weights that correspond to the closest and the farthest points to the reference points are found. Next, an update direction is calculated based on the best and the worst weight vectors (Algorithm 1 line 19). Once the update direction is determined the weights are updated by moving them in the gradient direction with the amount determined by the $step$ variable (Algorithm 1 line 20). This process is repeated for all of the weight vectors associated with each reference point.

It should be noted that unlike MOEA/D our algorithm does not form a neighborhood for each of the individuals. Instead, the evolutionary optimizer relies on the entire population to create new offspring. This eliminates the need to calculate the Euclidean distances between all of the weight vectors to form the neighborhoods. Because we only need to find a limited set of solutions near the user-supplied reference point rather than forming the entire frontier, the population size that we use is comparatively smaller than MOEA/D which makes R-MEAD more computationally effective.

V. Experimental Results

In this section we report the performance of our algorithm on a set of benchmark problems including ZDT1-ZDT4 and ZDT6 from the ZDT test suite [24] for two-objective problems and DTLZ1-DTLZ2 functions from DTLZ test suite [25] for three-objective problems using the Tchebycheff and the weighted-sum approaches.

A. Parameter Settings

The initial population size ($size1$) is set to 100 and 250 for the two objective and three objective problems respectively and the algorithm is executed for 10 iterations in order to generate the base weight vectors (See Algorithm 1). In the second stage of the optimization the population size ($size2$) is set to 30 and 60 for two and three objective problems respectively. We consistently used 450 iterations for the two objective problems and 600 for the three objective problems. The radius is set to 0.02 for the weighted-sum and 0.05 for the Tchebycheff approaches. The performance of the algorithm is not sensitive to the radius value and its sole purpose is to allow the users to adjust the size of the regions they wish to cover near the reference points. Since the way weight vectors are used in the Tchebycheff and the weighted-sum approaches is different, we picked the radius value such that both approaches cover a region with relatively equal sizes. The optimizer that we used in our framework is a simple implementation of Differential Evolution [26].

B. Benchmark Results

For all of the benchmark problems we used two reference points in the infeasible and the feasible regions where one of them is closer to the Pareto-optimal front and the other one is farther away from it. On ZDT1, the reference pionts $(0.3, 0.4)$ and $(0.8, 0.2)$ were used. Figure 3(a) shows the promising result of our algorithm on this function. On ZDT2 the reference points $(0.8, 0.3)$ and $(0.5, 0.9)$ were used. Figure 3(b) shows that our algorithm managed to find a reasonable region on the Pareto-optimal front and close to the reference points. Our reference points for ZDT3 were $(0.15, 0.40)$ and $(0.75, -0.20)$. Figure 3(c) illustrates our results on this function. On ZDT4 $(0.2, 0.4)$ and $(0.5, 0.4)$ were our reference points. Figure 3(d) shows the results on ZDT4 where a set of solutions were found on the Pareto-optimal front close to the reference points. We chose $(0.9, 0.3)$ and $(0.5, 0.7)$ as reference points for ZDT6. Figure 3(e) shows that we get excellent results for this function.

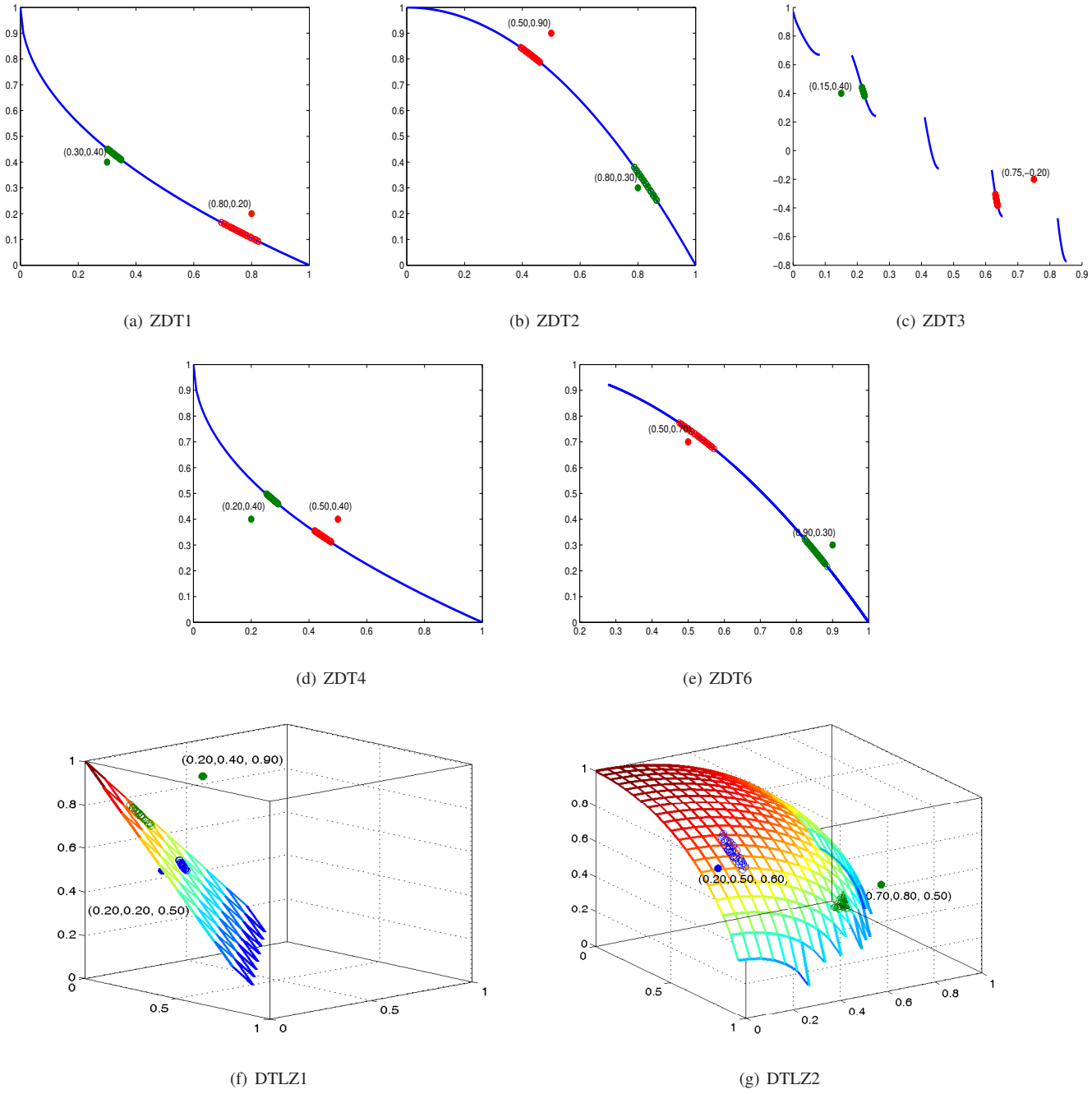
On DTLZ1 $(0.2, 0.4, 0.9)$ and $(0.2, 0.2, 0.5)$ are our reference points. We have chosen $(0.2, 0.5, 0.6)$ and $(0.7, 0.8, 0.5)$ as reference points for DTLZ2. These points were chosen in feasible and infeasible regions with various distances from the Pareto-optimal front. It can be seen from Figures 3(f) and 3(g) that the R-MEAD algorithm also managed to find suitable regions for the three objective problems. Overall, it can be seen from Figure 3 that the R-MEAD algorithm managed to find the desired regions as close as possible to the reference points on the Pareto-optimal front. The algorithm works consistently for the reference points in both feasible and infeasible regions.

C. Comparison between Weighted-Sum and Tchebycheff

In this section, instead of Tchebycheff we used weighted-sum as the decomposition method and run the algorithm with the same reference points. In all test functions we stop the algorithm after 450 generations. The results of the algorithm on test functions illustrate that weighted-sum approach does not work properly for non-convex Pareto-optimal fronts. Figure 4 shows the performance of the algorithm using the weighted-sum decomposition method on two objective problems. For ZDT2, ZDT3 and ZDT6 with non-convex Pareto-optimal fronts, the weighted-sum method could not find nicely distributed solutions close to the reference points on the Pareto-optimal front, as shown in Figures 4(b), 4(c) and 4(e). In contrast, on the functions with a convex Pareto-optimal front such as ZDT1 and ZDT4, the results are similar to that of the Tchebycheff method, as shown in Figures 4(a) and 4(d).

D. Faster Convergence to the Pareto-optimal front

In Section I, we mentioned that one of the difficulties faced by evolutionary multi-objective optimization algorithms in solving many-objective problems is the exponential increase in the number of solutions required to approximate the Pareto-optimal front as the number of objectives increases. It has been suggested that a reference point based approach allows a more focused search and eliminates the need to approximate the entire Pareto-front [17]. This approach can result in significant



(a) ZDT1 (b) ZDT2 (c) ZDT3

(d) ZDT4 (e) ZDT6

(f) DTLZ1 (g) DTLZ2

Fig. 3. Experimental results on ZDT1-ZDT4, ZDT6, DTLZ1 and DTLZ2 benchmark functions using Tchebycheff decomposition.

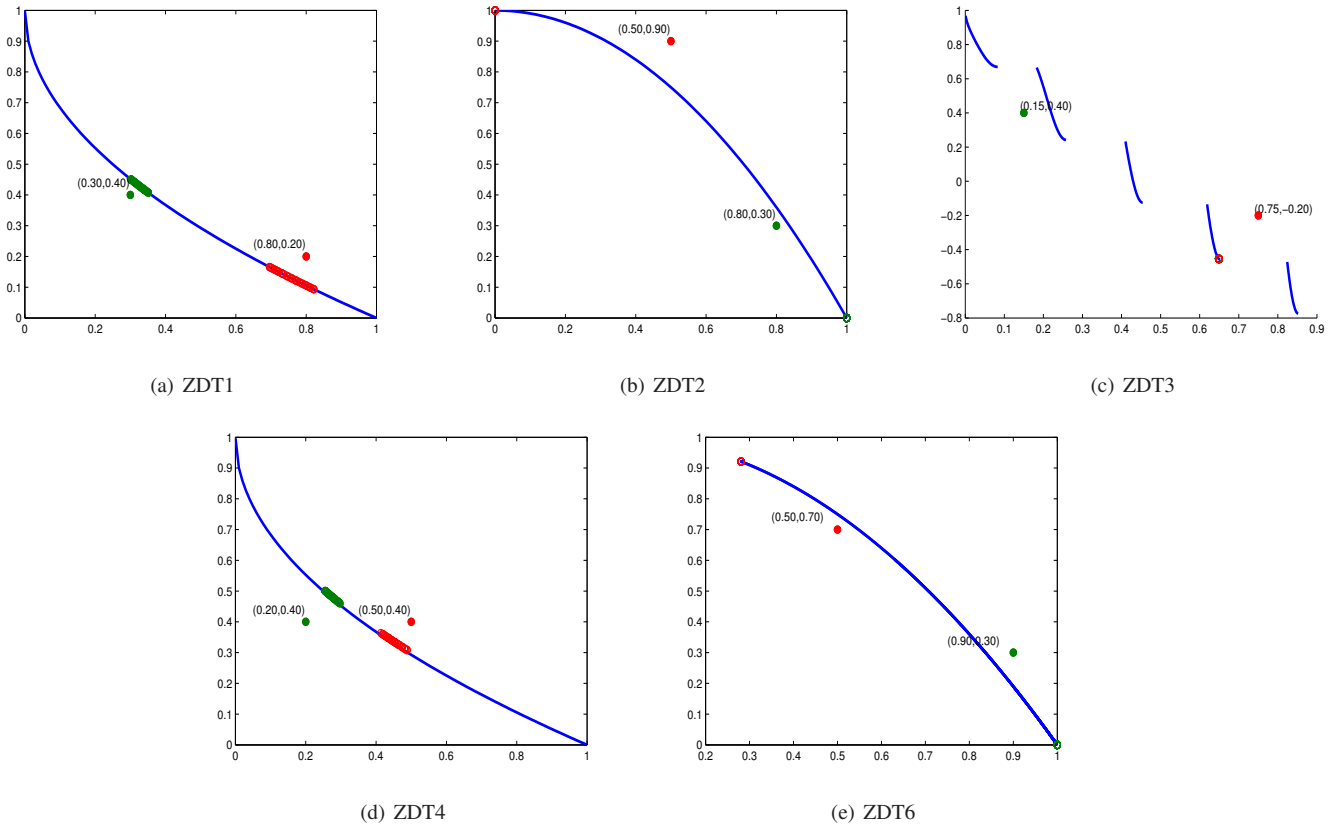
computational savings, especially in high dimensional objective spaces. In this section, we attempt to show this potential saving more quantitatively through experimentation.

We used Generational Distance (GD) [27] to measure the closeness of a solution front (PF_{sol}) to the Pareto-optimal front (PF_{true}) which is formulated as follows:

$$GD = \frac{(\sum_{i=1}^{n} d_i^p)^{\frac{1}{p}}}{n}, \tag{4}$$

where n is the number of solutions in PF_{sol} and d_i is the closest Euclidean distance from each point in PF_{sol} to a point in PF_{true}. We used a p value of 2 in our experiments.

Since we use Tchebycheff decomposition method in both R-MEAD and MOEA/D, they result in a similar spread of solutions in the regions they cover. This allows us to exclude the spread and focus solely on the convergence in our comparison. Additionally, in order to have a fair comparison, we consistently used the same value for the variable n in both R-MEAD and MOEA/D. Unlike dominance-based approaches where the first front is usually used to calculate the GD value, we set n to the population size in order to capture the



(a) ZDT1 (b) ZDT2 (c) ZDT3

(d) ZDT4 (e) ZDT6

Fig. 4. Experimental results on ZDT1-ZDT4 and ZDT6 benchmark functions using weighted-sum decomposition.

convergence behavior of the entire population.

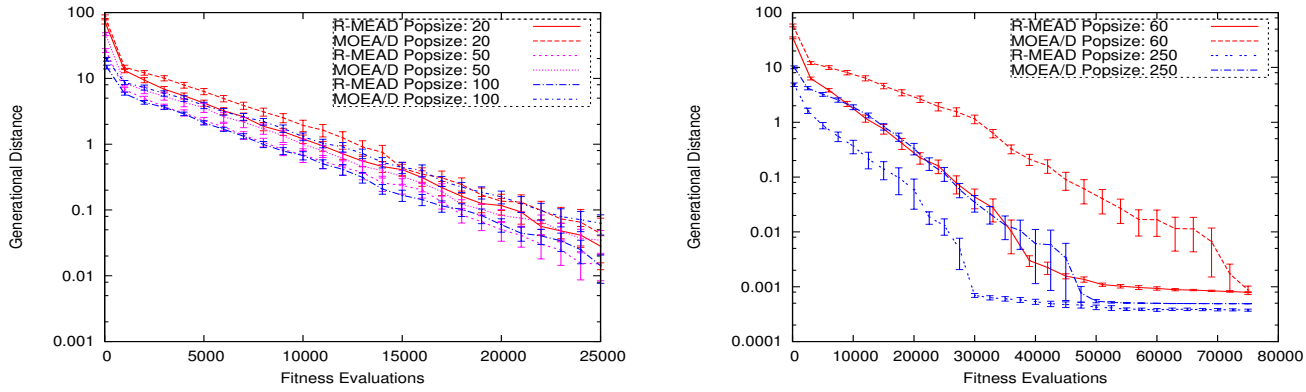
For our experiments we used a 2-objective and a 3-objective version of DTLZ1 function. In order to analyze the sensitivity of the algorithms to the population size, we used 20, 50 and 100 population sizes for 2-objective. For 3-objective we used 60 and 250 as the population sizes. The total number of fitness evaluations was set to 2.5×10^4 and 7.5×10^4 for 2-objective and 3-objective DTLZ1 respectively. Figure 5 shows how fast the GD value decreases through the course of evolution. Each point on the convergence plot is the average of 25 independent runs. It can be seen from Figure 5 that when an equal population size is used, R-MEAD consistently has a lower GD value than MOEA/D on both 2-objective and 3-objective DTLZ1. The fact that the error bars for the same population sizes do not overlap shows that R-MEAD algorithm converges significantly faster than MOEA/D. Figure 5(a), shows that the best results are achieved by R-MEAD with population sizes of 50 and 100 respectively.

On the 3-objective DTLZ1, Figure 5(b) shows that there is a more significant difference between the convergence speed of MOEA/D and R-MEAD as compared to that of the 2-objective DTLZ1. R-MEAD managed to obtain a similar convergence accuracy as compared to MOEA/D with significantly less number of fitness evaluations from around 2×10^4 to 3.5×10^4 depending on the population size. The increased 'gap' in the convergence speed from a 2-objective problem to a 3-objective problem suggests that a considerable amount of computational resources can be saved using a reference point based approach, especially when dealing with many-objective problems. It should be noted that although in the case of R-MEAD we have not taken the closeness to the reference point into account, the experiments that we conducted in Section V-B have shown that a region close to the reference point can be formed using around 3.6×10^4 fitness evaluations which is far less than 7.5×10^4 that we used in this section. By looking at Figure 5(b) we can see that R-MEAD has a much lower GD value around 3.6×10^4 which shows that not only it converged faster than MOEA/D but also managed to find a suitable region close to the reference point.

VI. Conclusion

In this paper, we have proposed a preference-based evolutionary multi-objective optimization through decomposition. The proposed approach has the potential to be applied to many-objective problems for the following major reasons:

- Use of decomposition techniques converts a multi-objective problem into single-objective which is not susceptible to the selection pressure problem common in dominance-based approaches.
- The use of a reference point allows the search to be focused on the desired regions which can potentially save considerable amount of computational resources.



(a) 2-objective DTLZ1

(b) 3-objective DTLZ1

Fig. 5. Convergence of Generational Distance measure for R-MEAD and MOEA/D on 2-objective and 3-objective DTLZ1.

The algorithm has been evaluated using two decomposition approaches, namely the weighted-sum and the Tchebycheff. The Tchebycheff approach was superior on most of the test problems especially when dealing with non-convex problems. In all of the benchmark problems, the proposed algorithm managed to find a set of solutions close to each of the provided reference points on the Pareto-optimal front. It has also been shown that R-MEAD results in a faster convergence as compared to MOEA/D especially when the number of objectives increases.

In this paper we have presented a very simple way of updating the weight vectors. In our future work, we intend to investigate the effectiveness of adapting the weight vectors when dealing with many-objective problems with complex non-convex Pareto-optimal fronts.

REFERENCES

[1] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithm*. Wiley, 2008.

[2] C. A. C. Coello and G. B. Lamont, Eds., *Applications Of Multi-Objective Evolutionary Algorithms*. World Scientific, 2004.

[3] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[4] F. di Pierro, *Many-Objective Optimization and Applications to Water Engineering: Theory and Practice*. LAP Lambert Academic Publishing, April 2010.

[5] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proc of 2008 IEEE Congress on Evolutionary Computation*, 2008, pp. 2419–2426.

[6] ——, "Evolutionary many-objective optimization," in *Proc. of 3rd International Workshop on Genetic and Evolving Fuzzy Systems*, no. March, 2008, pp. 47–52.

[7] K. Deb, J. Sundar, U. B. R. N, and S. Chaudhuri, "Reference point based multi-objective optimization using evolutionary algorithm," *International Journal of Computational Intelligence Research*, pp. 273–286, 2006.

[8] K. Deb, a. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, pp. 182–197, 2002.

[9] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[10] T. Murata and H. Ishibuchi, "Moga: multi-objective genetic algorithms," in *Evolutionary Computation, 1995., IEEE International Conference on*, vol. 1, nov-1 dec 1995, p. 289.

[11] E. J. Hughes, "Evolutionary many-objective optimisation: many once or one many?" in *Proc. of 2005 IEEE Congress on Evolutionary Computation*, 2005, pp. 222–227.

[12] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA: Kluwer, 1999.

[13] 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)*, pp. 392–403, 1998.

[14] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions*, pp. 712–731, 2007.

[15] C. Coello, "Handling preferences in evolutionary multiobjective optimization: a survey," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, 2000, pp. 30–37.

[16] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," *Multiple Criteria Decision Making Theory and Applications*, pp. 468–486, 1980.

[17] W. U.K. and X. Li, "Integrating user preferences with particle swarms for multi-objective optimization," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008, pp. 745–752.

[18] K. Deb, "Solving goal programming problems using multi-objective genetic algorithms," in *Proceedings of the 1999 Congress on Evolutionary Computation*, 1998.

[19] J. P. Ignizio, *Goal Programming and Extensions*. Lexington Books, 1976.

[20] D. Cvetkovic and I. Parmee, "Evolutionary design and multi-objective optimisation," in *In proc. of 6th European Congress on Intelligent Techniques and Soft Computing EUFIT'98*, 1998, pp. 397–401.

[21] Y. Jin and B. Sendhoff, "Incorporation of fuzzy preferences into evolutionary multiobjective optimization," in *4th Asia-Pacific Conference in Simulated Evolution and Learning*, 2002, pp. 26–30.

[22] J. Branke, T. Kaubler, and H. Schmeck, "Guidance in evolutionary multi-objective optimization," *Advances in Engineering Software*, pp. 499–507, 2001.

[23] K. Deb, "Multi-objective evolutionary algorithms : Introducing bias among pareto-optimal solutions multi-objective optimization," *Optimization*, pp. 263–292, 2003.

[24] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results." *Evolutionary computation*, pp. 173–95, 2000.

[25] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," Kanpur Genetic Algorithms Laboratory, Tech. Rep., 2001.

[26] R. Storn and K. Price, "Differential evolution . a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization 11 (4)*, pp. 341–359, 1995.

[27] D. V. Veldhuizen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Ph.D. dissertation, OH: Air Force Institute of Technology, Dayton, 1999, Technical Report No. AFIT/DS/ENG/99-01.