A Two-phase Many-Objective Evolutionary Algorithm with Penalty Based Adjustment for Reference Lines

Chunyang Zhu
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: zhuchunyangfun@163.com

Xinye Cai
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: xinye@nuaa.edu.cn

Zhun Fan
School of Engineering
Shantou University
Guangdong, China
Email: zfun@stu.edu.cn

Muhammad Sulaman
School of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Jiangsu, China
Email: sulman0909@gmail.com

***Abstract*—In this paper, we proposed a two-phase many-objective evolutionary algorithm to tackle many objective optimization problems. In the first phase, the algorithm focuses on achieving good convergence towards the boundary Pareto optimal solutions. In the second phase, it maintains a good balance between convergence and diversity by using a set of widely spread reference lines. In addition, a penalty based adjustment for reference line has been adopted to handle many objective optimization problems with incomplete PFs. The performance of our proposed algorithm is validated and compared with four state-of-the-art many objective evolutionary algorithms on DTLZ problems. The results show that our proposed algorithm is very competitive with other compared algorithms.**

I. INTRODUCTION

A *multiobjective optimization problem* (MOP) can be stated as follows:

$$\begin{aligned} \text{maximize} \quad & F(x) = (f_1(x), \ldots, f_m(x)) \\ \text{subject to} \quad & x \in \Omega \end{aligned} \tag{1}$$

where Ω is the *decision space*, $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions. The *attainable objective set* is $\{F(x)|x \in \Omega\}$. In the case when Ω is a finite set, (1) is called a discrete MOP.

Let $u, v \in R^m$, u is said to *dominate* v, denoted by $u \succ v$, if and only if $u_i \geq v_i$ for every $i \in \{1, \ldots, m\}$ and $u_j > v_j$ for at least one index $j \in \{1, \ldots, m\}$[1]. Given a set S in R^m, a point in it is called non-dominated if no other point in S can dominate it. A point $x^* \in \Omega$ is *Pareto-optimal* if $F(x^*)$ is non-dominated in the attainable objective set. $F(x^*)$ is then called a *Pareto-optimal (objective) vector*. In other words, any improvement in one objective of a Pareto optimal point must lead to deterioration to at least another objective. The set of all the Pareto-optimal points is called the *Pareto set* (*PS*) and the set of all the Pareto-optimal objective vectors is the *Pareto front* (*PF*) [1]. The ideal point Z^* is defined as $Z^* = (Z_1^*, Z_2^*, \ldots, Z_M^*)^T$,where $Z_i^* = \min_{x \in \Omega} f_i(x), i \in \{i \ldots M\}$. The nadir point B is defined as $B = (B_1, B_2, \ldots, B_M)^T$,where $B_i = \max_{x \in PS} f_i(x), i \in \{i \ldots M\}$.

[1]In the case of minimization, the inequality signs should be reversed.

Due to the population-based nature, evolutionary multiobjective optimization (EMO) methodologies have shown their superiority in approximating PFs in one run. Up to date, many multiobjective evolutionary algorithms (MOEAs) have satisfactory performances on MOPs with two or three objectives [2]–[6]. However, for many objective optimization problems (MaOPs) that have more than three objectives, traditional MOEAs encounter difficulties due to the following reasons.

1) MaOPs with an increasing number of objectives cause Pareto-based MOEAs to lose the selection pressure during the evolutionary process because almost all the solutions in a population become non-dominated with each other. It has been shown that the performance of the Pareto-based algorithms such as NSGA-II [4] deteriorate significantly on MaOPs.
2) Some evolutionary operators, such as crossover and mutation, become inefficient as the conflicts between convergence and diversity in the objective space become more severely with the increase of the objectives. Therefore, balancing the diversity and convergence at the same time has become a more difficult task in the high dimensional objective space.
3) In order to control the computational complexity, the population size of MOEAs cannot be very large; however, a small-sized population cannot make a good approximation to the whole PFs, especially when the the number of objectives in a MaOP is very high.

In recent years, a number of many-objective evolutionary algorithms (MaOEAs) have been designed to address the above challenges. For example, it is very straightforward to modify the Pareto dominance to enhance the selection pressure for

Pareto-based MaOEAs to tackle MaOPs [7] [8] [9]. The other class of MaOEAs is aggregation-based approaches (e.g. [10]–[14]), which use aggregation methods, such as Tchebycheff (TCH), weighted sum (WS) or penalty boundary intersection (PBI) to solve MaOP. The hybrid algorithms based on both dominance and decomposition(e.g. [15]–[19]) have also been designed for MaOPs. For example, the state-of-art NSGA-III [20], which can be considered as an extensive version of the NSGA-II, use Pareto dominance to maintain convergence while employing a set of uniformly distributed reference lines to maintain diversity. A hybridized evolutionary many-objective evolutionary algorithm, MOEA/DD [18], combines dominance- and decomposition- based approaches to balance the convergence and diversity for the evolutionary process.

More recently, a two-stage MaOEAs, called MOEA-R&D [21], implicitly divides the evolutionary process into the following two stages. At the first stage, an achievement scalarizing function (ASF) is used to lead the whole population to approximating the boundary Pareto optimal solutions for PFs. At the second stage, MOEA-R&D uses the approximated Nadir point to conduct space reduction for the whole population; Meanwhile, it maintains the diversity of the population by using a distance-based density metric. Generally, MOEA-R&D works well on MaOPs, such as DTLZ benchmark problems [22]. Nevertheless, it has the following limitations. First, ASF does not perform very well on approximating the boundary optimal solutions with special PFs, e.g., incomplete PFs. Second, it allocates too much resource to the first stage that causes inefficiency of the algorithm. Last but not least, when most solutions are already located inside the reduced objective space, which leads to the lack of the selection pressure in the second stage. Based on the above considerations, in this paper, we propose a novel two-phase MaOEA, called TPEA-PBA, to tackle MaOPs. TPEA-PBA explicitly divides the evolutionary status into two phases. At the first phase, local Pareto dominance is applied to approximate the boundary Pareto optimal solutions. At the second phase, a set of widely spread reference lines is maintained to balance the convergence and diversity at the same time. To address the MaOPs with incomplete and disconnected PFs , a penalty vector is adopted to evaluate the effectiveness of every reference line and adjust them accordingly.

The rest of this paper is organized as follows. Section II is devoted to the descriptions of our proposed algorithm for many-objective optimization. Experimental settings and comprehensive experiments are conducted in Section III. Section IV concludes this paper and the future work is also given in this section.

II. ALGORITHM

A. The Framework

Algorithm 1 presents the general framework of TPEA-PBA. As our algorithm divides the evolutionary process into two phases, two sets of reference line, γ for Phase One and λ for Phase Two, are included as the inputs of the algorithm. The output of the algorithm is the population $P = \{x^1, \dots, x^N\}$. In the first phase, boundary Pareto optimal solutions $(BS^1, \dots, BS^M)$ are approximated by the local nondominated solutions that have the maximal value in each of M objectives. The nadir point $B = (B_1, B_2, \cdots, B_M)$ can be approximated as

$$B_i = \max_{x \in BSs} f_i(x), i \in \{1, ...M\} \tag{2}$$

In the initialization procedure, the population P is initialized by randomly sampling the search space. Consequently, M boundary solutions are initialized as the solutions that have the maximal value in each of M objectives.

To determine the phase of the evolutionary status at the j-th generation, Δ, the maximal relative decrease of the nadir point vector over the last $inter = 200$ generations, is defined as follows.

$$\Delta = \max_{i \in \{1,...M\}} \frac{B_i^{\,j} - B_i^{\,j-inter}}{B_i^{\,j-inter}} \tag{3}$$

If Δ is larger than a preset value (0.001), the evolution is considered to be in Phase One; otherwise, it is considered to be in Phase Two. TPEA-PBA uses different selection strategies for different phases. After the selection and the update of the boundary optimal solutions, penalty vector Ref is updated to determine the effectiveness of every reference. More details of each component of the algorithm are explained in the following sections.

B. Reproduction

In reproduction, N offspring solutions are generated by applying variation operators to the population P. Two scenarios may exist based on the different phases of the algorithm. In Phase One, only the boundary solution and its nearest neighbor are selected for mating. Differential evolution (DE) is applied for N times so a population Q with a total number of N offspring is generated. In Phase Two, each solution is associated with a subproblem specified by a reference line. For each subproblem, the associated solution and one of its Ne nearest neighbors are selected for mating. DE is also applied to generate a solution for each subproblem thus a population Q with a total number of N offspring is generated.

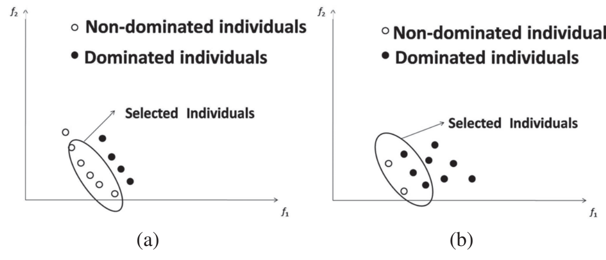


Fig. 1: (a) and (b) represent the correspond the selection process in Phase 1, where circle points (○) represent non-dominated individuals and blank circle points (●) represent dominated individuals.

Algorithm 1: TPEA-PBA

```
Input:
    MOP(1);
    a stopping generation; $maxgen$
    $N$: the population size;
    $M$: the number of objectives;
    a uniform spread of $M$ reference lines: $\gamma = \{\gamma^1 \dots, \gamma^M\}$;
    /*$\gamma^i = (0, \dots, 1, 0, \dots, 0)^T$, where the $i-$th element of $\gamma^i$ is 1 and other elements of $\gamma^i$ is 0*/
    a uniform spread of $N$ reference lines: $\lambda = \{\lambda^1 \dots, \lambda^N\}$;
    /*Developed from Das and Denniss method [23]*/
Output:
    $P$: the solution set ;
1:  Initialize a population $P = \{x^1, \dots, x^N\}$;
2:  $BS^i$ = $\text{argmax}_{x \in P} f_i(x), i \in \{1, \dots, M\}$;
3:  $B_i = \max_{x \in BSs} f_i(x), i \in \{1, ...M\}$;
4:  For each $Ref_i, i \in \{1, \dots, N\}, Ref_i = 0$;
5:  $phase = 1$; /*Phase One is default phase*/
6:  for $gen = 1; gen \leqslant maxgen; gen++$ do
7:    Use DE to generate a offspring population Q;
      /*Reproduction */
8:    if $\Delta \geqslant 0.001$ $and$ $phase == 1$ then
9:      $[P]$ = PhaseOneSelect($P \bigcup Q, \gamma$);
        /*Phase One Selection */
10:   else
11:     $phase = 2$;
12:     $[P\ ,T]$ = PhaseTwoSelect($P \bigcup Q, \lambda, B$);
        /*Phase Two Selection */
13:   end if
14:   $[BSs, Ref]$ = UpdateBSs&Ref($P, T, \gamma, \lambda, BSs, Ref, phase$);
      /*Update Boundary solutions and Penalty vector*/
15:   if $maxgen - gen == 50$ then
16:     $[\lambda]$ = UpdateRef($Ref, \lambda, gen$);
17:   end if
18: end for
```

Algorithm 2: Find-nondominated-solution(P)

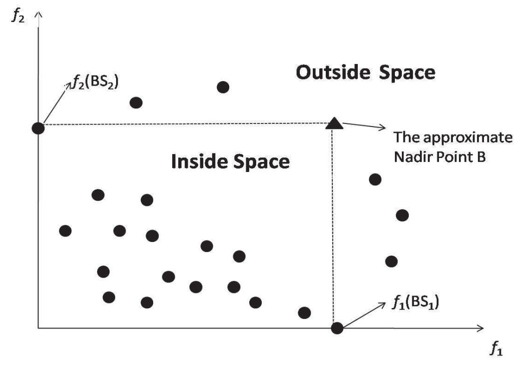
```
Input:
    a set of population: $P$;
Output:
    $\widehat{P}$ : the nondominated solutions;
    $\widetilde{P}$ : the dominated solutions;
1: $\widehat{P} = \phi; \widetilde{P} = \phi$;
2: for each $i \in P$ do
3:   if $\exists j \in P, j \succ i$ then
4:     $\widetilde{P} = \widetilde{P} \bigcup i$;
5:   else
6:     $\widehat{P} = \widehat{P} \bigcup i$;
7:   end if
8: end for
```

Algorithm 3: PhaseOneSelect(P, γ)

```
Input:
    $P$: the merge population;
    a uniform spread of $M$ reference lines: $\gamma = \{\gamma^1 \dots, \gamma^M\}$;
Output:
    $\bar{P}$: the elite population;
1:  $P$ is equally divided into $M$ subpopulations by $\gamma$. Each subpopulation $Subp^i$ contains $2 * N/M$ solutions that are close to the $i$-th reference line;
2:  $\bar{P} = \phi$;
3:  for each $Subp^i, i \in \{1, \dots, M\}$ do
4:    $j = 1$;
5:    $[\widehat{P}, \widetilde{P}] = Find-nondominated-solution(Subp^i)$;
6:    if $|\widehat{P}| > N/M$ then
7:      while $j \leqslant N/M$ do
8:        $\bar{P} = \bar{P} \bigcup \text{argmin}_{x \in \widehat{P}}\ d^{\perp}(F(x), \gamma^i)$;
          /*$d^{\perp}$ :perpendicular distance between $F(x)$ and $\gamma^i$*/
9:        $\widehat{P} = \widehat{P} \setminus \text{argmin}_{x \in \widehat{P}}\ d^{\perp}(F(x), \gamma^i)$;
10:       $j = j++$;
11:     end while
12:   else if $|\widehat{P}| == N/M$ then
13:     $\bar{P} = \bar{P} \bigcup \widehat{P}$;
14:   else if $|\widehat{P}| < N/M$ then
15:     $\bar{P} = \bar{P} \bigcup \widehat{P}$;
16:     while $j \leqslant N/M - |\widehat{P}|$ do
17:       $\bar{P} = \bar{P} \bigcup \text{argmin}_{x \in \widetilde{P}} f_i(x)$;
18:       $\widetilde{P} = \widetilde{P} \setminus \text{argmin}_{x \in \widetilde{P}} f_i(x)$;
19:       $j = j++$
20:     end while
21:   end if
22: end for
```

C. Selection for Phase One

The pseudo-code of the selection for Phase One is described in Algorithm 3. In Phase One, the merged population P with $2 \times N$ solutions are equally divided into M subpopulations based on Euclidean distance. Each subpopulation $Subp^i$ contains $2 \times N/M$ solutions that are close to the i-th reference line. For each subpopulation $Subp^i, i \in \{1, ...M\}$, nondominated solutions $\widehat{P}$ and dominated solutions $\widetilde{P}$ are separated by $Find-nondominated-solution$. If the size of $\widehat{P}$ is larger than N/M, the ones closest to the reference line i in $\widehat{P}$ are selected, as shown in Fig. 1(a). If the size of $\widehat{P}$ is less than N/M, the rest solutions are selected from $\widetilde{P}$, which have the smallest values regarding i-th objective, as show in Fig. 1(b). After selection, a new population $\bar{P}$ with N solutions are returned as output for Algorithm 3.



(a)

Fig. 2: The space reduction procedure in a 2-objective case.

D. Selection for Phase Two

The pesudo-code of the selection procedure for Phase Two is described in Algorithm 5.

In line 1, P is divided into N subpopulations. Each subpopulation $Subp^i$ is a solution set that are closest to the i-th reference line. Note that some subpopulation contains more than one solution while others may contain no solutions. The index of empty $Subp^i$ is stored as T. In other words, T stores the index of reference line which contains no solution.

In line 2, space reduction is conducted to delete solutions outside of the boundaries specified by the approximation of Nadir point B in the objective space. The solution set in P within the boundaries is noted by $\grave{P}$ while the one outside the boundaries is noted $\ddot{P}$. Fig. 2. shows the space reduction in the case of 2-objective optimization. The pesudo-code of space reduction is presented in Algorithm 4.

Line 3-15 shows the core selection procedures after space reduction. If the size of $\grave{P}$ is less than N, $\grave{P}$ is assigned to output $\bar{P}$ first and the rest $N - |\grave{P}|$ solutions are filled by the ones that have the shortest distance to the ideal point(Z^*) from $\ddot{p}$ (line 4-9). Otherwise, N solutions are selected from $\grave{P}$ as follows. For each non-empty $Subp^i$, only one solution that has the shortest distance to the approximation of the ideal point Z^* is kept for itself (line 11-16). For each empty $Subp^i$, one solution in the $\grave{P} \setminus \bar{P}$ that has the shortest perpendicular distance to the i-th reference line is put in it (line 11-22). After that, every $Subp^i$ is combined as the output (line 23).

E. Updating the Boundary Solutions and Penalty Vector

In the selection of Phase Two, the approximation of boundary optimal solutions, BSs, are used to conduct space reduction. In addition, the penalty vector, Ref, is used to update reference lines. The update of both BSs and Ref are presented in this section, detailed in Algorithm 6.

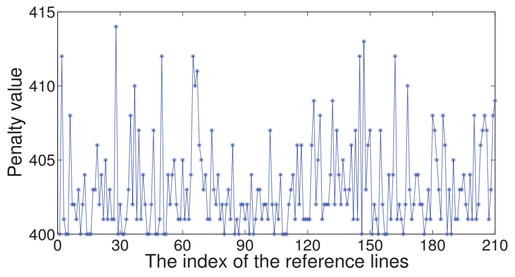
In line 1, P is equally divided into M subpopulations by reference line γ. Each subpopulation $Subp^i$ contains $2*N/M$ solutions that are close to the i-th reference line. For each $Subp^i$, the nondominated solution with the best i-th objective value is updated as the boundary solution for the i-th objective, BS^i (line 3-7).

Algorithm 4: SpaceReduction(P, B)

```
Input:
    a set of population: $P$;
    $B$: the nadir point;
Output:
    $\grave{P}$ : inside space solutions;
    $\ddot{P}$ : outside space solutions;
1: $\grave{P} = \phi; \ddot{P} = \phi$;
2: for each $P^i \in P, i \in \{1, \dots, |P|\}$ do
3:     if $f_j(P^i) \leqslant B_j, j \in \{1 \dots M\}$ then
4:         $\grave{P} = \grave{P} \bigcup P^i$;
5:     else
6:         $\ddot{P} = \ddot{P} \bigcup P^i$;
7:     end if
8: end for
```

Algorithm 5: PhaseTwoSelect(P, λ, B)

```
Input:
    $P$ : the merged population;
    a uniform spread of $N$ reference lines: $\lambda = \{\lambda^1 \dots, \lambda^N\}$;
    $B$: the nadir point;
Output:
    $\bar{P}$ : the elite population;
    $T$ : a set of empty reference lines which contains no solution;
1: $P$ is divided into $N$ subpopulations. Each subpopulation $Subp^i$ is a solution set that are closest to the $i$-th reference line. The index of empty $Subp^i$ is stored as $T$;
2: $[\grave{P}, \ddot{P}] = SpaceReduction(P, B)$;
3: $\bar{P} = \phi$;
4: if $|\grave{P}| < N$ then
5:     $\bar{P} = \grave{P}$;
6:     while $|\bar{P}| \leqslant N$ do
7:         $\bar{P} = \bar{P} \bigcup$ argmin$_{x \in \ddot{P}}$ $d(F(x), Z^*)$; /* $d$: distance between $F(x)$ and $Z^*$*/
8:         $\ddot{P} = \ddot{P} \setminus$ argmin$_{x \in \ddot{P}}$ $d(F(x), Z^*)$;
9:     end while
10: else
11:     for each $Subp^i, i \in \{1, \dots, N\}$ do
12:         $Subp^i = Subp^i \setminus \ddot{P}$;
13:         if $Subp^i$ $! = \phi$ then
14:             $Subp^i$ =argmin$_{x \in subp^i} d(F(x), Z^*)$;
15:         end if
16:     end for
17:     $\bar{P} = Subp^1 \bigcup \cdots Subp^N$;
18:     for each $Subp^i, i \in \{1, \dots, N\}$ do
19:         if $Subp^i$ $== \phi$ then
20:             $Subp^i$ =argmin$_{x \in \{\grave{P} \setminus \bar{P}\}} d^{\perp}(F(x), \lambda^i)$;
21:         end if
22:     end for
23:     $\bar{P} = Subp^1 \bigcup \cdots Subp^N$;
24: end if
```


(a) DTLZ2

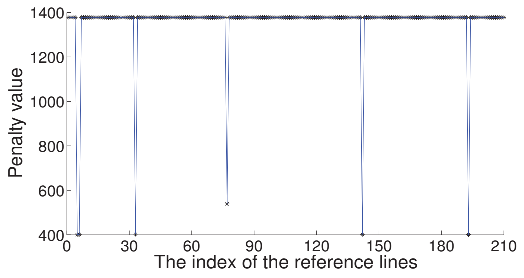

(b) DTLZ6

Fig. 3: The penalty value for each Reference line on DTLZ2(a) and DTLZ6(b) on 5-objectives.

The update of Ref is as follows. For j-th reference line, if it is empty ($\lambda^j \in T$) or it is a locally dominated solution ($P^j \in \delta$), the value of Ref_j is updated by 1 (line 15). An example of the penalty vector Ref is given in Fig. 3.

F. Adjustment of the Reference Lines

As the shapes of PFs are not known in advance, in Algorithem 8, the adjustment of reference lines is conducted online based on penalty vector Ref obtained in Algorithm 6, as follows.

First, the reference lines with the highest penalty values are deleted (line 1-4). The new reference lines are generated by $Variation$ procedure, described in Algorithm 7. A Roulette wheel selection is used to select "good" reference lines to generate new reference lines. The lower penalty value of a reference line, the better it is considered and the higher probability it will be selected to generate new reference line. The selection probability $prob_i$ for reference line $\lambda^i, i \in \{1, 2, \ldots, |\lambda|\}$ is defined as follows.

$$prob_i = \frac{\max\limits_{k\in 1,\ldots|\lambda|}(Ref_k) - Ref_i}{N \times \max\limits_{k\in 1,\ldots|\lambda|}(Ref_k) - \sum\limits_{k=1}^{|\lambda|} Ref_k} \quad (4)$$

As shown in Algorithm 7, in line 1, $sumprob$ is cumulative probability for $prob$ and calculated as follows.

$$sumprob_j = \begin{cases} 0 & j = 1 \\ \sum\limits_{i=1}^{j-1} prob_i & j \in \{2, ..., |\lambda| + 1\} \end{cases}$$

Algorithm 6: UpdateBSs&Ref(P,T, γ,λ, $BSs, Ref, phase$)

```
Input:
    $P$ : the elite population;
    $T$ : a set of empty reference lines which contains no solution;
    a uniform spread of $M$ reference lines: $\gamma = \{\gamma^1 \ldots \gamma^M\}$;
    a uniform spread of $N$ reference lines: $\lambda = \{\lambda^1 \ldots, \lambda^N\}$;
    $BSs$: the approximation of boundary optimal solutions;
    $Ref$: the penalty vector;
    $phase$: the phase of the evolutionary status;
Output: $BSs$;
        $Ref$;
1:  $P$ is equally divided into $M$ subpopulations by reference line $\gamma$. Each subpopulation $Subp^i$ contains $2 * N/M$ solutions that are close to the $i$-th reference line;
2:  $\delta = \phi$; /*$\delta$ is a set which contains dominated solution*/
3:  for each $Subp_i, i \in \{1, 2, \ldots, M\}$ do
4:      $[\widehat{P}, \widetilde{P}]$ = Find-nondominated-solution($Subp_i$);
5:      $\delta = \delta \bigcup \widetilde{P}$;
6:      $BS^i$ = argmax$_{x\in\widehat{P}} f_i(x)$;
7:  end for
8:  if $phase == 1$ then
9:      for $j \in \{1, \ldots, N\}$ do
10:         $Ref_j = Ref_j + +$;
11:     end for
12: else
13:     for $j \in \{1, \ldots, N\}$ do
14:         if $\lambda^j \in T \,||\, P^j \in \delta$ then
15:             $Ref_j = Ref_j + +$;
16:         end if
17:     end for
18: end if
```

Algorithm 7: Variation(λ, Ref)

```
Input:
    a uniform spread of $N$ reference lines: $\lambda = \{\lambda^1 \ldots, \lambda^N\}$;
    $Ref$ : a penalty vector for $\lambda$;
Output:
    $\lambda^j$: a new reference line;
1:  Calculating the accumulative probability $sumprob$;
2:  $i = Random(M)$; /*Randomly select a dimension*/
3:  if $rand \in \{sumprob(j), sumprob(j+1)\}, j \in \{1, \ldots, |\lambda|\}$ then
4:      ${\lambda^j}_i = {\lambda^j}_i + 0.001 * (2 * rand(1) - 1)$; /*Disturbance of selected reference line*/
5:  end if
```

Algorithm 8: UpdateRef(Ref, λ, gen)

```
Input:
    $Ref$ : a penalty vector;
    a uniform spread of $N$ reference lines: $\lambda = \{\lambda^1 \ldots, \lambda^N\}$;
    $gen$ : the current generation number;
Output:
    $\lambda$ : a set of reference lines;
1: for $i = 1; i <= N; i++$ do
2:     if $Ref_i == gen$ then
3:         $\lambda = \lambda \setminus \{\lambda_i\}$;
4:     end if
5: end for
6: while $|\lambda| < N$ do
7:     $\lambda = \lambda \bigcup Variation(\lambda, Ref)$;
8: end while
```

III. EXPERIMENTS

A. Parameter Settings

In our experiments, DTLZ benchmark suite is used to verify TPEA-PBA. The number of decision variables is set to $n = m + k - 1$ for all the DTLZ test problems, where k is set to 5 for DTLZ1 and 10 for DTLZ2, DTLZ3, DTLZ4, DTLZ5 and DTLZ6. k is set to 20 for DTLZ7 as suggested in [18]. TPEA-PBA is compared with MOEAD/DE, GrEA, NSGA-III and MOEA/DD. The number of function evaluations is set to 300000 for all the compared algorithms. The population size for 4-objective test problems is set to 220 for all compared algorithms. For 5-objective test problems, population size is set to 210 except GrEA and for GrEA, population size is set to 212 according to the original paper. Each algorithm is run 30 times independently for each test instance. Other parameter settings for compared algorithms can be referred to [18], [20], [24], [25]. The other parameter settings for TPEA-PBA are summarized as follows.

1) For the DE operator, we set CR = 1.0 and F = 0.5.
2) Neighborhood size: Ne = 10.

B. Performance Metric

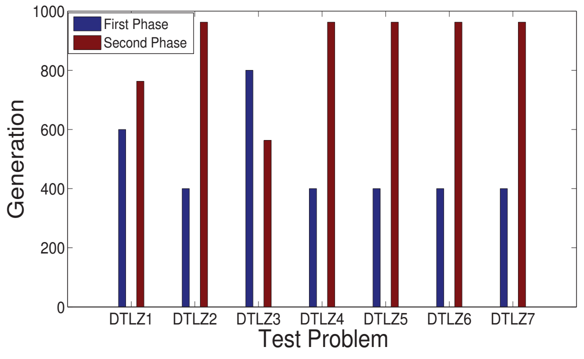
The following performance metric is used in our studies: **Inverted Generational Distance** (IGD) [26]: It measures the average distance from a set of reference points P^* in the PF to the approximation set P. It can be formulated as follows.

$$IGD(P, P^*) = \frac{1}{|P^*|} \sum_{v \in P^*} dist(v, P) \tag{5}$$

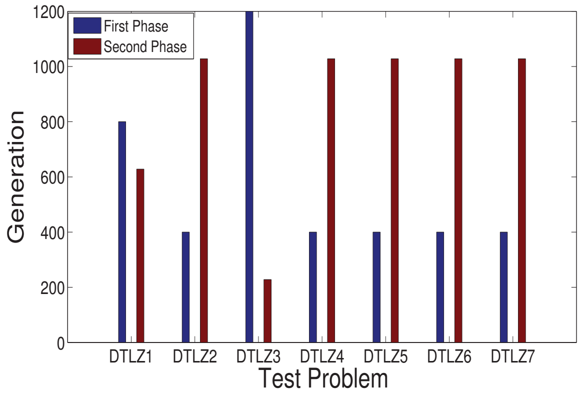
where $dist(v, P)$ is the Euclidean distance between the solution v and its nearest point in P, and $|P^*|$ is the cardinality of P^*. If $|P^*|$ is large enough to represent the PF very well, $IGD(P, P^*)$ could measure both the diversity and convergence of P in a sense.

C. Experimental Results

In this section, experimental results of all the compared algorithms in terms of IGD are presented in table I. From the table, it can be observed that TPEA-PBA obtains significantly better performance than all the other compared algorithms on all 5-objective optimization problems except for DTLZ7. In DTLZ7, TPEA-PBA is also the best algorithm, although its performance is not significantly better than that of NSGA-III. For 4-objective optimization problems, TPEA-PBA obtains the best performance on all the test problems except for DTLZ5 and DTLZ7. For DTLZ5, the performance of TPEA-PBA is significantly better than that of MOEA/DD, GrEA and NSGA-III but worse than that of MOEA/D-DE. For DTLZ7, the performance of TPEA-PBA is significantly better than that of MOEA/D-DE, GrEA and MOEA/DD but worse than that of NSGA-III. In general, TPEA-PBA achieves the best performance among all the compared algorithms.



(a)



(b)

Fig. 4: Two-phase distribution of computing resources for 4-objective DTLZ (a) and 5-objective DTLZ (b) problems.

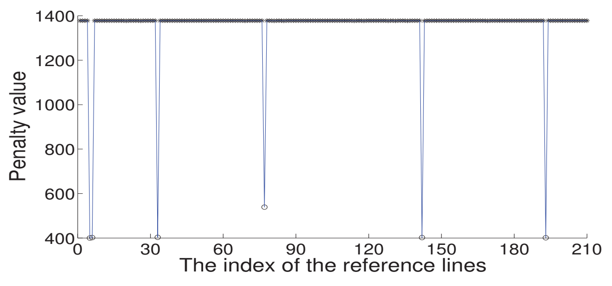
D. More Discussions on the Two-Phase Mechanism

To investigate the two-phase mechanism in TPEA-PBA, the allocations of computational resource for each phase on different benchmark problems are shown in Fig. 4. It can be observed that the computational proportions of Phase One (convergence phase) for DTLZ1 and DTLZ3 are higher than other benchmark problems for TPEA-PBA, due to the facts that DTLZ1 and DTLZ3 are more difficult to converge than

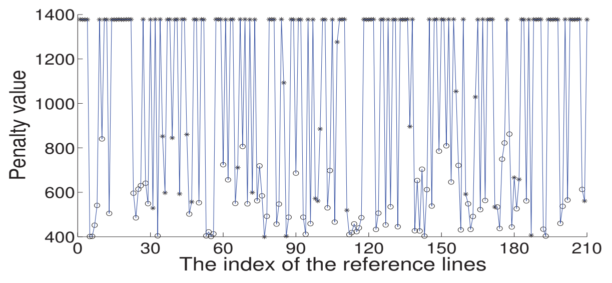
TABLE I: Mean and standard deviation values of IGD, obtained by TPEA-PBA, MOEA/DD, MOEA/D-DE, GrEA, NSGA-III, on DTLZ instances with different number of objectives.

instance	IGD					
	m	TPEA-PBA	MOEA/DD	MOEA/D-DE	GrEA	NSGA-III
DTLZ1	4	**3.031E-02(1.212E-04)**	3.210E-02(2.894E-06)†	8.146E-02(3.653E-03)†	7.754E-01(2.105E-02)†	3.230E-02(1.058E-03)†
DTLZ2	4	**8.621E-02(3.360E-04)**	8.740E-02(8.343E-07)†	1.703E-01(9.002E-04)†	3.771E-01(1.769E-02)†	8.743E-02(3.671E-05)†
DTLZ3	4	**8.553E-02(1.833E-04)**	8.743E-02(2.532E-05)†	3.235E-01(6.345E-01)†	3.696E-01(1.660E-02)†	8.808E-02(2.991E-03)†
DTLZ4	4	**8.664E-02(3.773E-04)**	8.740E-02(6.635E-07)†	1.943E-01(2.767E-02)†	3.777E-01(1.515E-02)†	8.740E-02(6.295E-06)†
DTLZ5	4	1.931E-02(6.103E-03)	6.556E-02(7.081E-03)†	**1.606E-02(4.044E-04)**	4.394E-02(1.141E-02)†	2.774E-02(4.099E-03)†
DTLZ6	4	**1.206E-02(3.036E-03)**	1.164E-01(2.446E-02)†	1.628E-02(5.088E-04)†	4.092E-02(1.479E-02)†	9.072E-02(1.965E-02)†
DTLZ7	4	1.607E-01(6.088E-03)	3.223E-01(1.118E-02)†	3.995E-01(1.781E-02)†	2.512E+00(7.663E-01)†	**1.565E-01(6.235E-03)**‡
DTLZ1	5	**4.681E-02(2.670E-04)**	5.201E-02(2.090E-05)†	1.252E-01(1.082E-02)†	3.676E-01(9.234E-02)†	5.193E-02(1.753E-04)†
DTLZ2	5	**1.311E-01(7.204E-04)**	1.332E-01(2.054E-06)†	2.640E-01(3.947E-04)†	1.462E-01(3.511E-03)†	1.332E-01(2.645E-04)†
DTLZ3	5	**1.297E-01(3.947E-04)**	1.333E-01(1.460E-04)†	4.774E-01(2.034E-01)†	8.510E-01(3.286E-01)†	1.341E-01(3.200E-03)†
DTLZ4	5	**1.325E-01(8.323E-04)**	1.332E-01(1.965E-06)†	2.934E-01(1.796E-02)†	1.440E-01(2.719E-03)†	1.331E-01(1.266E-04)†
DTLZ5	5	**2.445E-02(8.079E-03)**	1.238E-01(6.816E-03)†	3.890E-02(1.189E-03)†	3.174E-02(4.000E-03)†	9.550E-02(2.769E-02)†
DTLZ6	5	**3.525E-02(1.092E-02)**	1.622E-01(1.848E-02)†	3.885E-02(1.792E-03)†	1.652E-01(2.189E-02)†	1.364E+00(9.325E-02†)
DTLZ7	5	**3.033E-01(1.664E-02)**	5.519E-01(2.445E-02)†	5.782E-01(4.145E-02)†	1.152E+00(3.526E-02)†	4.116E-01(1.255E-01)

Wilcoxon's rank sum test at a 0.05 significance level is performed between TPEA-PBA and each of the other competing algorithms. † and ‡ denotes that the performance of the corresponding algorithm is significantly worse than or better than that of TPEA-PBA, respectively. The best mean is highlighted in boldface with gray background.



(a) DTLZ6



(b) DTLZ7

Fig. 5: The penalty values on two 5-objective DTLZ6 (a) and DTLZ7 (b) problems.

other benchmark problems according to [27]. This observation indicates TPEA-PBA is able to allocate computational resources adaptively based on the addressed problems.

E. The Effects of the Penalty Vector

Extensive experiments have been conducted to investigate the effects of the penalty vector as show in Fig.5. Where the circle ($\circ$) represents the true effective and star ($*$) represents the ineffective reference lines. The penalty vector obtained by TPEA-PBA can efficiently reflect the effectiveness of the reference lines.

TABLE II: Mean and standard deviation values of IGD, obtained by the algorithm TPEA-PBA with adjustment for Reference Lines and TPEA with non-adjustment for Reference Lines on DTLZ instances with different number of objectives.

instance	IGD		
	m	TPEA-PBA	TPEA
DTLZ5	4	**1.931E-02(6.103E-03)**	3.884E-02(1.236E-02)†
DTLZ6	4	**1.206E-02(3.036E-03)**	2.017E-02(6.081E-03)†
DTLZ7	4	**1.607E-01(6.088E-03)**	1.942E-01(2.383E-03)†
DTLZ5	5	**2.445E-02(8.079E-03)**	3.352E-02(1.077E-02)†
DTLZ6	5	**3.525E-02(1.092E-02)**	4.166E-02(1.316E-02)
DTLZ7	5	**3.033E-01(1.664E-02)**	3.334E-01(1.882E-02) †

F. The Effects of the Adjustment of the Reference Lines

We conduct further experiments to investigate the effects of the adjustment of reference lines in TPEA-PBA on DTLZ 5-7, as they are all benchmark problems with incomplete PFs. In Tables II, TPEA-PBA is compared with the one without the adjustment of reference lines (TPEA). It can be observed from the table that the performance of TPEA-PBA is much better than that of TPEA, which indicates the adjustment of the reference lines in TPEA-PBA is effective to MaOPs with incomplete PFs.

IV. Conclusion

In this paper, we proposed a two-phase many-objective evolutionary algorithm to tackle many objective optimization problems. In the first phase, the algorithm focuses on achieving good convergence on the boundary Pareto optimal solutions. In the second phase, the algorithm maintain a good balance between convergence and diversity by using a set of widely spread reference lines. In addition, a penalty based adjustment for reference line has been adopted to handle many objective optimization problems with incomplete PFs. The performance of our proposed algorithm is compared with MOEA/DD, MOEAD/DE, GrEA and NSGA-III on DTLZ benchmark problems.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (NSFC) under grant 61300159, 61175073 and 61473241, by the Natural Science Foundation of Jiangsu Province under grant BK20130808, by China Postdoctoral Science Foundation under grant 2015M571751 and by the Fundamental Research Funds for the Central Universities of China under grant NZ2013306.

References

[1] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.

[2] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[3] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[4] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[5] Eckart Zitzler and Simon Künzli. Indicator-based Selection in Multiobjective Search. In X. Yao et al., editor, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science vol. 3242.

[6] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.

[7] L.S. Batista, F. Campelo, F.G. Guimaraes, and J.A. Ramirez. A comparison of dominance criteria in many-objective optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2359–2366, 2011.

[8] F. di Pierro, Soon-Thiam Khu, and D.A. Savic. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *Evolutionary Computation, IEEE Transactions on*, 11(1):17–45, 2007.

[9] Zhenan He, G.G. Yen, and Jun Zhang. Fuzzy-based pareto optimality for many-objective evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 18(2):269–285, 2014.

[10] Ke Li, Sam Kwong, Qingfu Zhang, and K. Deb. Interrelationship-based selection for decomposition multiobjective optimization. *Cybernetics, IEEE Transactions on*, 45(10):2076–2088, 2015.

[11] Ke Li, Qingfu Zhang, Sam Kwong, Miqing Li, and Ran Wang. Stable matching-based selection in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 18(6):909–923, 2014.

[12] Ke Li, A. Fialho, S. Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 18(1):114–130, 2014.

[13] Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 13(2):284–302, 2009.

[14] L. Wang and Q. Zhang. Constrained subproblems in decomposition based multiobjective evolutionary algorithm. *Evolutionary Computation, IEEE Transactions on*, PP(99):1–1, 2015.

[15] Yi Mei, Ke Tang, and Xin Yao. Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem. *IEEE Trans. Evolutionary Computation*, 15(2):151–165, 2011.

[16] Xinye Cai and Ou Wei. A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem. In *10th IEEE International Conference on Control and Automation*, 2013.

[17] Hui Li and D. Landa-Silva. An adaptive evolutionary multi-objective approach based on simulated annealing. *Evolutionary Computation*, 19:561–595, 2011.

[18] Ke Li, K. Deb, Qingfu Zhang, and Sam Kwong. An evolutionary many-objective optimization algorithm based on dominance and decomposition. *Evolutionary Computation, IEEE Transactions on*, 19(5):694–716, 2015.

[19] Xinye Cai, Yexing Li, Zhun Fan, and Qingfu Zhang. An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. *Evolutionary Computation, IEEE Transactions on*, 19(4):508–523, 2015.

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

[21] Z. He and G.G. Yen. Many-objective evolutionary algorithm: Objective space reduction + diversity improvement. *Evolutionary Computation, IEEE Transactions on*, PP(99):1–1, 2015.

[22] K. Deb, L. Thiele, M. Laumanns, and E Zitzler. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 825–830, 2002.

[23] I. Das and Dennis Je. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *Siam Journal on Optimization*, 8(3):631–657, 2000.

[24] Qingfu Zhang, Wudong Liu, and Hui Li. The performance of a new version of MOEA/D on CEC09 unconstrained mop test instances. Working Report CES-491, School of CS and EE, University of Essex, Feburary 2009.

[25] Shengxiang Yang, Miqing Li, Xiaohui Liu, and Jinhua Zheng. A grid-based evolutionary algorithm for many-objective optimization. *Evolutionary Computation, IEEE Transactions on*, 17(5):721–736, 2013.

[26] P.A.N. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 7(2):174–188, 2003.

[27] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. *Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH*, pages 105–145, 2005.