MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition

Qingfu Zhang, *Senior Member, IEEE*, and Hui Li

***Abstract*—Decomposition is a basic strategy in traditional multiobjective optimization. However, it has not yet been widely used in multiobjective evolutionary optimization. This paper proposes a multiobjective evolutionary algorithm based on decomposition (MOEA/D). It decomposes a multiobjective optimization problem into a number of scalar optimization subproblems and optimizes them simultaneously. Each subproblem is optimized by only using information from its several neighboring subproblems, which makes MOEA/D have lower computational complexity at each generation than MOGLS and nondominated sorting genetic algorithm II (NSGA-II). Experimental results have demonstrated that MOEA/D with simple decomposition methods outperforms or performs similarly to MOGLS and NSGA-II on multiobjective 0–1 knapsack problems and continuous multiobjective optimization problems. It has been shown that MOEA/D using objective normalization can deal with disparately-scaled objectives, and MOEA/D with an advanced decomposition method can generate a set of very evenly distributed solutions for 3-objective test instances. The ability of MOEA/D with small population, the scalability and sensitivity of MOEA/D have also been experimentally investigated in this paper.**

***Index Terms*—Computational complexity, decomposition, evolutionary algorithm, multiobjective optimization, Pareto optimality.**

I. Introduction

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

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

where Ω is the *decision (variable) space*, $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions and R^m is called the *objective space*. The *attainable objective set* is defined as the set $\{F(x)|x \in \Omega\}$.

If $x \in R^n$, all the objectives are continuous and Ω is described by

$$\Omega = \{x \in R^n | h_j(x) \leq 0, j = 1, \ldots, m\}$$

where h_j are continuous functions, we call (1) a *continuous MOP*.

Very often, since the objectives in (1) contradict each other, no point in Ω maximizes all the objectives simultaneously. One has to balance them. The best tradeoffs among the objectives can be defined in terms of Pareto optimality.

Manuscript received September 4, 2006; revised November 9, 2006.

The authors are with the Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk; hlil@essex.ac.uk).

Digital Object Identifier 10.1109/TEVC.2007.892759

Let $u, v \in R^m$, u is said to *dominate* 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] A point $x^* \in \Omega$ is *Pareto optimal* to (1) if there is no point $x \in \Omega$ such that $F(x)$ dominates $F(x^*)$. $F(x^*)$ is then called a *Pareto optimal (objective) vector*. In other words, any improvement in a Pareto optimal point in one objective must lead to deterioration in at least one other 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].

In many real-life applications of multiobjective optimization, an approximation to the PF is required by a decision maker for selecting a final preferred solution. Most MOPs may have many or even infinite Pareto optimal vectors. It is very time-consuming, if not impossible, to obtain the complete PF. On the other hand, the decision maker may not be interested in having an unduly large number of Pareto optimal vectors to deal with due to overflow of information. Therefore, many multiobjective optimization algorithms are to find a manageable number of Pareto optimal vectors which are evenly distributed along the PF, and thus good representatives of the entire PF [1]–[4]. Some researchers have also made an effort to approximate the PF by using a mathematical model [5]–[8].

It is well-known that a Pareto optimal solution to a MOP, under mild conditions, could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the f_i's. Therefore, approximation of the PF can be decomposed into a number of scalar objective optimization subproblems. This is a basic idea behind many traditional mathematical programming methods for approximating the PF. Several methods for constructing aggregation functions can be found in the literature (e.g., [1]). The most popular ones among them include the weighted sum approach and Tchebycheff approach. Recently, the boundary intersection methods have also attracted a lot of attention [9]–[11].

There is no decomposition involved in the majority of the current state-of-the-art multiobjective evolutionary algorithms (MOEAs) [2]–[4], [12]–[19]. These algorithms treat a MOP as a whole. They do not associate each individual solution with any particular scalar optimization problem. In a scalar objective optimization problem, all the solutions can be compared based on their objective function values and the task of a scalar objective evolutionary algorithm (EA) is often to find one single optimal solution. In MOPs, however, domination does not define a complete ordering among the solutions in the objective space and MOEAs aim at producing a number of Pareto optimal solutions as diverse as possible for representing the whole PF. Therefore, conventional selection operators, which were

[1]This definition of domination is for maximization. All the inequalities should be reversed if the goal is to minimize the objectives in (1). "Dominate" means "be better than."

originally designed for scalar optimization, cannot be directly used in nondecomposition MOEAs. Arguably, if there is a fitness assignment scheme for assigning an individual solution a relative fitness value to reflect its utility for selection, then scalar optimization EAs can be readily extended for dealing with MOPs, although other techniques such as mating restriction [20], diversity maintaining [21], some properties of MOPs [22], and external populations [23] may also be needed for enhancing the performances of these extended algorithms. For this reason, fitness assignment has been a major issue in current MOEA research. The popular fitness assignment strategies include alternating objectives-based fitness assignment such as the vector evaluation genetic algorithm (VEGA) [24], and domination-based fitness assignment such as Pareto archived evolutionary strategy (PAES) [14], strength Pareto evolutionary algorithm II (SPEA-II) [15], and nondominated sorting genetic algorithm II (NSGA-II) [16].

The idea of decomposition has been used to a certain extent in several metaheuristics for MOPs [25]–[29]. For example, the two-phase local search (TPLS) [25] considers a set of scalar optimization problems, in which the objectives are aggregations of the objectives in the MOP under consideration, a scalar optimization algorithm is applied to these scalar optimization problems in a sequence based on aggregation coefficients, a solution obtained in the previous problem is set as a starting point for solving the next problem since its aggregation objective is just slightly different from that in the previous one. The multiobjective genetic local search (MOGLS) aims at simultaneous optimization of all aggregations constructed by the weighted sum approach or Tchebycheff approach [29]. At each iteration, it optimizes a randomly generated aggregation objective.

In this paper, we propose a new multiobjective evolutionary algorithm based on decomposition (MOEA/D). MOEA/D explicitly decomposes the MOP (1) into N scalar optimization subproblems. It solves these subproblems simultaneously by evolving a population of solutions. At each generation, the population is composed of the best solution found so far (i.e. since the start of the run of the algorithm) for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions to two neighboring subproblems should be very similar. Each subproblem (i.e., scalar aggregation function) is optimized in MOEA/D by using information only from its neighboring subproblems. MOEA/D has the following features.

- MOEA/D provides a simple yet efficient way of introducing decomposition approaches into multiobjective evolutionary computation. A decomposition approach, often developed in the community of mathematical programming, can be readily incorporated into EAs in the framework MOEA/D for solving MOPs.
- Since MOEA/D optimizes N scalar optimization problems rather than directly solving a MOP as a whole, issues such as fitness assignment and diversity maintenance that cause difficulties for nondecomposition MOEAS could become easier to handle in the framework of MOEA/D.
- MOEA/D has lower computational complexity at each generation than NSGA-II and MOGLS. Overall, MOEA/D outperforms, in terms of solution quality, MOGLS on 0–1 multiobjective knapsack test instances when both algorithms use the same decomposition approach. MOEA/D with the Tchebycheff decomposition approach performs similarly to NSGA-II on a set of continuous MOP test instances. MOEA/D with an advanced decomposition approach performs much better than NSGA-II on 3-objective continuous test instances. MOEA/D using a small population is able to produce a small number of very evenly distributed solutions.
- Objective normalization techniques can be incorporated into MOEA/D for dealing with disparately scaled objectives.
- It is very natural to use scalar optimization methods in MOEA/D since each solution is associated with a scalar optimization problem. In contrast, one of the major shortcomings of nondecomposition MOEAs is that there is no easy way for them to take the advantage of scalar optimization methods.

This paper is organized as follows. Section II introduces three decomposition approaches for MOPs. Section III presents MOEA/D. Sections IV and V compare MOEA/D with MOGLS and NSGA-II and show that MOEA/D outperforms or performs similarly to MOGLS and NSGA-II. Section VI presents more experimental studies on MOEA/D. Section VII concludes this paper.

II. Decomposition of Multiobjective Optimization

There are several approaches for converting the problem of approximation of the PF into a number of scalar optimization problems. In the following, we introduce three approaches, which are used in our experimental studies.

A. Weighted Sum Approach [1]

This approach considers a convex combination of the different objectives. Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^T$ be a weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. Then, the optimal solution to the following scalar optimization problem:

$$\begin{aligned} &\text{maximize } g^{ws}(x|\boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ &\text{subject to } x \in \Omega \end{aligned} \tag{2}$$

is a Pareto optimal point to (1),[2] where we use $g^{ws}(x|\lambda)$ to emphasize that λ is a coefficient vector in this objective function, while x is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors λ in the above scalar optimization problem. If the PF is concave (convex in the case of minimization), this approach could work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconcave PFs. To overcome these shortcomings, some effort has been made to incorporate other techniques such as ε-constraint into this approach, more details can be found in [1].

B. Tchebycheff Approach [1]

In this approach, the scalar optimization problem is in the form

$$\begin{aligned} &\text{minimize } g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i |f_i(x) - z_i^*|\} \\ &\text{subject to } x \in \Omega \end{aligned} \tag{3}$$

where $z^* = (z_1^*, \ldots, z_m^*)^T$ is the reference point, i.e., $z_i^* = \max\{f_i(x) | x \in \Omega\}$[3] for each $i = 1, \ldots, m$. For each Pareto

[2] If (1) is for minimization, "maximize" in (2) should be changed to "minimize."

[3] In the case when the goal of (1) is minimization, $z_i^* = \min\{f_i(x) | x \in \Omega\}$.

optimal point x^* there exists a weight vector λ such that x^* is the optimal solution of (3) and each optimal solution of (3) is a Pareto optimal solution of (1). Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector. One weakness with this approach is that its aggregation function is not smooth for a continuous MOP. However, it can still be used in the EA framework proposed in this paper since our algorithm does not need to compute the derivative of the aggregation function.

C. Boundary Intersection (BI) Approach

Several recent MOP decomposition methods such as Normal-Boundary Intersection Method [9] and Normalized Normal Constraint Method [10] can be classified as the BI approaches. They were designed for a continuous MOP. Under some regularity conditions, the PF of a continuous MOP is part of the most top right[4] boundary of its attainable objective set. Geometrically, these BI approaches aim to find intersection points of the most top boundary and a set of lines. If these lines are evenly distributed in a sense, one can expect that the resultant intersection points provide a good approximation to the whole PF. These approaches are able to deal with nonconcave PFs. In this paper, we use a set of lines emanating from the reference point. Mathematically, we consider the following scalar optimization subproblem[5]:

$$\begin{aligned} &\text{minimize } g^{bi}(x|\lambda, z^*) = d \\ &\text{subject to } z^* - F(x) = d\lambda, \\ &\qquad x \in \Omega \end{aligned} \tag{4}$$

where λ and z^*, as in the above subsection, are a weight vector and the reference point, respectively. As illustrated in Fig. 1, the constraint $z^* - F(x) = d\lambda$ ensures that $F(x)$ is always in L, the line with direction λ and passing through z^*. The goal is to push $F(x)$ as high as possible so that it reaches the boundary of the attainable objective set.

One of the drawbacks of the above approach is that it has to handle the equality constraint. In our implementation, we use a penalty method to deal with the constraint. More precisely, we consider[6]:

$$\begin{aligned} &\text{minimize } g^{bip}(x|\lambda, z^*) = d_1 + \theta d_2 \\ &\text{subject to } x \in \Omega \end{aligned} \tag{5}$$

where

$$d_1 = \frac{\|(z^* - F(x))^T \lambda\|}{\|\lambda\|}$$

$$\text{and } d_2 = \|F(x) - (z^* - d_1\lambda)\|.$$

$\theta > 0$ is a preset penalty parameter. Let y be the projection of $F(x)$ on the line L. As shown in Fig. 2, d_1 is the distance between z^* and y. d_2 is the distance between $F(x)$ and L. If θ is set appropriately, the solutions to (4) and (5) should be very close. Hereafter, we call this method the penalty-based boundary intersection (PBI) approach.

The advantages of the PBI approach (or general BI approaches) over the the Tchebycheff approach are as follows.

[4]In the case of minimization, it will be part of the most left bottom.

[5]In the case when the goal of (1) is minimization, this equality constraint in this subproblem should be changed to $F(x) - z^* = d\lambda$.

[6]In the case when the goal of (1) is minimization, $d_1 = \|(F(x) - z^*)^T\lambda\|/\|\lambda\|$ and to $d_2 = \|F(x) - (z^* + d_1\lambda)\|$.

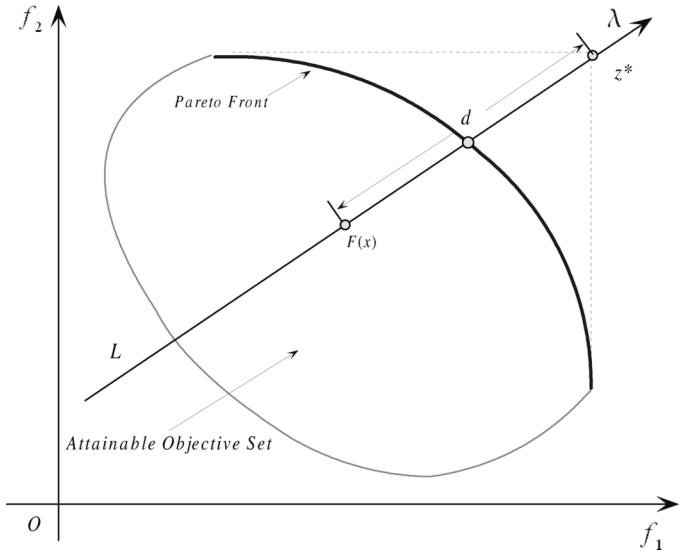

Fig. 1. Illustration of boundary intersection approach.

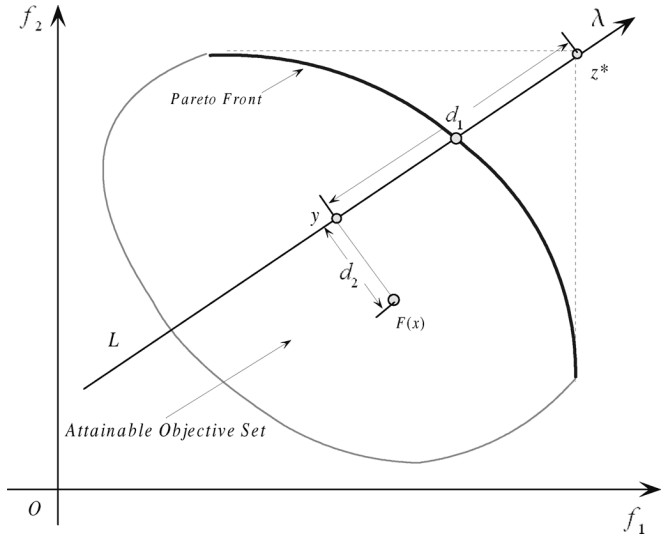

Fig. 2. Illustration of the penalty-based boundary intersection approach.

- In the case of more than two objectives, let both the PBI approach and the Tchebycheff approach use the same set of evenly distributed weight vectors, the resultant optimal solutions in the PBI should be much more uniformly distributed than those obtained by the Tchebycheff approach [9], particularly when the number of weight vectors is not large.
- If x dominates y, it is still possible that $g^{te}(x|\lambda, z^*) = g^{te}(y|\lambda, z^*)$, while it is rare for g^{bip} and other BI aggregation functions.

However, these benefits come with a price. One has to set the value of the penalty factor. It is well-known that a too large or too small penalty factor will worsen the performance of a penalty method.

The above approaches can be used to decompose the approximation of the PF into a number of scalar optimization problems. A reasonably large number of evenly distributed weight vectors usually leads to a set of Pareto optimal vectors, which may not be evenly spread but could approximate the PF very well.

There are many other decomposition approaches in the literature that could also be used in our algorithm framework. Since

our major purpose is to study the feasibility and efficiency of the algorithm framework. We only use the above three decomposition approaches in the experimental studies in this paper.

III. The Framework of Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D)

A. General Framework

Multiobjective evolutionary algorithm based on decomposition (MOEA/D), the algorithm proposed in this paper, needs to decompose the MOP under consideration. Any decomposition approaches can serve this purpose. In the following description, we suppose that the Tchebycheff approach is employed. It is very trivial to modify the following MOEA/D when other decomposition methods are used.

Let $\lambda^1, \ldots, \lambda^N$ be a set of even spread weight vectors and z^* be the reference point. As shown in Section II, the problem of approximation of the PF of (1) can be decomposed into N scalar optimization subproblems by using the Tchebycheff approach and the objective function of the jth subproblem is

$$g^{te}(x|\lambda^j, z^*) = \max_{1 \le i \le m} \{\lambda_i^j |f_i(x) - z_i^*|\} \tag{6}$$

where $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$. MOEA/D minimizes all these N objective functions simultaneously in a single run.

Note that g^{te} is continuous of λ, the optimal solution of $g^{te}(x|\lambda^i, z^*)$ should be close to that of $g^{te}(x|\lambda^j, z^*)$ if λ^i and λ^j are close to each other. Therefore, any information about these g^{te}'s with weight vectors close to λ^i should be helpful for optimizing $g^{te}(x|\lambda^i, z^*)$. This is a major motivation behind MOEA/D.

In MOEA/D, a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\{\lambda_1, \ldots, \lambda_N\}$. The neighborhood of the ith subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. The population is composed of the best solution found so far for each subproblem. Only the current solutions to its neighboring subproblems are exploited for optimizing a subproblem in MOEA/D.

At each generation t, MOEA/D with the Tchebycheff approach maintains:

- a population of N points $x^1, \ldots, x^N \in \Omega$, where x^i is the current solution to the ith subproblem;
- $FV^1, \ldots, FV^N$, where FV^i is the F-value of x^i, i.e., $FV^i = F(x^i)$ for each $i = 1, \ldots, N$;
- $z = (z_1, \ldots, z_m)^T$, where z_i is the best value found so far for objective f_i;
- an external population (EP), which is used to store nondominated solutions found during the search.

The algorithm works as follows:

Input:

- MOP (1);
- a stopping criterion;
- N: the number of the subproblems considered in MOEA/D;
- a uniform spread of N weight vectors: $\lambda^1, \ldots, \lambda^N$;
- T: the number of the weight vectors in the neighborhood of each weight vector.

Output: EP.

Step 1) Initialization:

Step 1.1) Set EP $= \emptyset$.

Step 1.2) Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.3) Generate an initial population $x^1, \ldots, x^N$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 1.4) Initialize $z = (z_1, \ldots, z_m)^T$ by a problem-specific method.

Step 2) Update:

For $i = 1, \ldots, N$, do

Step 2.1) Reproduction: Randomly select two indexes k, l from $B(i)$, and then generate a new solution y from x^k and x^l by using genetic operators.

Step 2.2) Improvement: Apply a problem-specific repair/improvement heuristic on y to produce y'.

Step 2.3) Update of z: For each $j = 1, \ldots, m$, if $z_j < f_j(y')$, then set $z_j = f_j(y')$.

Step 2.4) Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{te}(y'|\lambda^j, z) \le g^{te}(x^j|\lambda^j, z)$, then set $x^j = y'$ and $FV^j = F(y')$.

Step 2.5) Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Step 3) Stopping Criteria: If stopping criteria is satisfied, then stop and output EP. Otherwise, go to **Step 2**.

In initialization, $B(i)$ contains the indexes of the T closest vectors of λ^i. We use the Euclidean distance to measure the closeness between any two weight vectors. Therefore, λ^i's closest vector is itself, and then $i \in B(i)$. If $j \in B(i)$, the jth subproblem can be regarded as a neighbor of the ith subproblem.

In the ith pass of the loop in Step 2, the T neighboring subproblems of the ith subproblem are considered. Since x^k and x^l in Step 2.1 are the current best solutions to neighbors of the ith subproblem, their offspring y should hopefully be a good solution to the ith subproblem. In Step 2.2, a problem-specific heuristic[7] is used to repair y in the case when y invalidates any constraints, and/or optimize the ith g^{te}. Therefore, the resultant solution y' is feasible and very likely to have a lower function value for the neighbors of ith subproblem. Step 2.4 considers all the neighbors of the ith subproblem, it replaces x^j with y' if y' performs better than x^j with regard to the jth subproblem. FV^j is needed in computing the value of $g^{te}(x^j|\lambda^j, z)$ in Step 2.4.

Since it is often very time-consuming to find the exact reference point z^*, we use z, which is initialized in Step 1.4 by a

[7]An exemplary heuristic can be found in the implementation of MOEA/D for the MOKP in Section IV.

problem-specific method[8] and updated in Step 2.3, as a substitute for z^* in g^{te}. The external population EP, initialized in Step 1.1, is updated by the new generated solution y' in Step 2.5.

In the case when the goal in (1) is to minimize $F(x)$, the inequality in Step 2.3 should be reversed.

B. Discussions

1) Why a Finite Number of Subproblems are Considered in MOEA/D: A weight vector used in MOEA/D is one of the N preselected weight vectors. MOEA/D spends about the same amount of effort on each of the N aggregation functions, while MOGLS randomly generates a weight vector at each iteration, aiming at optimizing all the possible aggregation functions. Recall that a decision maker only needs a finite number of evenly distributed Pareto solutions, optimizing a finite of selected scalar optimization subproblems is not only realistic but also appropriate. Since the computational resource is always limited, optimizing all the possible aggregation functions would not be very practical, and thus may waste some computational effort.

2) How Diversity is Maintained in MOEA/D: As mentioned in Section I, a MOEA needs to maintain diversity in its population for producing a set of representative solutions. Most, if not all, of nondecomposition MOEAs such as NSGA-II and SPEA-II use crowding distances among the solutions in their selection to maintain diversity. However, it is not always easy to generate a uniform distribution of Pareto optimal objective vectors in these algorithms. In MOEA/D, a MOP is decomposed into a number of scalar optimization subproblems. Different solutions in the current population are associated with different subproblems. The "diversity" among these subproblems will naturally lead to diversity in the population. When the decomposition method and the weight vectors are properly chosen, and thus the optimal solutions to the resultant subproblems are evenly distributed along the PF, MOEA/D will have a good chance of producing a uniform distribution of Pareto solutions if it optimizes all these subproblems very well.

3) Mating Restriction and the Role of T in MOEA/D: T is the size of the neighborhood. Only current solutions to the T closest neighbors of a subproblem are used for optimizing it in MOEA/D. In a sense, two solutions have a chance to mate only when they are for two neighboring subproblems. This is a mating restriction. Attention should be paid to the setting of T. If T is too small, two solutions chosen (x^k and x^l) for undergoing genetic operators in Step 2.1 may be very similar since they are for very similar subproblems, consequently, y', the solution generated in Step 2.2, could be very close to their parents x^k and x^l. Therefore, the algorithm lacks the ability to explore new areas in the search space. On the other hand, if T is too large, x^k and x^l may be poor for the subproblem under consideration, and so is their offspring y'. Therefore, the exploitation ability of the algorithm is weakened. Moreover, a too large T will increase the computational overhead of Step 2.4.

C. Variants of MOEA/D

We can use any other decomposition methods in MOEA/D. When the weighted sum approach is used, MOEA/D does not need to maintain z.

[8]Two exemplary methods can be found in Sections IV and V.

Step 2.2 allows MOEA/D to be able to make use of a scalar optimization method very naturally. One can take the $g^{te}(x|\lambda^i, z)$ or $g^{ws}(x|\lambda^i)$ as the objective function in the heuristic in Step 2.2. Although it is one of the major features of MOEA/D, Step 2.2 is not a must in MOEA/D, particularly if Step 2.1 can produce a feasible solution.

Using the external population EP is also an option, although it is often very helpful for improving the performance of the algorithm. An alternative is to return the final internal population as an approximation to the PF when EP is not maintained. Of course, other sophisticated strategies [21] for updating EP can be easily adopted in the framework of MOEA/D. One can also limit the size of EP to avoid any possible memory overflow problem.

The cellular multiobjective genetic algorithm (cMOGA) of Murata *et al.* [40] can also be regarded as an evolutionary algorithm using decomposition. In essence it uses the same neighborhood relationship for mating restriction. cMOGA differs from MOEA/D in selecting solutions for genetic operators and updating the internal population, and it has to insert solutions from its external population to its internal population at each generation for dealing with nonconvex PFs, since it uses weighted sums of the objectives as its guided functions and there is no mechanism for keeping the best solution found so far to each subproblem in its internal population.

IV. Comparison With MOGLS

In the following, we first introduce MOGLS and then analyze the complexity of MOGLS and MOEA/D. We also compare the performances of these two algorithms on a set of test instances of the multiobjective 0/1 knapsack problem. The major reasons for choosing MOGLS for comparison are that it is also based on decomposition and performs better than a number of popular algorithms on the multiobjective 0/1 knapsack problem [29].

A. MOGLS

MOGLS was first proposed by Ishibuchi and Murata in [28], and further improved by Jaszkiewicz [29]. The basic idea is to reformulate the MOP (1) as simultaneous optimization of all weighted Tchebycheff functions or all weighted sum functions. In the following, we give a brief description of Jaszkiewicz's version of MOGLS.

At each iteration, MOGLS maintains:

- a set of current solutions (CS), and the F-values of these solutions;
- an external population (EP), which is used to store nondominated solutions.

If MOGLS optimizes weighted Tchebycheff functions, it should also maintain:

- $z = (z_1, \ldots, z_m)^T$, where z_i is the largest value found so far for objective f_i.

MOGLS needs two control parameters K and S. K is the size of its temporary elite population and S is the initial size of CS.

MOGLS works as follows:

Input:
- MOP (1);
- a stopping criterion;
- K: the size of temporary elite population;
- S: the size of initial population.

Output: EP.

Step 1) Initialization:

Step 1.1) Generate S initial solutions $x^1, \ldots, x^S$ randomly or by a problem-specific method. Then, CS is initialized to be $\{x^1, \ldots, x^S\}$.

Step 1.2) Initialize $z = (z_1, \ldots, z_m)^T$ by a problem-specific method.

Step 1.3) EP is initialized to be the set of the F-values of all the nondominated solutions in CS.

Step 2) Update:

Step 2.1) Reproduction:

Uniformly randomly generate a weight vector λ.

From CS select the K best solutions, with regard to the Tchebycheff aggregation function g^{te} with the weight vector λ, to form a temporary elite population (TEP).

Draw at random two solutions from TEP, and then generate a new solution y from these two solutions by using genetic operators.

Step 2.2) Improvement: Apply a problem-specific repair/improvement heuristic on y to generate y'.

Step 2.3) Update of z: For each $j = 1, \ldots, m$, if $z_j < f_j(y')$, then set $z_j = f_j(y')$.

Step 2.4) Update of Solutions in TEP:

If y' is better than the worst solution in TEP with regard to g^{te} with the weight vector λ and different from any solutions in TEP with regard to F-values, then add it to the set CS.

If the size of CS is larger than $K \times S$, delete the oldest solution in CS.

Step 2.5) Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominates $F(y')$.

Step 3) Stopping Criteria: If stopping criteria is satisfied, then stop and output EP. Otherwise, go to **Step 2**.

As in MOEA/D with the Tchebycheff approach, z is used as a substitute for z^* in g^{te}. In the case of MOGLS with the weighted sum approach, g^{te} should be replaced by g^{ws} and there is no need to store z. Therefore, Step 2.3 should be removed. MOGLS needs to keep the F-values of all the current solutions since these F-values will be used in Step 2.4 for computing the values of g^{te}.

B. Comparison of Complexity of MOEA/D and MOGLS

1) Space Complexity: During the search, MOEA/D needs to maintain its internal population of N solutions, and external population EP, while MOGLS stores the set of current solutions CS and its external population EP. The size of CS increases until it reaches its upper bound, which is suggested to be set as $K \times S$ in [29]. Therefore, if $K \times S$ in MOGLS is much larger than N in MOEA/D and both algorithms produce about the same number of nondominated solutions, then the space complexity of MOEA/D is lower than that of MOGLS.

2) Computational Complexity: The major computational costs in both MOEA/D and MOGLS are involved in their Step 2. Both a single pass of Step 2 in MOEA/D and Step 2 in MOGLS generates one new trial solution y'. Let us compare the computational complexity in a single pass of Step 2 in MOEA/D and Step 2 in MOGLS.

- Step 2.1 in MOEA/D versus Step 2.1 in MOGLS: MOGLS has to generate TEP. One has to compute the g^{te}-values of all the points in CS, which needs $O(m \times |\mathrm{CS}|)$ basic operations. It also needs $O(K \times |\mathrm{CS}|)$ basic operations for selecting TEP if a naive selection method is employed, while Step 2.1 in MOEA/D only needs to randomly pick two solutions for genetic operators. Note that $|\mathrm{CS}|$, the size of CS, could be very large (e.g, it was set from 3000 to 7000 in [29]), the computational cost of Step 2.1 in MOGLS is much higher than that of Step 2.1 in MOEA/D.
- Steps 2.2 and 2.3 in MOEA/D are the same as Steps 2.2 and 2.3 in MOGLS, respectively.
- Step 2.4 in MOEA/D versus Step 2.4 in MOGLS: Step 2.4 in MOEA/D needs $O(T)$ basic operations, while Step 2.4 in MOGLS needs $O(K)$ basic operations. In the case when T and K are close as in our experiments, there is no significant difference in computational costs between them.

Therefore, we can conclude that each pass of Step 2 in MOEA/D involves less computational cost than Step 2 in MOGLS does.

C. Multiobjective 0–1 Knapsack Problem

Given a set of n items and a set of m knapsacks, the multiobjective 0–1 knapsack problem (MOKP) can be stated as

$$\begin{aligned} \text{maximize } & f_i(x) = \sum_{j=1}^{n} p_{ij} x_j,\ i = 1, \ldots, m \\ \text{subject to } & \sum_{j=1}^{n} w_{ij} x_j \le c_i,\ i = 1, \ldots, m \\ & x = (x_1, \ldots, x_n)^T \in \{0,1\}^n \end{aligned} \tag{7}$$

where $p_{ij} \ge 0$ is the profit of item j in knapsack i, $w_{ij} \ge 0$ is the weight of item j in knapsack i, and c_i is the capacity of knapsack i. $x_i = 1$ means that item i is selected and put in all the knapsacks.

The MOKP is NP-hard and can model a variety of applications in resource allocation. A set of nine test instances of the above problem have been proposed in [13] and widely used in testing multiobjective heuristics. MOGLS outperforms a number of MOEAs on these test instances [29]. In this paper,

we will also use these nine instances for comparing the performances of MOEA/D and MOGLS.

D. Implementations of MOEA/D and MOGLS for MOKP

1) Repair Method: To apply an EA for the MOKP, one needs a heuristic for repairing infeasible solutions. Several repair approaches have been proposed for this purpose [13], [29].

Let $y = (y_1, \ldots, y_n)^T \in \{0,1\}^n$ be an infeasible solution to (7). Note that w_{ij} and p_{ij} in (7) are nonnegative, one can remove some items from it (i.e., change the values of some y_i from 1 to 0) for making it feasible. Recently, Jaszkiewicz proposed and used the following greedy repair method[9].

Input:
- MOP (7);
- a solution: $y = (y_1, \ldots, y_n)^T$;
- an objective function to be maximized: $g : \{0,1\}^n \rightarrow R$

Output: a feasible solution $y' = (y'_1, \ldots, y'_n)^T$.

Step 1) If y is feasible, then set $y' = y$ and return y'.

Step 2) Set $J = \{j | 1 \leq j \leq n \text{and } y_j = 1\}$ and $I = \{i | 1 \leq i \leq m \text{ and } \sum_{j=1}^{n} w_{ij} y_j > c_i, \}$.

Step 3) Select $k \in J$ such that

$$k = \arg\min_{j \in J} \frac{g(y) - g(y^{j-})}{\sum_{i \in I} w_{ij}}$$

where y^{j-} is different from y only in position j, i.e., $y_i^{j-} = y_i$ for all $i \neq j$ and $y_j^{j-} = 0$.

Set $y_k = 0$ and go to **Step 1**.

In this approach, items are removed one by one from y until y becomes feasible. An item with the heavy weights (i.e., $\sum_{i \in I} w_{ij}$) in the overfilled knapsacks and little contribution to $g(x)$ (i.e., $g(y) - g(y^{j-})$) is more likely to be removed.

2) Implementation of MOGLS: To have a fair comparison, we directly use Jaszkiewicz's latest implementation of MOGLS for the MOKP, the details of MOGLS with the Tchebycheff approach are given as follows.

- **Initialization of** $z = (z_1, \ldots, z_m)^T$: Taking each f_i as the objective function, apply the repair method on a randomly generated point and produce a feasible solution. Set z_i to be the f_i value of the resultant point.
- **Initialization of** EP **and** CS:

  Set EP $= \emptyset$ and CS $= \emptyset$. Then, **Repeat** S times:

  1. Randomly generate a weight vector λ by using the sampling method described in [29].

  2. Randomly generate a solution $x = (x_1, \ldots, x_n)^T \in \{0,1\}^n$, where the probability of $x_i = 1$ equals to 0.5.

  3. Taking $-g^{te}(x|\lambda, z)$ as the objective function, apply the repair method to x and obtain a feasible solution x'.

  4. Add x' to CS. Remove from EP all the vectors dominated by $F(y')$, then add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

- **Genetic operators in Step 2.1**: The genetic operators used are the one-point crossover operator and the standard mutation operator. The one-point crossover is first applied to the two solutions and generates one child solution, then the standard mutation operator mutates it to produce a new solution y. The mutation mutates each position of the child solution independently with probability 0.01.
- **Heuristic in Step 2.2**: The repair method described in this section is used.

3) Implementation of MOEA/D: We use the same genetic operators in Step 2.1 and the repair method in Step 2.2 as in the implementation of MOGLS. The initialization of z is also the same as in MOGLS. Initialization of x^i (the initial solution to the ith subproblem) is performed as follows.

- **Initialization of** x^i **in Step 1.3**: Taking $-g^{te}(x|\lambda^i, z)$ as the objective function, apply the repair method to a randomly generated solution. Set x^i to be the resultant solution.

Tchebycheff aggregation function g^{te} is used in the above implementations of MOEA/D and MOGLS.

The implementations of these two algorithms with the weighted sum approach in our experimental studies are the same as their counterparts with the Tchebycheff approach, except that they use g^{ws} as the objective in the repair method and do not maintain z.

E. Parameter Setting

The setting of S and K in MOGLS, which determines the size of CS, is the same as in [29]. K is set to 20 for all the instances. The values of S for different instances are given in Table I.

T in MOEA/D is set to 10 for all the test instances. The setting of N and $\lambda^1, \ldots, \lambda^N$ in MOEA/D is controlled by a parameter H. More precisely, $\lambda^1, \ldots, \lambda^N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}.$$

Therefore, the number of such vectors is

$$N = C_{H+m-1}^{m-1}.$$

Table I lists the value of N and H in MOEA/D for each test instance. For the instances with two objectives, the value of N in MOEA/D is the same as that of S in MOGLS. For all the instances with three objectives, $H = 25$, and therefore $N = 351$. For all the instances with four objectives, $H = 12$, and then $N = 455$. It should be noted that the size of the internal population CS in MOGLS can reach $K \times S$, much larger than that of the internal population in MOEA-D.

The above method for generating weight vectors in MOEA-D works well in our experiments. It could, however, result in a very large N when m, the number of the objectives is large. To overcome this shortcoming, one should resort to advanced experimental design methods [30], [31] to generate weight vectors.

[9]This approach is used in the latest version of his implementation, which can be downloaded from his web http://www-idss.cs.put.poznan.pl/~jaszkiewicz/ and is slightly better than that used in his earlier paper [29].

TABLE I
PARAMETER SETTING OF MOEA/D AND MOGLS FOR THE TEST INSTANCES OF THE 0/1 KNAPSACK PROBLEM

Instance		S	N (H)
m: # of objectives	n: # of items	in MOGLS	in MOEA/D
2	250	150	150 (149)
2	500	200	200 (199)
2	750	200	250 (249)
3	250	200	351 (25)
3	500	250	351 (25)
3	750	300	351 (25)
4	250	250	455 (12)
4	500	300	455 (12)
4	750	350	455 (12)

Both of the algorithms stop after $500 \times S$ calls of the repair method.

In our experimental studies, both g^{ws} and g^{te} have been used in the repair method. In the following, W-MOEA/D (W-MOGLS) stands for MOEA/D (MOGLS) in which g^{ws} is used, while T-MOEA/D (T-MOGLS) represents MOEA/D (MOGLS) in which g^{te} is used.

F. Experimental Results

Both MOGLS and MOEA/D have been independently run for 30 times for each test instance on identical computers (Pentium(R) 3.2 GHZ, 1.00 GB). Due to the nature of MOPs, multiple performance indexes should be used for comparing the performances of different algorithms [2], [32]. In our experiments, the following performance indexes are used.

- **Set Coverage (C-metric)**: Let A and B be two approximations to the PF of a MOP, $C(A, B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A, i.e.,

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \text{ dominates } u\}|}{|B|}$$

  $C(A, B)$ is not necessarily equal to $1 - C(B, A)$. $C(A, B) = 1$ means that all solutions in B are dominated by some solutions in A, while $C(A, B) = 0$ implies that no solution in B is dominated by a solution in A.
- **Distance from Representatives in the PF (D-metric)**: Let P^* be a set of uniformly distributed points along the PF. Let A be an approximation to the PF, the average distance from P^* to A is defined as

$$D(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|}$$

  where $d(v, A)$ is the minimum Euclidean distance between v and the points in A. If $|P^*|$ is large enough to represent the PF very well, $D(A, P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $D(A, P^*)$, set A must be very close to the PF and cannot miss any part of the whole PF.

  In the case when we do not know the actual PF, we can set P^* to be an upper approximation of the PF. Jaszkiewicz has produced a very good upper approximation to each 0/1 knapsack test instance by solving the linear programming relaxed version of (3) with a number of uniformly distributed λ's [29]. The number of the points in the upper approximation is 202 for each of the bi-objective instances, 1326 for the 3-objective instances, and 3276 for the 4-objectives. In our experiments, P^* is set as such an approximation.

TABLE II
AVERAGE CPU TIME (IN SECONDS) USED BY MOEA/D AND MOGLS

Decomposition Method			Tchebycheff		Weighted Sum	
	m	n	MOEA/D	MOGLS	MOEA/D	MOGLS
Instance	2	250	3.93	26.47	3.70	29.37
	2	500	10.40	70.80	9.40	72.97
	2	750	20.13	127.30	17.87	129.60
	3	250	7.00	81.70	6.53	78.17
	3	500	17.50	147.70	15.30	137.13
	3	750	31.90	219.30	26.73	202.47
	4	250	17.33	188.80	19.60	186.17
	4	500	42.60	292.10	45.17	287.60
	4	750	75.17	425.33	70.93	396.20

TABLE III
AVERAGE SET COVERAGE BETWEEN MOEA/D (A) AND MOGLS (B)

Decomposition Method			Tchebycheff		Weighted Sum	
	m	n	$C(A, B)$	$C(B, A)$	$C(A, B)$	$C(B, A)$
Instance	2	250	93.17	3.99	45.06	40.81
	2	500	95.45	3.54	55.93	29.47
	2	750	97.22	2.21	73.98	15.85
	3	250	78.83	7.88	45.22	21.76
	3	500	90.97	2.42	61.74	10.09
	3	750	98.03	0.39	78.01	4.33
	4	250	29.01	26.48	19.34	26.11
	4	500	35.98	14.44	25.53	16.41
	4	750	52.70	6.49	41.40	8.52

Table II gives the average CPU time used by each algorithm for each instance. Table III presents the means of the C-metric values of the final approximations obtained by the two algorithms with two different repair methods.

Table IV shows the mean and standard deviation of the D-metric values in MOEA/D and MOGLS for each instance. Fig. 3 shows the evolution of the average D-metric value of EP from P^* in 30 runs with the number of the calls of the repair method in each algorithm for each test instance. Since the ranges of D-metric values are large, we use the logarithmic scale for the axes of the average D-metric value in Fig. 3. Figs. 4 and 5 plot the distributions of EP with the lowest D-metric value found in each algorithm for the three bi-objective instances.

We can make the following remarks:

- With the same number of the calls of a repair method (i.e., the same number of trial solutions), it is evident from Table II that MOEA/D needs less computational

TABLE IV
D-METRIC VALUES OF THE SOLUTIONS FOUND BY MOEA/D AND MOGLS. THE NUMBERS IN PARENTHESES REPRESENT THE STANDARD DEVIATION

Decomposition Method			Tchebycheff		Weighted Sum	
	m	n	MOEA/D	MOGLS	MOEA/D	MOGLS
Instance	2	250	54.10 (4.57)	96.38 (6.03)	37.17(2.98)	38.18(3.21)
	2	500	184.85 (11.88)	328.00 (19.85)	79.07(5.41)	98.63 (9.16)
	2	750	437.07 (21.19)	765.66 (44.95)	166.04(13.63)	274.20 (22.70)
	3	250	158.71 (6.58)	217.25 (8.45)	97.75(7.23)	141.21 (12.25)
	3	500	489.27 (15.91)	701.70(27.40)	270.31 (11.92)	419.15 (23.86)
	3	750	960.17 (23.62)	1378.64 (54.63)	446.12(19.12)	768.30 (31.86)
	4	250	253.23 (7.17)	301.32 (6.33)	176.52 (7.25)	266.17 (9.27)
	4	500	763.96 (14.81)	964.41 (24.85)	431.94(11.59)	725.16 (24.57))
	4	750	1546.44 (27.78)	1994.78 (72.84)	761.57 (17.29)	1246.54 (29.16)

time than MOGLS does. On average, MOEA/D requires about 14% of the CPU time that MOGLS needs. In other words, MOEA/D is seven times as fast as MOGLS. This observation agrees with our analysis of the computational complexity of MOEA/D and MOGLS in Section IV-B.

- Fig. 3 clearly indicates that for all the test instances, W-MOEA/D (T-MOEA/D) needs fewer calls of a repair method than MOGLS for minimizing the D-metric value, which suggests that MOEA/D is more efficient and effective than MOGLS for the MOKP.
- Tables III and IV show that the final EP obtained by W-MOEA/D (T-MOEA/D) is better than that obtained by W-MOGLS (T-MOGLS), in terms of both D-metric and C-metric, for all the test instances except instance 250–4 in which W-MOEA/D is slightly worse than W-MOGLS in C-metric. Taking instance 500–3 as an example, on average, 90.97% of the final solutions generated by T-MOGLS are dominated by those generated by T-MOEA/D, and only 2.42% vice versa. The difference between the approximations by T-MOEA/D and T-MOGLS on instances 250–2, 500–2 and 750–2 can be visually detected from Fig. 5, while difference between W-MOEA/D and W-MOGLS in middle part of the fronts on instance 750–2 can be spotted from Fig. 4.
- Table IV also shows that the standard deviation of D-metric in W-MOEA/D (T-MOEA/D) is smaller than that in W-MOGLS (T-MOGLS) for all the instances, which implies that MOEA/D is more stable than MOGLS.
- Table IV and Fig. 3 reveal that the weighted sum approach outperforms the Tchebycheff approach in both MOEA/D and MOGLS, which suggests that different decomposition approaches in MOEA/D and MOGLS can have different performances.

Overall, we can claim that MOEA/D is computationally much cheaper and can produce better approximations than MOGLS on these MOKP test instances.

V. COMPARISON WITH NSGA-II ON CONTINUOUS MOPS

A. Multiobjective Continuous Test Suites

We use five widely used bi-objective ZDT test instances [33] and two 3-objective instances [34] in comparing MOEA/D with NSGA-II [16], one of the most successful nondecomposition MOEAs. All these test instances are minimization of the objectives.

- ZDT1

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{\frac{f_1(x)}{g(x)}}\right]$$
$$\text{where } g(x) = 1 + \frac{9\left(\sum_{i=2}^{n} x_i\right)}{(n-1)}$$

  and $x = (x_1, \ldots, x_n)^T \in [0,1]^n$. Its PF is convex. $n = 30$ in our experiments.
- ZDT2

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right]$$

  where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. The PF of ZDT2 is nonconvex.
- ZDT3

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)}\sin(10\pi x_1)\right]$$

  where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. Its PF is disconnected. The two objectives are disparately scaled in the PF, f_1 is from 0 to 0.852, while f_2 from -0.773 to 1.
- ZDT4

$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{\frac{f_1(x)}{g(x)}}\right]$$

  where
$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}\left[x_i^2 - 10\cos(4\pi x_i)\right]$$

  and $x = (x_1, \ldots, x_n)^T \in [0,1] \times [-5,5]^{n-1}$. It has many local PFs. $n = 10$ in our experiments.

- ZDT6

$$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$
$$f_2(x) = g(x)\left[1 - \left(\frac{f_1(x)}{g(x)}\right)^2\right]$$
$$\text{where } g(x) = 1 + 9\left[\frac{\left(\sum_{i=2}^{n} x_i\right)}{(n-1)}\right]^{0.25}$$

  and $x = (x_1, \ldots, x_n)^T \in [0,1]^n$. Its PF is nonconvex. The distribution of the Pareto solutions in the Pareto front is very nonuniform, i.e., for a set of uniformly distributed points in the Pareto set in the decision space, their images crowd in a corner of the Pareto front in the objective space. $n = 10$ in our experiments.

- DTLZ1

$$f_1(x) = (1 + g(x))x_1x_2$$
$$f_2(x) = (1 + g(x))x_1(1 - x_2)$$
$$f_3(x) = (1 + g(x))(1 - x_1)$$

  where

$$g(x) = 100(n-2) + 100\sum_{i=3}^{n}\{(x_i - 0.5)^2 - \cos[20\pi(x_i - 0.5)]\}$$

  and $x = (x_1, \ldots, x_n)^T \in [0,1]^n$. Its PF is nonconvex. The function value of a Pareto optimal solution satisfies $\sum_{i=1}^{3} f_i = 1$ with $f_i \geq 0$, $i = 1, 2, 3$. $n = 10$ in our experiments.

- DTLZ2

$$f_1(x) = (1 + g(x))\cos\left(\frac{x_1\pi}{2}\right)\cos\left(\frac{x_2\pi}{2}\right)$$
$$f_2(x) = (1 + g(x))\cos\left(\frac{x_1\pi}{2}\right)\sin\left(\frac{x_2\pi}{2}\right)$$
$$f_3(x) = (1 + g(x))\sin\left(\frac{x_1\pi}{2}\right) \quad (8)$$

  where

$$g(x) = \sum_{i=3}^{n} x_i^2$$

  and $x = (x_1, \ldots, x_n)^T \in [0,1]^2 \times [-1,1]^{n-2}$. Its PF is nonconvex. The function value of a Pareto optimal solution satisfies $\sum_{i=1}^{3} {f_i}^2 = 1$ with $f_i \geq 0$, $i = 1, 2, 3$. $n = 10$ in our experiments.

B. NSGA-II [16]

NSGA-II does not use an external population. NSGA-II maintains a population P_t of size N at generation t and generates P_{t+1} from P_t in the following way.

Step 1) Use selection, crossover and mutation to create an offspring population Q_t from P_t.

Step 2) Choose N best solutions from $P_t \bigcup Q_t$ to form P_{t+1}.

The characteristic feature of NSGA-II is that it uses a fast nondominated sorting and crowded distance estimation procedure for comparing qualities of different solutions in **Step 2** and selection in **Step 1**. The computational complexity of each generation in NSGA-II is $O(mN^2)$, where m is the number of the objectives and N is its population size.

C. Variant of MOEA/D Used in Comparison

To have a fair comparison, we use the following variant of MOEA/D in our experiments.

- There is no external population EP. Instead, the final internal population is returned as an approximation to the PF. Step 2.5 is not needed.
- No repair/improvement method is used, therefore, Step 2.2 is not needed.
- g^{ts} is used in Step 2.4.

We do not use g^{ws} mainly because the weighted sum approach is unable to deal with nonconvex PFs as is the case in some test instances. Since NSGA-II has no external population, we do not maintain an external population in this implementation of MOEA/D. In comparison with NSGA-II, the only extra memory requirement in MOEA/D is for storing z. The size of z is $O(m)$, which is very small compared with the population size.

D. Comparison of Computational Complexity of the Variant of MOEA/D and NSGA-II

In the above variant of MOEA/D, the major computational costs are in Step 2. Step 2 in MOEA/D generates N trial solutions, so does NSGA-II at each generation. Note that Steps 2.2 and 2.5 have been removed from this variant, Step 2.1 just randomly pick two solutions for genetic operators, Step 2.3 performs $O(m)$ comparisons and assignments, and Step 2.4 needs $O(mT)$ basic operations since its major costs are to compute the values of g^{te} for T solutions and computation of one such a value requires $O(m)$ basic operations. Therefore, the computational complexity of Step 2 in the above variant of MOEA/D is $O(mNT)$ since it has N passes. If both MOEA/D and NSGA-II use the same population size, the ratio between their computational complexities at each generation is

$$\frac{O(mNT)}{O(mN^2)} = \frac{O(T)}{O(N)}.$$

Since T is smaller than N, the variant of MOEA/D has lower computational complexity than NSGA-II at each generation.

E. Experimental Setting

In our experimental studies, the implementation of NSGA-II follows [16]. The population size N in both NSGA-II and MOEA/D is set to be 100 for all the 2-objective test instances, and 300 for the other two 3-objective test instances. Both algorithms stop after 250 generations.

Initial populations are generated by uniformly randomly sampling from the feasible search space in both algorithms. z^i in MOEA/D is initialized as the lowest value of f_i found in the initial population. The simulated binary crossover (SBX) and polynomial mutation are used in both NSGA-II and MOEA/D. More precisely, in Step 2.1 of MOEA/D, the crossover operator generates one offspring, which is then modified by the mutation operator. The setting of the control parameters in these two

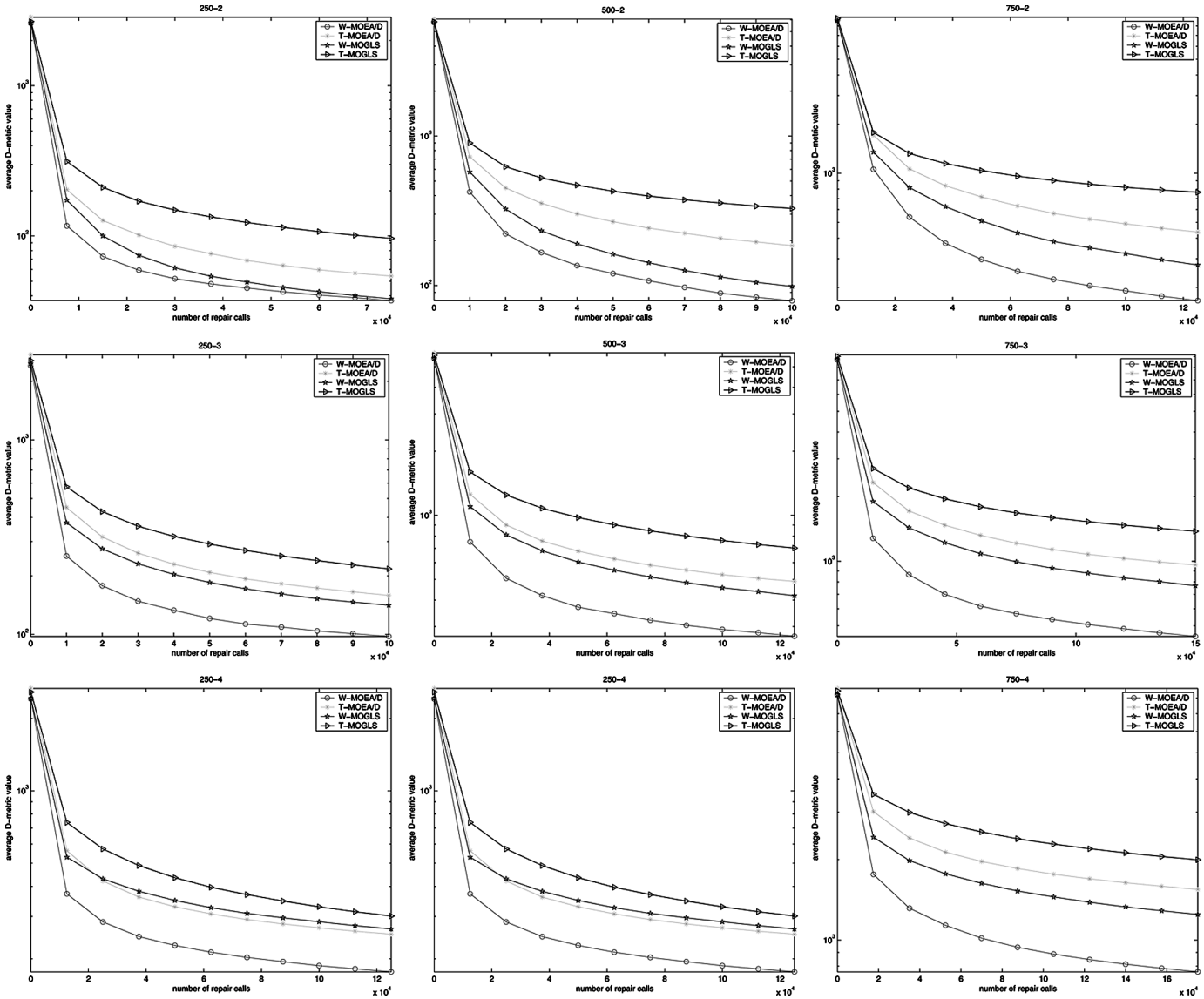

Fig. 3. The evolution of the average D-metric in MOEA/D and MOGLS for all the MOKP instances. The base 10 logarithmic scale is used for the y axis in all these figures.

operators is the same in both algorithms. Following the practice in [16], the distribution indexes in both SBX and the polynomial mutation are set to be 20. The crossover rate is 1.00, while the mutation rate is $1/n$, where n is the number of decision variables.

In MOEA/D, the setting of weight vectors $\lambda^1, \ldots, \lambda^N$ is the same as in Section IV-E, T is set to be 20.

Both MOEA/D and NSGA-II have been run 30 times independently for each test instance.

F. Experimental Results

Table V presents the average CPU time used by each algorithm for each test instance. It is clear from Table V that, on average, MOEA/D runs about twice as fast as NSGA-II with the same number of function evaluations for the 2-objective test instances, and more than 8 times for the 3-objective test instances. This observation is consistent with our complexity analysis in Section V-D.

As in the MOKP, we use both the C-metric and D-metric to compare the performances of these two algorithms. To compute D-metric values, P^* is chosen to be a set of 500 uniformly distributed points in the PF for all the 2-objective test instances, and 990 points for 3-objective instances.

The number of function evaluations matters when the objective functions are very costly to evaluate. Fig. 6 presents the evolution of the average D-metric value of the current population to P^* with the number of function evaluations in each algorithm for each test instance. These results indicate that MOEA/D converges, in terms of the number of the function evaluations, much faster than NSGA-II in minimizing the D-metric value for ZDT4, ZDT6, DTLZ1 and DTLZ2, and at the about same speed as or a bit slower than NSGA-II for the other three test instances.

Table VI shows that in terms of C-metric, the final solutions obtained by MOEA/D is better than those obtained by NSGA-II for all the test instances but ZDT4.

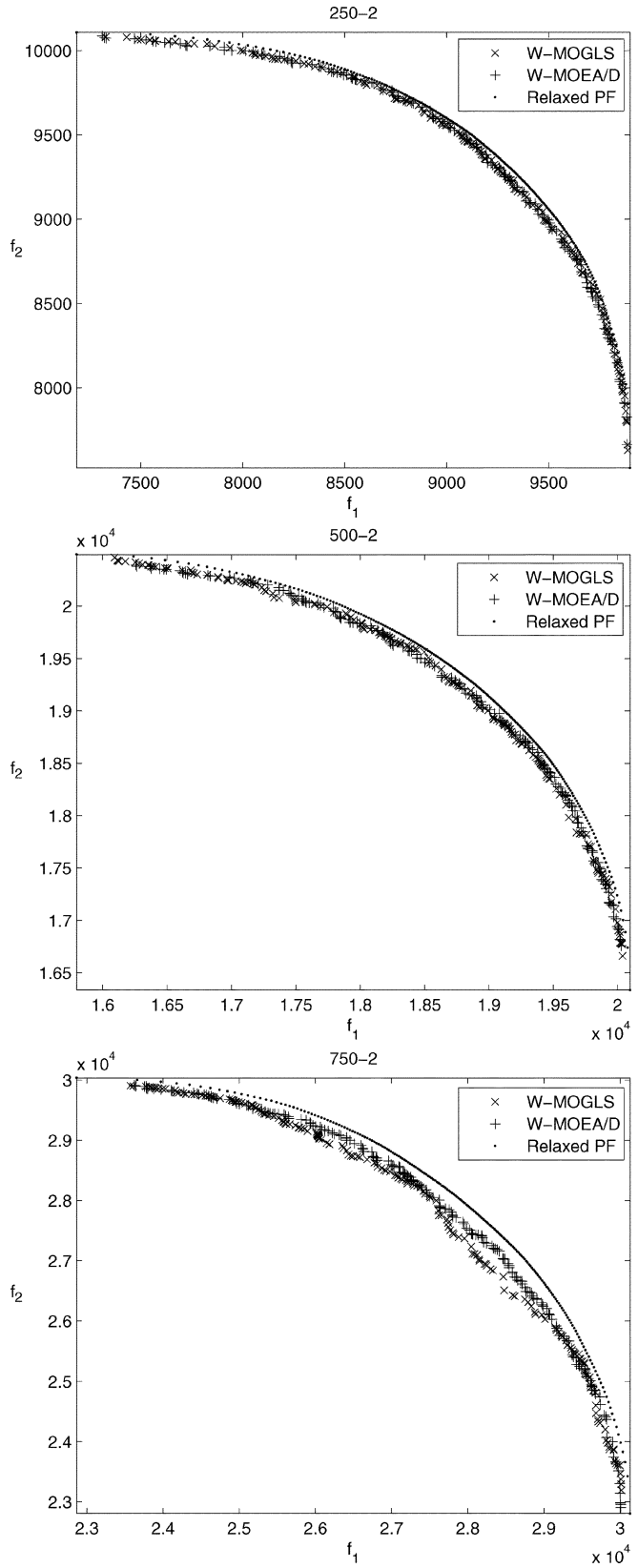

Fig. 4. Plots of the nondominated solutions with the lowest D-metric in 30 runs of MOEA/D and MOGLS with the weighted sum approach for all the 2-objective MOKP test instances.

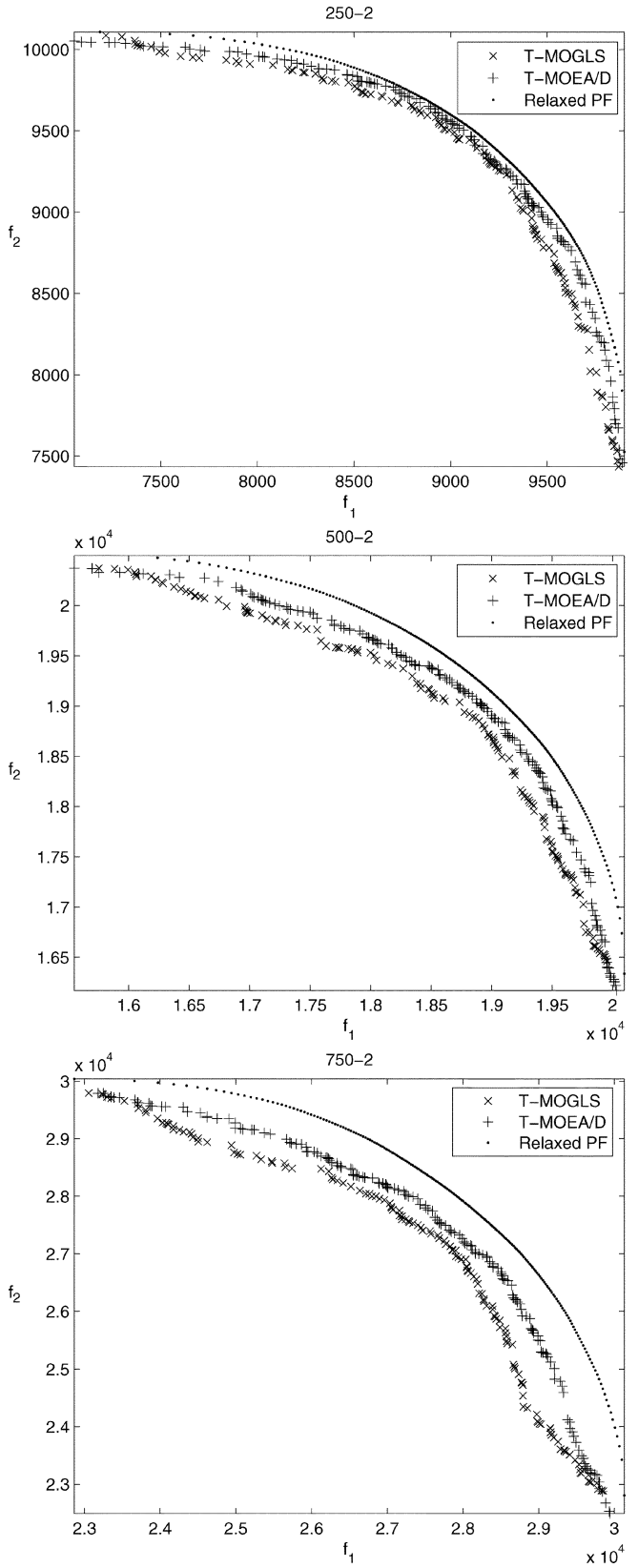

Fig. 5. Plots of the nondominated solutions with the lowest D-metric in 30 runs of MOEA/D and MOGLS with the Tchebycheff approach for all the 2-objective MOKP test instances.

Table VII presents the mean and standard deviation of D-metric value of the final solutions obtained by each algorithm for each test instance. This table reveals that in terms of D-metric, the final solutions obtained by MOEA/D is better than NSGA-II for ZDT4, ZDT6 and the two 3-objective instances, and slightly worse for other three instances.

TABLE V
AVERAGE CPU TIME (IN SECONDS) USED BY NSGA-II AND MOEA/D WITH THE TCHEBYCHEFF APPROACH

		NSGA-II	MOEA/D
Instance	ZDT1	1.03	0.60
	ZDT2	1.00	0.47
	ZDT3	1.03	0.57
	ZDT4	0.77	0.33
	ZDT6	0.73	0.27
	DTLZ1	10.27	1.20
	DTLZ2	8.37	1.10

TABLE VI
AVERAGE SET COVERAGE BETWEEN MOEA/D WITH THE TCHEBYCHEFF APPROACH (A) AND NSGA-II (B)

		$C(A,B)$	$C(B,A)$
Instance	ZDT1	15.88	1.64
	ZDT2	15.53	5.37
	ZDT3	14.61	2.92
	ZDT4	10.74	23.09
	ZDT6	99.56	0
	DTLZ1	7.84	0.49
	DTLZ2	9.91	0.00

TABLE VII
D-METRIC VALUES OF THE SOLUTIONS FOUND BY MOEA/D WITH THE TCHEBYCHEFF APPROACH AND NSGA-II. THE NUMBERS IN PARENTHESES REPRESENT THE STANDARD DEVIATION

		NSGA-II	MOEA/D
Instance	ZDT1	0.0050 (0.0002)	0.0055 (0.0039)
	ZDT2	0.0049 (0.0002)	0.0079 (0.0109)
	ZDT3	0.0065 (0.0054)	0.0143 (0.0091)
	ZDT4	0.0182 (0.0237)	0.0076 (0.0023)
	ZDT6	0.0169 (0.0028)	0.0042 (0.0003)
	DTLZ1	0.0648 (0.1015)	0.0317 (0.0005)
	DTLZ2	0.0417 (0.0013)	0.0389 (0.0001)

Figs. 7 and 8 show in the objective space, the distribution of the final solutions obtained in the run with the lowest D-value of each algorithm for each test instance. It is evident that as to the uniformness of final solutions, MOEA/D is better than NSGA-II for ZDT1, ZDT2, ZDT6, and the two 3-objective test instances; it is about the same as NSGA-II for ZDT4, and worse than NSGA-II for ZDT3, on which MOEA/D found fewer solutions than NSGA-II in the two left segments of the PF. The poor performance of MOEA/D on ZDT3 could be attributed to the fact that the objectives in ZDT3 are disparately scaled.

We can conclude from the above results that MOEA/D with the Tchebycheff approach needs less CPU time than NSGA-II with the same number of function evaluations for these test instances. In terms of the solution quality, these two algorithms have about the same performance on the 2-objective test instances, but MOEA/D is better than NSGA-II on the two 3-objective instances.

G. A Bit More Effort on MOEA/D

In the above experiments, we used a very naive implementation of MOEA/D. The decomposition method is the classic Tchebycheff approach. We did not perform any objective normalization.

As mentioned in Section II-C, the Tchebycheff approach may perform worse, in terms of solution uniformness, than BI approaches, particularly when the number of objectives is more than two. It is also well-known that objective normalization is very useful for increasing the solution uniformness when the objectives are disparately scaled.

Revisiting Figs. 7 and 8, one may not be quite satisfied with the uniformness in the solutions found by the above implementation of MOEA/D for ZDT3, DTLZ1, and DTLZ2. Note that the two objectives in ZDT3 are disparately scaled, and DTLZ1 and DTLZ2 have three objectives, two questions naturally arise.

- Can MOEA/D with other advanced decomposition methods such as the PBI approach find more evenly distributed solutions for 3-objective test instances like DTLZ1 and 2?
- Can MOEA/D with objective normalization perform better in the case of disparately scaled objectives as in ZDT3?

We have performed some experiments to study these two issues. In the following, we report our experimental results.

1) MOEA/D With the PBI Approach: We have tested MOEA/D with the PBI approach on the two 3-objective test instances. In our experiment, the penalty factor in g^{bip} is set to be 5 and the other settings are exactly the same as in the implementation of MOEA/D with the Tchebycheff approach in the above subsection.

Table VIII compares the average D-metric values of NSGA-II and MOEA/D with two different decomposition approaches. Fig. 9 shows the distributions of the nondominated fronts with the lowest D-metric values in 30 independent runs of MOEA/D with the PBI approach for two 3-objective test instances. It is very clear from these results and Fig. 8 that MOEA/D with the PBI approach performs much better than NSGA-II and MOEA/D with the Tchebycheff approach on these two 3-objective instances. These results suggest that incorporating other advanced decomposition approaches into MOEA/D would be worthwhile studying in solving MOPs with more than two objectives. We would like to point out that MOEA/D with advanced decomposition approaches may require a bit more human effort, for example, to set the value of the penalty factor in the PBI approach.

2) Objective Normalization: We still use the Tchebycheff approach and do not change the parameter settings in the following experiments. However, we incorporate a simple objective normalization technique into MOEA/D to study if the performance can be improved in the case of disparately scaled objectives.

A lot of effort has been made on the issue of objective normalization in the communities of both mathematical programming and evolutionary computation [1], [35], [36]–[38]. A very simple normalization method is to replace objective f_i ($i = 1, \ldots, m$) by[10]

$$\bar{f}^i = \frac{f_i - z_i^*}{z_i^{\mathrm{nad}} - z_i^*} \tag{9}$$

where $z^* = (z_1^*, \ldots, z_m^*)^T$, as in Section II, is the reference point, $z^{\mathrm{nad}} = (z_1^{\mathrm{nad}}, \ldots, z_m^{\mathrm{nad}})^T$ is the nadir point in the

[10]Here, we assume that the goal of (1) is minimization.

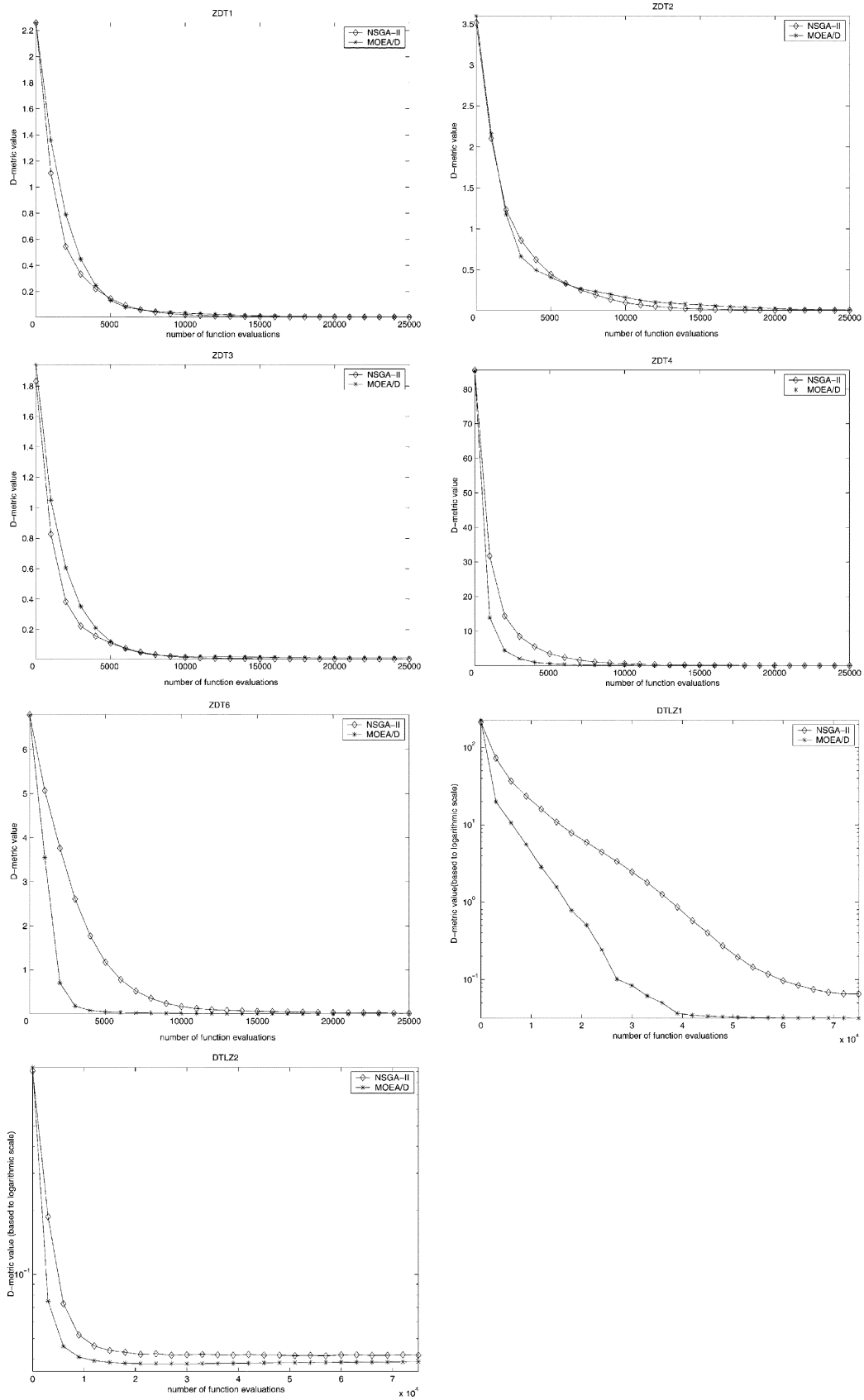

Fig. 6. The evolution of D-metric value in both NSGA-II and MOEA/D with the Tchebycheff approach for each test instance.

objective space, i.e., $z_i^{\mathrm{nad}} = \max\{f_i(x)|x \in PS\}$. In other words, z^{nad} defines the upper bound of PF, the Pareto front. In such a way, the range of each objective in the PF becomes [0, 1].

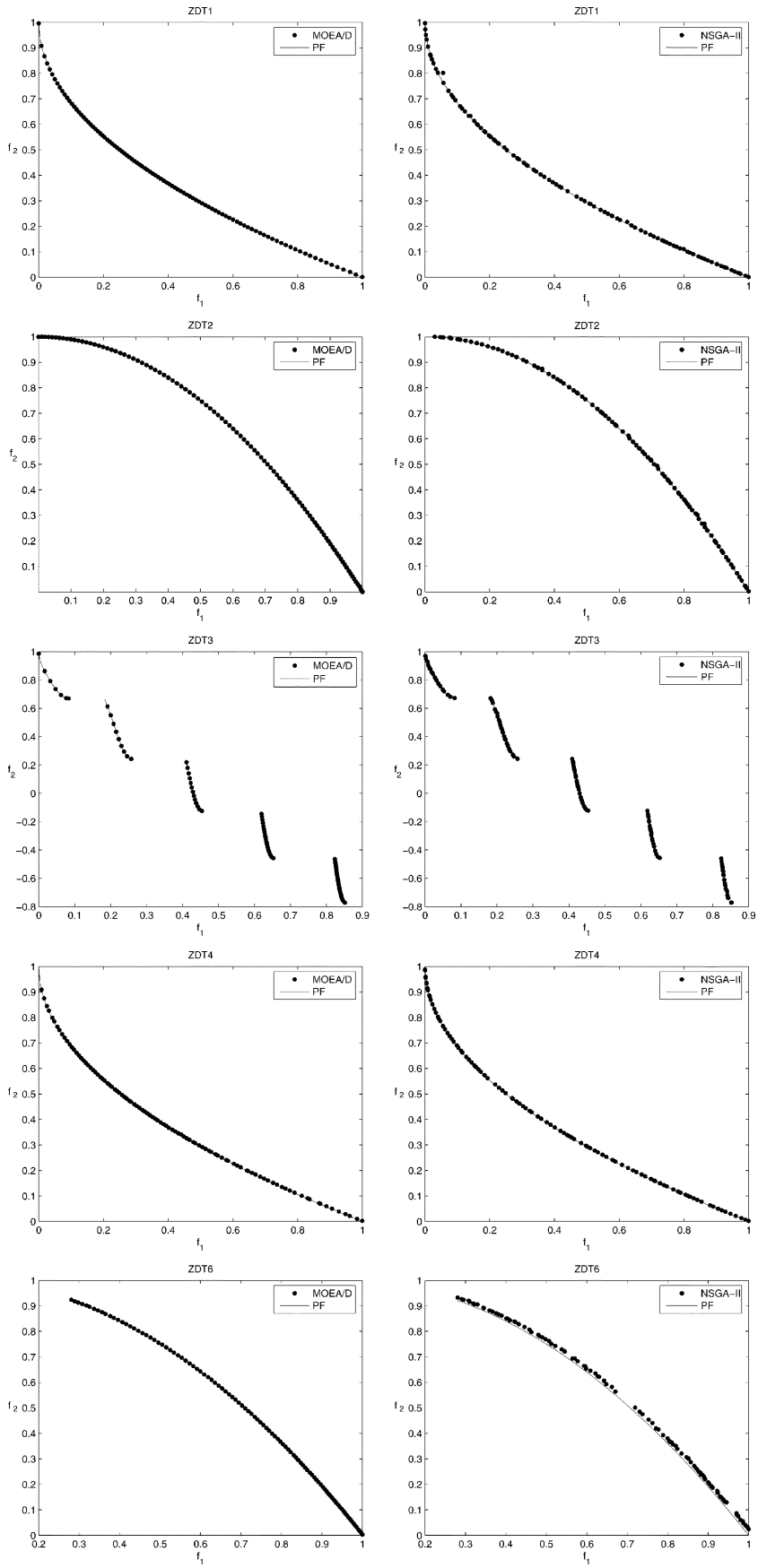

Fig. 7. Plot of the nondominated front with the lowest D-metric value found by NSGA-II and MOEA/D with the Tchebycheff approach for each 2-objective test instance. The left panel is for MOEA/D and the right panel for NSGA-II.

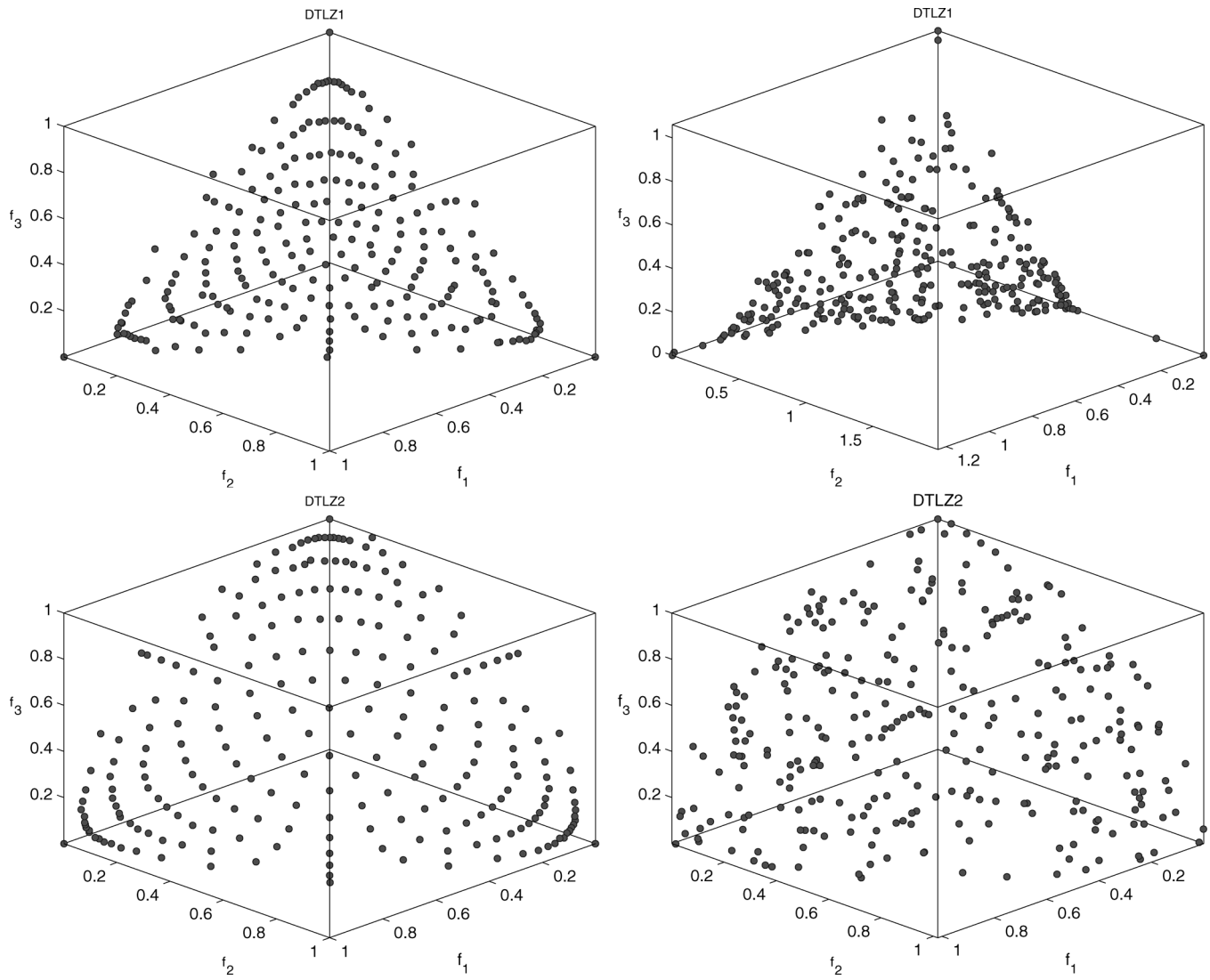

Fig. 8. Plot of the nondominated front with the lowest D-metric value found by NSGA-II and MOEA/D with the Tchebycheff approach for each 3-objective test instance. The left panel is for MOEA/D and the right panel for NSGA-II.

It is not easy or necessary to compute z^{nad} and z^* beforehand. In our implementation, we replace z^* by z,[11] and each z_i^{nad} by $\tilde{z}_i^{\mathrm{nad}}$, the largest value of f_i in the current population. Therefore, in Step 2.4 of MOEA/D with the Tchebycheff approach for continuous MOPs, g^{te} is replaced by

$$\max_{1\le i\le m}\left\{\lambda_i\left|\frac{f_i-z_i}{\tilde{z}_i^{\mathrm{nad}}-z_i}\right|\right\}. \tag{10}$$

We have tested MOEA/D with the Tchebycheff approach and the above normalization technique on ZDT3 and a modified version of ZDT1 in which f_2 is replaced by $10f_2$ such that the two objectives are disparately scaled. Fig. 10 plots the nondominated solutions obtained by a single run of MOEA/D with and without normalization for the modified version of ZDT1. Fig. 11 shows the nondominated solutions obtained by a single run of MOEA/D with normalization for ZDT3.

[11]Please refer to the description of MOEA/D for the details of initialization and update of z.

Figs. 10 and 11, together with the plots of the solutions found by NSGA-II and MOEA/D without normalization for ZDT3, clearly indicate that normalization does improve MOEA/D significantly, in terms of uniformness, in the case of disparately scaled objectives.

In summary, we have very positive answers to the two questions raised at the outset of this subsection. Using other sophisticated decomposition and normalization techniques in MOEA/D will be our future research topics.

VI. Scalability, Sensitivity, and Small Population in MOEA/D

A. Sensitivity of T in MOEA/D

T is a major control parameter in MOEA/D. To study the sensitivity of the performance to T in MOEA/D for both continuous and discrete MOPs, we have tested different settings of T in the implementation of MOEA/D with the weighted sum approach in Section IV for knapsack instance 250–2, and the implementation of MOEA/D with the Tchebycheff approach in

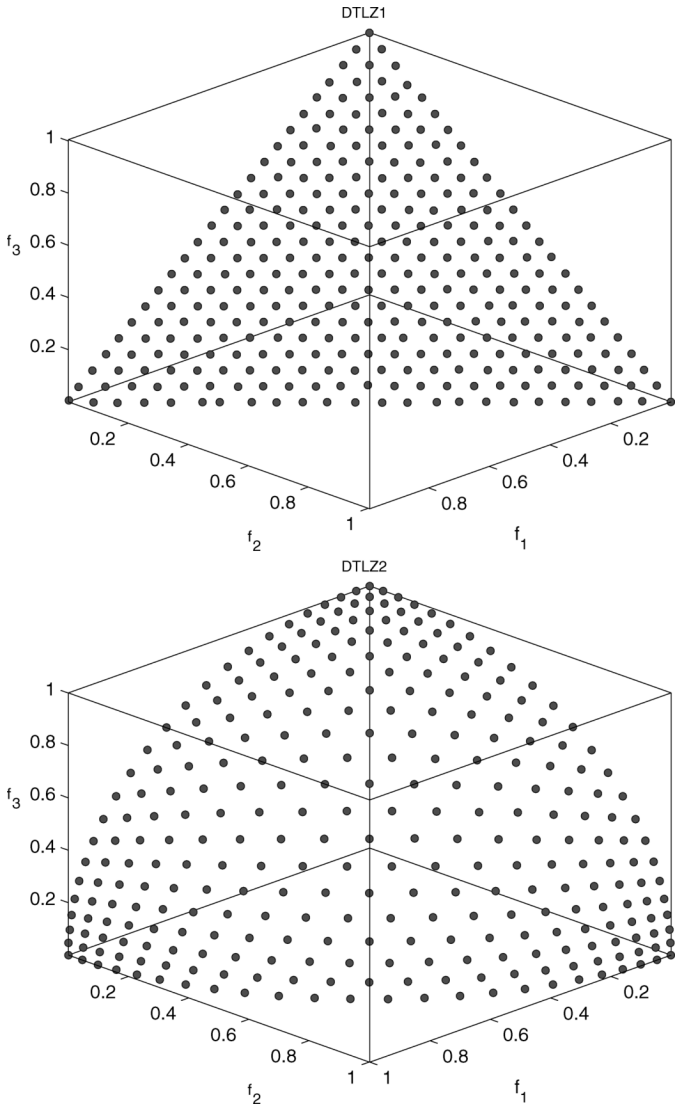

Fig. 9. Plot of the nondominated fronts with the lowest D-metric values in MOEA/D with the PBI approach for two 3-objective test instances.

TABLE VIII
COMPARISON OF AVERAGE D-METRIC VALUES OF THE SOLUTIONS FOUND BY NSGA-II, AND MOEA/D WITH THE TCHEBYCHEFF AND PBI APPROACHES. THE NUMBERS IN PARENTHESES REPRESENT THE STANDARD DEVIATION

Instance	NSGA-II	MOEA/D with Tchebycheff	MOEA/D with PBI
DTLZ1	0.0648 (0.1015)	0.0317 (0.0005)	0.0232 (0.0018)
DTLZ2	0.0417 (0.0013)	0.0389 (0.0001)	0.0280 (4.7034e-006)

Section V for ZDT1. All the parameter settings are the same as in Sections IV-E and V-E, except the settings of T. As clearly shown in Fig. 12, MOEA/D performs very well with T from 10 to 50 on knapsack instance 250–2, and it works very well for all the values of T except very small ones on ZDT1. Thus, we can claim that MOEA/D is not very sensitive to the setting of T, at least for MOPs that are somehow similar to these test instances.

Fig. 12 also reveals that MOEA/D does not work well on both instances when T is very small. This could be due to what was mentioned in Section III-B3, MOEA/D with too small T is poor at exploration. The fact that MOEA/D with large T works poorly on knapsack instance 250–2 could be explained by our

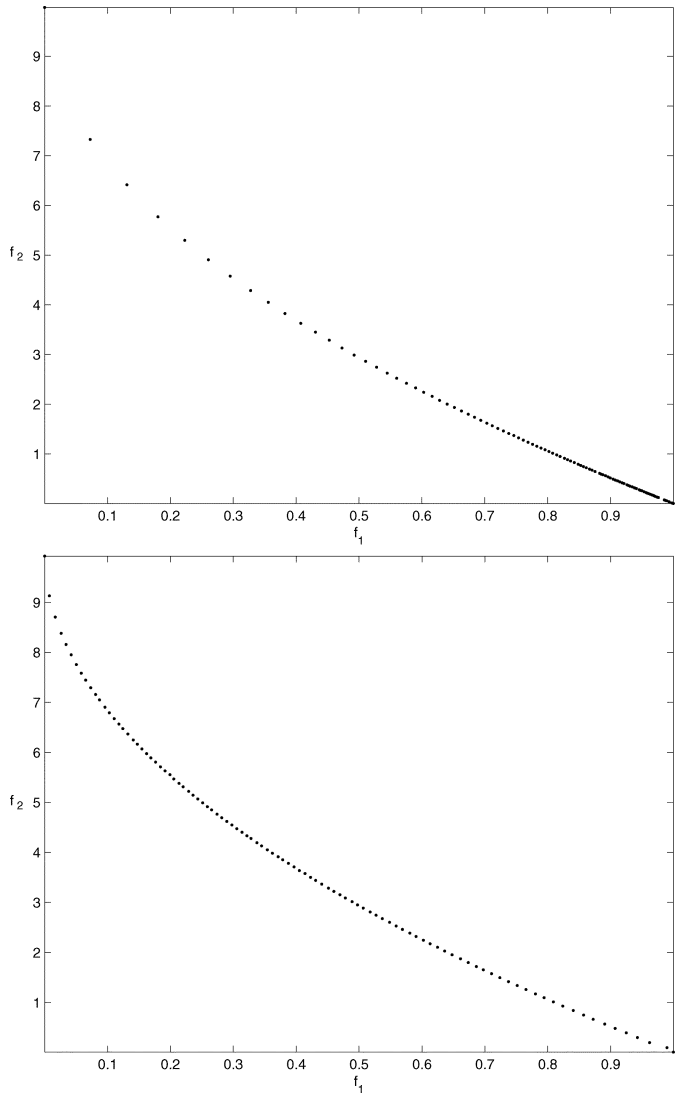

Fig. 10. Plot of the nondominated fronts found by MOEA/D with and without normalization for the modified version of ZDT1 in which f_2 is replaced by $10\,f_2$.

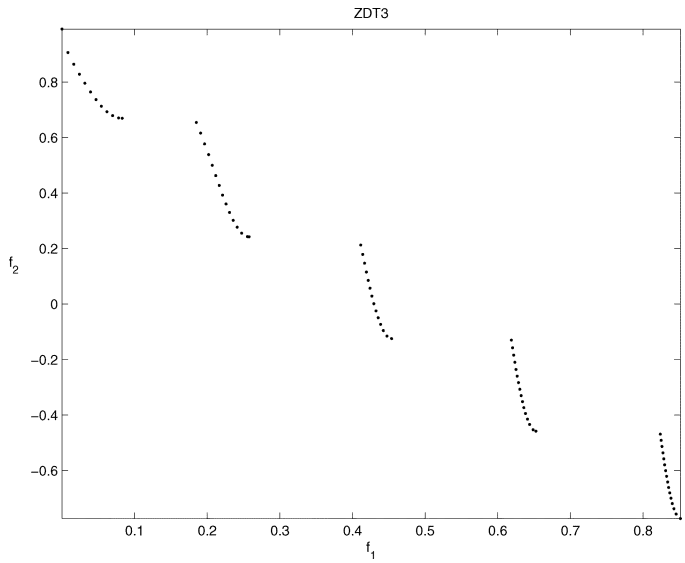

Fig. 11. Plot of the nondominated front found by MOEA/D with normalization for ZDT3.

analysis in Section III-B3 since the solutions to two subproblems with very different weight vectors are far different in this

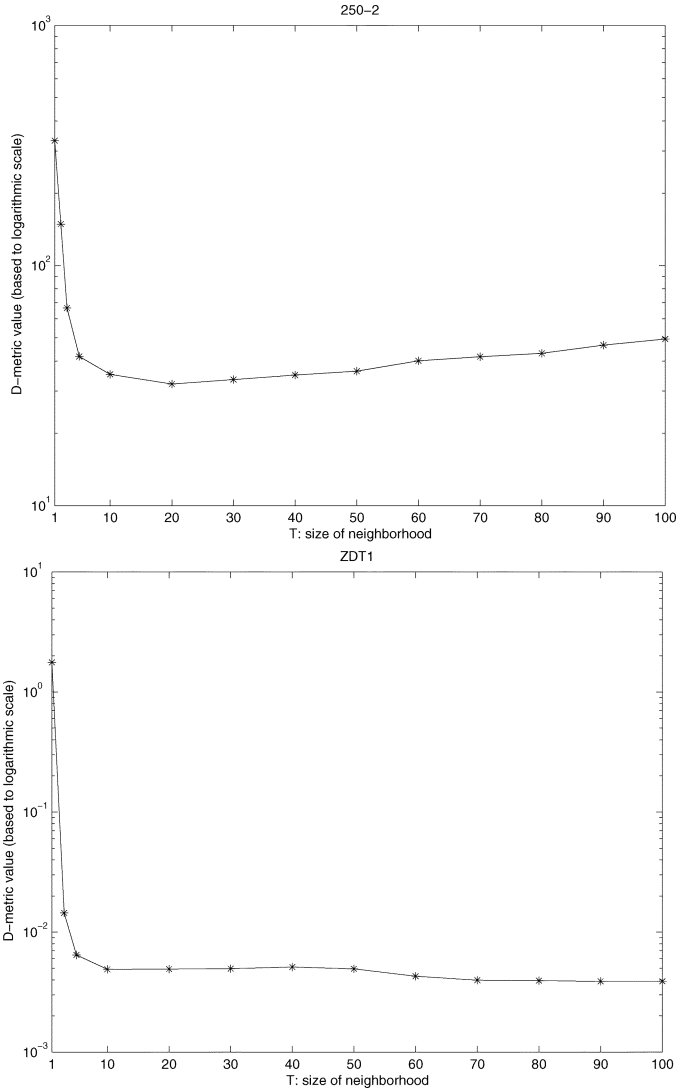

Fig. 12. The average D-metric value versus the value of T in MOEA/D for knapsack instance 2–250 and ZDT1.

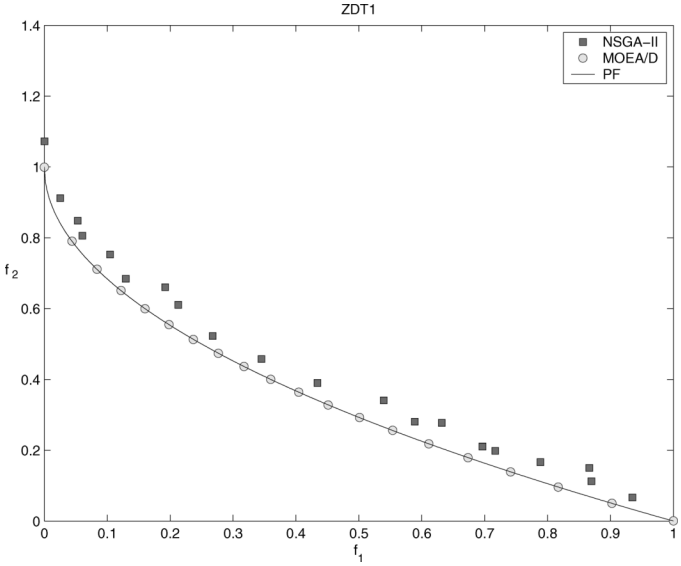

Fig. 13. Plot of the nondominated fronts found by MOEA/D and NSGA-II using small population ($N = 20$).

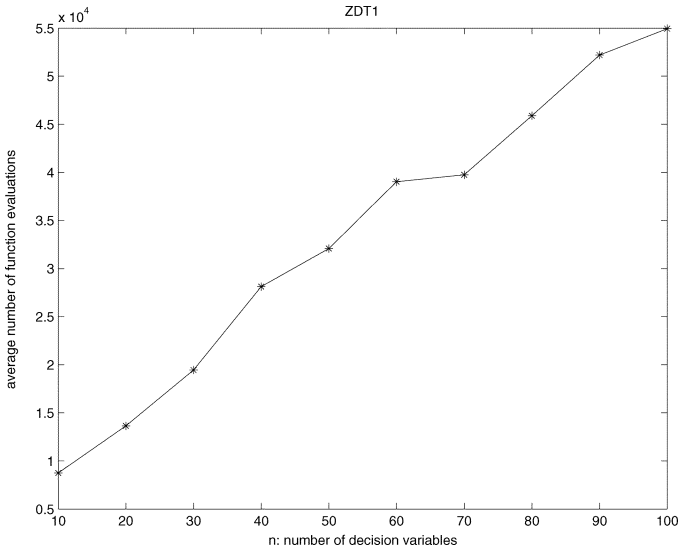

Fig. 14. The average numbers of function evaluations used for reducing the D-metric value under 0.005 for ZDT1 with different numbers of decision variables.

instance. The reason that MOEA/D with large T performs well on ZDT1 could be because even if two weight vectors are different, the optimal solutions to their associated subproblems are very similar, in fact, they are the same in all the components except the first one.

B. MOEA/D Using Small Population

As mentioned in Section I, a decision maker may not want to have a huge number of Pareto optimal solutions at high computational cost. They are often interested in obtaining a small number of evenly distributed solutions at low computational cost. In the following, we will show that MOEA/D using small population could serve this purpose. We take ZDT1 as an example and use MOEA/D with the Tchebycheff approach in Section V. All the parameter settings are the same as in Section V except the population size $N = 20$. For comparison, we also run NSGA-II with $N = 20$ on ZDT1. Both algorithms stop after 250 generations as in Section V.

Fig. 13 plots the final solutions obtained in a single run of NSGA-II and MOEA/D with $N = 20$. It is evident that MOEA/D found 20 very evenly distributed Pareto solutions, while NSGA-II failed in reaching the PF within the given number of generations. Clearly, this advantage of MOEA/D comes from its decomposition strategy.

C. Scalability

To study how the computational cost, in terms of the number of function evaluations, increase as the number of decision variable increases, we have tried, on ZDT1 with different numbers of decision variables, the variant of MOEA/D in Section V-C, with the same parameter settings except the maximal number of generation is 1000 (i.e., the maximal number of function evaluations is $100 \times 1{,}000 = 10^5$). We have found that in every run among 30 independent runs for each number of decision variables, MOEA/D has lowered the D-metric value blow 0.005, therefore the final solutions are a very good approximation to the PF by our experience in Section V. Fig. 14 gives the average

numbers of function evaluations used for reducing the D-metric value under 0.005 for ZDT1 with different numbers of decision variables. It is evident that the average number of function evaluations linearly scales up, as the number of decision variables increases.

The linear salability of MOEA/D in this test instance should be due to two facts: 1) The number of scalar optimization subproblems in MOEA/D is fixed to be 100, no matter how many the decision variables are. One does not need to increase the number of subproblems as the number of decision variables increases since the intrinsic dimensionality of a MOP is determined by its number of objectives, but not its number of decision variables [1], [39]. 2) The complexity of each scalar optimization could scale up linearly with the number of decision variables.

VII. Conclusion

Decomposition was widely used in traditional mathematical programming methods for solving MOPs. In contrast, most MOEAs treat a MOP as a whole and mainly rely on domination for measuring the solution quality during their search. These algorithms may not be very good at generating an even distribution of solutions along the PF.

This paper has proposed a simple and generic evolutionary multiobjective optimization algorithm based on decomposition, called MOEA/D. It first uses a decomposition method to decompose the MOP into a number of scalar optimization problems. Then, an EA is employed for optimizing these subproblems simultaneously. Each individual solution in the population of MOEA/D is associated with a subproblem. A neighborhood relationship among all the subproblems is defined based on the distances of their weight vectors. In MOEA/D, optimization of a subproblem uses the current information of its neighboring subproblems since two neighboring subproblems should have close optimal solutions. We have compared MOEA/D with MOGLS and NSGA-II on multiobjective knapsack problems and continuous multiobjective problems, respectively. Our analysis has shown that MOEA/D has lower computational complexity than MOGLS and NSGA-II, and the experimental results have also confirmed it. In terms of solution quality, MOEA/D with simple decomposition methods have outperformed or performed similarly to MOGLS and NSGA-II on most test instances.

We have shown that MOEA/D with the PBI approach is able to generate a very uniform distribution of representative Pareto optimal solutions on the PF on two continuous 3-objective test instances. We have also demonstrated that MOEA/D with a naive objective normalization technique can deal with disparately scaled objectives very well. These results suggest that more efficient and effective implementations of MOEA/D could be obtained if more effort is made.

We have experimentally investigated the scalability and the sensitivity to the neighborhood size T of MOEA/D. We have found that the computational cost linearly scales up with the number of decision variables, and MOEA/D is not very sensitive to the setting of T. We have also shown that MOEA/D is good at finding a small number of uniformly distributed Pareto solutions at low computational cost.

Combination of mathematical programming methods and EAs has been proven to be very successful in scalar optimization problems. Our work in this paper provides a very natural way for introducing decomposition strategies, which have been studied in the community of mathematical programming for a long time, into EAs for multiobjective optimization. New developments in decomposition strategies and other techniques in scalar optimization techniques can be readily integrated with EAs in the framework of MOEA/D.

Acknowledgment

The authors are grateful to X. Yao, the anonymous associate editor, and anonymous referees for their insightful comments. They also thank A. Zhou and S. Lucas for his help on this work.

References

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA: Kluwer, 1999.
[2] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. New York: Wiley, 2001.
[3] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA: Kluwer, 2002.
[4] K. Tan, E. Khor, and T. Lee, *Multiobjective Evolutionary Algorithms and Applications*, ser. Advanced Information and Knowledge Processing. Berlin, Germany: Springer-Verlag, 2005.
[5] E. Polak, "On the approximation of solutions to multiple criteria decision making problems," in *Multiple Criteria Decision Making*, M. Zeleny, Ed. Berlin, Germany: Springer-Verlag, 1976, vol. 123, Lecture Notes in Economics and Mathematical Systems, pp. 271–282.
[6] S. Helbig, "On a constructive approximation of the efficient outcomes in bicriterion vector optimization," *J. Global Optim.*, vol. 5, pp. 35–48, 1994.
[7] M. Wiecek, W. Chen, and J. Zhang, "Piecewise quadratic approximation of the non-dominated set for bi-criteria programs," *J. Multi-Criteria Decision Analysis*, vol. 10, no. 1, pp. 35–47, 2001.
[8] S. Ruzika and M. Wiecek, "Approximation methods in multiobjective programming," *J. Optim. Theory Appl.*, vol. 126, no. 3, pp. 473–501, Sep. 2005.
[9] I. Das and J. E. Dennis, "Normal-bounday intersection: A new method for generating Pareto optimal points in multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, Aug. 1998.
[10] A. Messac, A. Ismail-Yahaya, and C. Mattson, "The normalized normal constraint method for generating the Pareto frontier," *Struct Multidisc. Optim.*, vol. 25, pp. 86–98, 2003.
[11] C. A. Mattson, A. A. Mullur, and A. Messac, "Smart Pareto filter: Obtaining a minimal representation of multiobjective design space," *Eng. Optim.*, vol. 36, no. 6, pp. 721–740, 2004.
[12] C. A. C. Coello, "An updated survey of GA-based multiobjective optimization techniques," *ACM Comput. Surv.*, vol. 32, no. 2, pp. 109–143, 2000.
[13] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[14] J. D. Knowles and D. W. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation," in *Proc. Congr. Evol. Comput.*, Washington, D.C., Jul. 1999, pp. 98–105.
[15] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, Eds., Athens, Greece, pp. 95–100.
[16] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[17] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 325–343, Aug. 2003.

[18] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer, "Voronoi-based estimation of distribution algorithm for multi-objective optimization," in *Congress on Evolutionary Computation (CEC)*, Y. Shi, Ed. Piscataway, NJ: IEEE Press, 2004, pp. 1594–1602.
[19] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.
[20] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, 2000.
[21] J. D. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 100–116, Apr. 2003.
[22] Q. Zhang, A. Zhou, and Y. Jin, "Modelling the regularity in an estimation of distribution algorithm for continuous multiobjective optimisation with variable linkages," Dept. Comput. Sci., Univ. Essex, Colchester, U.K., Tech. Rep. CSM-459, 2006, (a revised version of this paper has been accepted for publication by the IEEE Trans. Evol. Comput., 2007).
[23] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.
[24] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.
[25] L. Paquete and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," in *Proc. Evol. Multi-Criterion Optim.*, 2003, pp. 479–493.
[26] E. J. Hughes, "Multiple single objective Pareto sampling," in *Proc. Congr. Evol. Comput.*, Canberra, Australia, 2003, pp. 2678–2684.
[27] Y. Jin, T. Okabe, and B. Sendhoff, "Adapting weighted aggregation for multiobjective evolutionary strategies," in *Evolutionary Multicriterion Optimization*. : Springer, 2001, vol. 1993, LNCS, pp. 96–110.
[28] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 392–403, Aug. 1998.
[29] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – A comparative experiment," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 402–412, Aug. 2002.
[30] Q. Zhang and Y. W. Leung, "Orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr.l 1999.
[31] N. J. W. T. J. Santers and W. I. Notz, *The Design of Analysis of Computer Experiments*. Berlin, Germany: Springer-Verlag, 2003.
[32] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.
[33] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[34] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. Congr. Evol. Comput.*, May 2002, vol. 1, pp. 825–830.
[35] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*. New York: Wiley, 1986.
[36] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*. New York: Wiley, 1997.
[37] P. J. Bentley and J. P. Wakefield, "Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms," in *Proc. 2nd On-Line World Conf. Soft Comput. Eng. Design Manuf.*, Jun. 1997, pp. 231–240.
[38] I. C. Parmee and D. Cvetkovic, "Preferences and their application in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 42–57, Feb. 2002.
[39] M. Ehrgott, *Multicriteria Optimization*, 2nd ed. Berlin, Germany: Springer-Verlag, 2005.
[40] T. Murata, H. Ishibuchi, and M. Gen, "Specification of Genetic Search Direction in Cellular Multiobjective Genetic Algorithm," in *Evolutionary Multicriterion Optimization*. Berlin, Germany: Springer-Verlag, 2001, LNCS1993, pp. 82–95.

Qingfu Zhang (M'01–SM'06) received the B.Sc. degree in mathematics from Shanxi University, Shanxi, China, in 1984, the M.Sc. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

He is currently a Reader in the Department of Computer Science, University of Essex, Colchester, U.K. Before joining the University of Essex in 2000, he was with the National Laboratory of Parallel Processing and Computing at the National University of Defence Science and Technology (China), Hong Kong Polytechnic University (Hong Kong), the German National Research Centre for Information Technology (now Fraunhofer–Gesellschaft, Germany) and the University of Manchester Institute of Science and Technology (U.K.). His main research areas are evolutionary computation, optimization, neural networks, data analysis and their applications.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B and a Guest Editor of a forthcoming Special Issue on Evolutionary Algorithms Based on Probabilistic Models in the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Hui Li received the B.Sc. and M.Sc degrees in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 1999 and 2002, respectively. He is currently working towards the Ph.D. degree in computer science at the University of Essex, Colchester, U.K.

His main research areas are evolutionary computation, multiobjective optimization, metaheuristics, and portfolio optimization.