A Primary Theoretical Study on Decomposition-Based Multiobjective Evolutionary Algorithms

Yuan-Long Li, *Student Member, IEEE*, Yu-Ren Zhou, Zhi-Hui Zhan, *Member, IEEE*, and Jun Zhang, *Senior Member, IEEE*

***Abstract*—Decomposition-based multiobjective evolutionary algorithms (MOEAs) have been studied a lot and have been widely and successfully used in practice. However, there are no related theoretical studies on this kind of MOEAs. In this paper, we theoretically analyze the MOEAs based on decomposition. First, we analyze the runtime complexity with two basic simple instances. In both cases the Pareto front have one-to-one map to the decomposed subproblems or not. Second, we analyze the runtime complexity on two difficult instances with bad neighborhood relations in fitness space or decision space. Our studies show that: 1) in certain cases, polynomial-sized evenly distributed weight parameters-based decomposition cannot map each point in a polynomial sized Pareto front to a subproblem; 2) an ideal serialized algorithm can be very efficient on some simple instances; 3) the standard MOEA based on decomposition can benefit a runtime cut of a constant fraction from its neighborhood coevolution scheme; and 4) the standard MOEA based on decomposition performs well on difficult instances because both the Pareto domination-based and the scalar subproblem-based search schemes are combined in a proper way.**

***Index Terms*—Decomposition-based multiobjective evolutionary algorithms (MOEAs), runtime analysis, theoretical study.**

I. Introduction

MULTIOBJECTIVE optimization has been a popular study topic in evolutionary computation. Recently, a vast number of multiobjective evolutionary algorithms (MOEAs) [1]–[3] have been developed to solve multiobjective optimization problems (MOPs). Among these algorithms, the NSGA-II proposed by Deb *et al*. [2] provides an essential way to organize and control a population to evolve with the Pareto domination relationship. On the other hand, in an MOEA based on decomposition (MOEA/D), Zhang and Li [3] apply a decomposition method to MOEA and solve MOPs by solving a series of scalar optimization problems. Numerous MOEAs have been developed with similar ideas to the two representative MOEAs mentioned above.

An interesting question emerges concerning the two different kinds of MOEAs: how do their different features affect the performances? They present two ways in dealing with MOPs: 1) the NSGA-II way—dealing with an MOP as a whole and 2) the MOEA/D way—dealing with an MOP by decomposition. Besides practical experiments, theoretical studies on these algorithms might be meaningful to reveal some important facts on their differences.

There are only a few theoretical studies on MOEAs currently, like runtime analysis of MOEAs in [4]–[7]. These studies are a straightforward extension of runtime analysis of single-objective optimization evolutionary algorithms (EAs). We notice that current theoretical studies on MOEAs are based on simplified MOEAs like simple evolutionary multiobjective optimizer (SEMO) [54], which deals with an MOP as a whole similar to NSGA-II does.

However, a theoretical analysis on MOEA/D that deals with MOP by decomposition is not available, which will be very useful in providing some insights. Thus in this paper, we present the primary theoretical study results on decomposition-based MOEAs for the first time. In this paper, we focus on analyzing the cases when MOEA/D is used to solve discrete combinatorial optimization problems.

Different discrete combinatorial MOP instances are studied here. On one hand, some simple MOP instances like the counting ones counting zeros (COCZ) and leading ones trailing zeros (LOTZ) problem are analyzed here. On the other hand, we present runtime analysis results on some more complicated MOP instances to analyze the efficiency of MOEA/D and to show how it can combine two different kinds of search schemes, i.e., the Pareto domination-based search scheme and the scalar subproblem-based search scheme.

Manuscript received September 4, 2014; revised January 2, 2015, April 7, 2015, July 23, 2015, and October 7, 2015; accepted November 2, 2015. Date of publication November 17, 2015; date of current version July 26, 2016. This work was partially supported by the National High-Technology Research and Development Program (863 Program) of China under Grant 2013AA01A212; by the National Natural Science Foundations of China (NSFC) under Grants 61402545, 61202130, and 61170081; the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306038; the Project for Pearl River New Star in Science and Technology under Grant 201506010047; the NSFC Key Program under Grant 61332002; the NSFC for Distinguished Young Scholars under Grant 61125205; and by the NSFC Joint Fund with Guangdong under Key Projects U1201258 and U1135005. *(Corresponding author: Zhi-Hui Zhan.)*

The authors are with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Guangzhou 510006, China, and also with the Key Laboratory of Software Technology, Education Department of Guangdong Province, Guangzhou 510006, China (e-mail: zhanapollo@163.com; junzhang@ieee.org).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2015.2501315

The detailed contributions of this paper are summarized as follows.

1) We prove that in certain cases, polynomial-sized evenly distributed weight parameters-based decomposition cannot map each point in a polynomial sized Pareto front to a subproblem.
2) We propose a simple ideal serialized MOEA/D in order to show in an ideal case how fast the algorithm can be when it is used to solve some simple MOP instances like COCZ and LOTZ.
3) We present runtime analysis results on standard parallel MOEA/D to show its time efficiency on simple MOP instances and show that its neighborhood coevolution scheme can help to reduce the runtime at least by a constant fraction. That is, 50% of the time can be reduced to cover the whole Pareto front after the optimal reference point is found when solving the analyzed examples.
4) We also present runtime analysis results on standard MOEA/D to show how it can benefit from incorporating Pareto domination-based and scalar subproblem-based search schemes together. We analyze MOP instances for which MOEAs only using Pareto domination-based or scalar subproblem-based search scheme will fail while MOEA/D can perform well with both kinds of search schemes properly combined.

The remainder of this paper is organized as follows. In Section II, the recent progresses of theoretical study on EAs are introduced and a brief summary of the MOEA studies is provided. In Section III, the algorithms to be analyzed in the paper are described in detail. In Section IV, runtime analysis results on two simple MOP instances are provided. In Section V, runtime complexity on more complex MOP instances with deceptive and plateau objectives is analyzed. In Section VI, we conclude this paper.

II. Recent Progress of Theoretical Studies on EAs and MOEAs

A. Recent Progress of Theoretical Studies on EAs

Several different methods are widely used in the theoretical studies of EAs [8]. Fitness partition [51], [52] method is the very basic method, in which the objective fitness domain is partitioned into several levels. By considering the expected time the algorithm spent at each level, the total expected time for an algorithm to solve the problem can be computed. A basic theorem on fitness partition method is as follows and will be used in the analysis below. Without losing of generality, maximization problem is considered here.

Theorem 1. Consider a maximization problem with fitness function $f : D \to \mathbf{R}$. Suppose D can be partitioned into L subsets and for any solution x_i in subset i and x_j in subset j, we have $f(x_i) < f(x_j)$ if $i < j$. In addition, subset L only contains optimal solution(s) of f. If an algorithm can jump from subset k to subset $k+1$ or higher with probability at least p_k, then the total expected time for the algorithm to find the optimum satisfies

$$E(T) \leq \sum_{k=1}^{L-1} \frac{1}{p_k}. \tag{1}$$

Drift analysis [9], [49] is also a widely used tool to analyze EAs. By considering the expected progress achieved in a single step at each search point, drift analysis can be used to analyze problems which are hard to be analyzed by fitness partition. Drift analysis has been improved in different ways such as in [10], [47] and [50] and has become a simple yet powerful tool to analyze runtime complexity of EAs.

Another useful method is to use Markov chain model. The evolution process of an EA can be modeled by a Markov chain, as usually the generation of the new population is only decided by the current population. As the absorbing time of a Markov model can be computed when the transition matrix is known, Markov model can be used in the runtime analysis of EAs [11]. In recent works, Yu *et al.* [56] and Yu and Qian [57] proposed the "switch analysis" method, which studies the "aligned mapping" between two Markov chains.

With the above analysis methodology studies, lots of efforts have been made in using these methods to examine the runtime complexity of simple EAs on different problems. For example, runtime analysis results of simple EAs on pseudo Boolean functions [12] and various typical combinatorial optimization problems such as minimum spanning tree [53], single-source shortest path [13], traveling salesman problem [14], makespan scheduling [15], and minimum set cover problem [16] have been reported. Noisy problems are studied in [17], [58]. In [60], the hardness of fitness functions respect to EAs is analyzed. Most of the above results are based on $(1+1)$EA, in which only one individual is used in evolution. There are also results reported on EAs with multiple individuals [46]. Analyzing the effects of different evolutionary operators has been done, where Lehre and Yao [18] studied the mutation-selection balance problem of EAs. Besides the usual runtime complexity studies on EAs, in a recent study, Jansen and Zarges [19] proposed to analyze the performance of EAs with a fixed budget, which provides clues on how to set the termination conditions of EAs in practice. A review of theoretical studies on EAs for discrete optimization problems can be found in [48]. There are also studies focusing on analyzing the runtime of EAs for continuous optimization [43]–[45], which aim to study the convergence rate of the algorithms. In this paper, we will only focus on the case of discrete optimization.

As for theoretical studies on MOEAs, there are several results such as [4]–[7]. Laumanns *et al.* [4] analyzed the runtime of simple SEMO on pseudo Boolean functions and provided the basic runtime analysis results on MOEAs. Qian *et al.* [5] studied how crossover can be used to improve the efficiency of MOEAs. In [6], the runtime complexity of various MOPs is studied with several different MOEAs. In [7], a detailed study on set-based MOEAs with the cyclic problem is provided. However, as we mentioned before, most of these works focus on the runtime complexity analysis of Pareto domination-based MOEAs; no studies have been done on decomposition-based MOEAs like MOEA/D, to our best knowledge.

B. Recent Progress on MOEAs

In a typical MOP, several different objectives need to be optimized together. Without losing of generality, we assume that each objective is supposed to be maximized. For an MOP with objectives $f_1, f_2, \ldots, f_m$, given a solution x we have a fitness vector $(f_1(x), \ldots, f_m(x))$. Due to that different objectives usually cannot be maximized at the same time, a Pareto domination relation is used to indicate the quality of a solution. If a solution x' satisfies $f_i(x') \geq f_i(x)$ for $i = 1, \ldots, m$ and for at least one index i it satisfies $f_i(x') > f_i(x)$, x' dominates x. With this relation the optimal solutions of an MOP compose a set called Pareto optimal set which contains all solutions that cannot be dominated by any other solutions, and their fitness vectors compose the Pareto front. A large number of MOEAs have been proposed to solve MOPs. Below we present a brief review on the recent progresses of MOEAs.

Because of the Pareto domination-based selection mechanism, diversity problem has been a significant problem for MOEAs and has been studied a lot. For example, in [20] a territory around each individual to prevent crowding is proposed. In [21], an MOEA with binary space partitioning tree-based density estimation is designed, which can be used to solve MOPs with strictly disconnected Pareto fronts.

Many MOEAs have been proposed recently in terms of different concerns. For example, in [22] multiple populations are used to solve MOP. In [23], dynamic population size setting is shown to be helpful. In [61], online diversity assessment is proposed for MOEA. Hybrid MOEA is investigated in [25]. Interactive MOEA is studied in [26] which introduces some supervised preference information to guide the search. MOEAs have also been studied in other optimization situations such as constrained optimization [27] and dynamic optimization [28]. Recently, many-objective (more than three objectives) MOEAs have been studied from various aspects [24], [63]–[66].

MOEAs are widely used in practical applications, such as power dispatch [29], filter design [30], and association rules mining [31]. A survey on applying MOEAs to data mining can be found in [32] and [59]. Applications of MOEAs on finance and economic problems are summarized in [33].

C. Introduction to MOEA/D

In MOEA/D, a series of scalar optimization problems are constructed from an MOP through the Tchebycheff approach [34]. Considering an MOP with m objectives to be maximized, the Tchebycheff scalar optimization subproblem is defined as

$$\min \max_{1 \leq i \leq m} \{w_i | f_i(x) - z_i*|\} \tag{2}$$

where w_i is a weight parameter for the ith objective; Denote $z* = (z_1*, \ldots, z_m*)$ is a reference point where z_i* is set to the current best fitness of the ith objective [3]. In MOEA/D, the original MOP is decomposed into many subproblems in form of (2).

In the following, we only study the case when $m = 2$, i.e., bi-objectives, which is common in the studies and applications of MOEAs. In this case, the kth subproblem $F_k(x)(k = 0, \ldots, H)$ in MOEA/D can be written as

$$F_k(x) = \max\left\{\lambda^k | f_1(x) - z_1*|, \left(1 - \lambda^k\right) | f_2(x) - z_2*|\right\} \tag{3}$$

where $(l^k, (1 - l^k))$ is the weight vector of the kth subproblem and satisfies $l^k = k/H$. There are $H + 1$ scalar problems to be solved after the decomposition.

The key features of MOEA/D are as follows.

1) Tchebycheff approach used in MOEA/D can decompose an MOP into several scalar optimization problems, which can be solved with any optimization method for single objective optimization.
2) Neighborhood-based coevolution is used in MOEA/D. The neighborhood relationship based on the decomposition method mentioned above can be easily defined and improves the search efficiency because similar subproblems usually have similar optima.

MOEA/D has been extensively investigated from various aspects. Li and Zhang [35] and Ishibuchi *et al.* [36] studied the neighborhood relationship used in MOEA/D and proposed different definitions of neighborhoods. Zhao *et al.* [37] proposed using ensemble neighbor sizes. Zhang *et al.* [38] proposed using dynamical resource allocation methods to balance the computational resources spent for different subproblems. Adaptive scalarizing MOEA/D is studied in [39] and [40]. In [41], MOEA/D is improved in solving expensive MOPs. In [62], an external archive is introduced into MOEA/D to solve combinatorial optimization problems with better efficiency.

Despite that MOEA/D has been improved in various ways in the literatures, we find that no theoretical studies have been done on MOEA/D. In the following sections, we will do this paper for the first time.

III. Algorithms to be Analyzed

In this section, we introduce all the algorithms that will be analyzed in the following sections. In this paper, we will mainly focus on the standard MOEA/D algorithm. To do a comparison study with other single individual-based simple MOEAs, we also introduce a simplified MOEA/D. The basic MOEA-SEMO will be compared here as an MOEA without decomposition. Detailed operations of these algorithms are shown below.

Algorithm 1 shows the standard MOEA/D we used in the following analysis. The algorithm uses simple mutation method (like the other algorithms to be compared) to generate new individuals to simplify the analysis. The standard MOEA/D is hard to analyze theoretically, because the population-based random iteration procedure is hard to analyze. So, we propose a simplified version named seri-MOEA/D as shown in Algorithm 2. For seri-MOEA/D, in each generation only one individual is generated and only one subproblem is targeted.

The design of seri-MOEA/D is to create a kind of simple MOEA like SEMO, but still keeps the key features of

Algorithm 1 MOEA/D for Bi-Objective Problems

1: *Input:* an MOP, stopping criterion, the number of subproblems considered in MOEA/D N=1+H. Weight vectors λ, T (neighbor size).
2: *Initialization:* Pareto optimal set P=$\varnothing$. For each subproblem, find the T closest subproblems according to the Euclidean distance of their weight vectors. These subproblems compose the neighbor of the subproblem, which is denoted by B_k for the kth subproblem (k=0,...,H). Initialize the reference point $z*$. Initialize a population X=$\{x_0,\ldots,x_H\}$ and compute the fitness vector $(f_1(x_k), f_2(x_k))$ and the subproblem fitness $F_k(x_k)$ for k=0,...,H.
3: **while** stop criterion is not satisfied **do**
4: **for** each subproblem k=0,...,H, **do**
5: *Reproduction and improvement:* Generate new solution for the kth subproblem by mutating individual x_k to generate a candidate as x_k'.
6: *Update each solution in* B_k with x_k': for each solution x_j in B_k, if $F_j(x_k')\leq F_j(x_j)$, update $x_j = x_k$' and its fitness records.
7: *Update* z*: if $f_i(x_k')>z_i*$, set $z_i*=f_i(x_k')$, i=1,2.
8: *Update P:* remove any solutions dominated by x_k'. If no solution in P can dominate x_k', add x_k' into P.
9: **end for**
10: **end while**
11: **Output**: P (Pareto optimal set).

Algorithm 2 Seri-MOEA/D for Bi-Objective Problems

1: *Input:* an MOP, stopping criterion, the number of subproblems considered in MOEA/D N=1+H. Weight parameters λ^k: always set λ^0=0 and λ^H=1.
2: *Initialization:* Use the (1+1)EA to find the optimal solution x of objective f_2, which is the solution of the 0th subproblem because λ^0=0; x is used as the search start point and it is used to initialize the Pareto optimal set P=$\{x\}$. Set the subproblem index k=1.
3: **while** $k\leq H$ **do**
4: *Reproduction and improvement:* Generate a new solution x' by mutating x.
5: *Try to minimize current subproblem:*
If $F_k(x')\leq F_k(x)$, update x=x'.
6: *Try to move-on to the next subproblem:* If x is the optimal solution of $F_k(x)$, update k=k+1;
7: *Update P:* Remove any solutions dominated by x'. If no solution in P can dominate x', and x' *has different fitness vector to the solutions in P*, then add x' into P. If x' has the same fitness vector with a solution in P, replace that solution with x'.
8: **end while**
9: **Output:** P (Pareto optimal set).

Algorithm 3 SEMO

1: *Initialization:* Pareto optimal set P=$\varnothing$. Randomly generate an individual x and add x into P.
2: **while** stop criterion is not satisfied, **do:**
3: Uniformly select a solution x from P at random.
4: *Mutate* x with 1-bit or bitwise mutation operator to generate a candidate solution x'.
5: **if** x' cannot be dominated by any solution in P && *has different fitness vector to the solutions in P* **do**
6: Add x' into P and eliminate any solutions in P that can be dominated by x'.
7: **end if**
8: **if** x' and a solution in P have the same fitness vector **do**
9: Replace that solution with x'.
10: **end if**
11: **end while**
12: **Output** P.

Algorithm 4 SEOF Initialization Method

1: Solve each objective with (1+1)EA, stop only when the optimum is found.
2: Use the optimal solutions of all objectives to initialize P (Pareto optimal set).

Algorithm 5 (1 + 1)EA

1: *Initialization:* A random individual x.
2: **while** x is not optimal **do**
3: Use local or bitwise mutation to generate x'.
4: **if** x' has better or same fitness than x, **do**
5: replace x with x'.
6: **end if**
7: **end while**
8: **Output**: x.

standard MOEA/D. In seri-MOEA/D, the decomposition feature of MOEA/D is retained. Moreover, due to the fact that the solution of the current subproblem is used to solve the next subproblem, the neighborhood coevolution feature of MOEA/D is also retained (though in an ordered way) in seri-MOEA/D. As seri-MOEA/D is a rather ideal algorithm, we will focus on the standard MOEA/D but present some runtime bound results on seri-MOEA/D to make a comparison with other single individual-based simple MOEAs.

To compare the runtime bounds, the SEMO [54] used in the subsequent analysis is also introduced here as shown in Algorithm 3. A special initialization method solve-each-objective-first (SEOF) shown in Algorithm 4 can be used instead of the initialization procedure (line 1) of Algorithm 3, which can be useful as shown in the following analysis. Simple (1 + 1)EA used in SEOF is described in Algorithm 5.

Note that a Pareto optimal set updating condition—"with different fitness vector to the solutions in P"—is used in all the above algorithms. As there can be many solutions with the same fitness vector, such setting can avoid adding solutions corresponding to the same point in the Pareto front into the Pareto optimal set. With this restriction the algorithm can focus on finding new solutions to cover the whole Pareto front.

Simple mutation operators are used in the above algorithms to generate new solutions. The 1-bit and bitwise mutation operators used in MOEA/D, seri-MOEA/D, and SEMO are as follows: for a solution with n bits, 1-bit mutation uniformly randomly selects a bit of x and flips it; bitwise mutation flips each bit in x with a probability $1/n$. In the following analysis, 1-bit mutation is used in most cases except when bitwise mutation must be used to solve a problem. The reason is that 1-bit mutation is easy to analyze; however, for some cases, bitwise mutation should be used so that the solution can change multiple bits at one time, which can be critical when a better solution can only be found in several bits of changes away.

Note that the seri-MOEA/D, which is a serialized and simplified version of MOEA/D to ease the runtime analysis, turns out to be a bit similar to the greedy evolutionary multiobjective optimizer (GEMO) [4], which is designed to improve the efficiency of SEMO. Different from SEMO, GEMO uses the Pareto optimal solution most recently found as the parent to generate the offspring. This way, GEMO benefits from neighborhood relations of Pareto optimal solutions like seri-MOEA/D does. However, seri-MOEA/D is different in that all optimal solutions are found in order, while for GEMO, the first Pareto optimal solution can be anyone and GEMO restarts its search in certain cases which can affect its efficiency (see the simulation results in the next section). Also, it should be noted that the proposed seri-MOEA/D is an implementation of the two phase local search [55].

IV. Runtime Analysis on Two Simple MOP Instances

In this section, we present runtime analysis results of MOEA/D on two simple MOP instances with different decomposition situations. The Tchebycheff approach used in MOEA/D decomposes an MOP into a series of single-objective optimization problems. As suggested in [3], for a bi-objective optimization problem, the weights for each objective indicated by λ^k and $(1-\lambda^k)$ are set to be evenly distributed in [0,1], i.e., $\lambda^k = k/H, k = 0, \ldots, H$, where $H+1$ is the number of subproblems after decomposition. A question then comes out whether these subproblems can match the Pareto front well such that every point in the Pareto front can be found. The following runtime analysis answers this question to a certain extent. In the following analysis, for the standard MOEA/D, H is set to be the problem scale n. For the seri-MOEA/D, we assume we can have a good estimation of the optimal fitness value of each objective function so we simply set the reference point z^* to be optimal fitness value of each objective. For each MOP instance, we first present results on simple MOEAs like seri-MOEA/D and SEMO, then we show results on standard MOEA/D.

A. Case With One-to-One Match

One interesting case is that the subproblems are mapping one-to-one to the points in the Pareto front. The following simple bi-objective pseudo Boolean problem COCZ [4] is used here:

$$f_1(x) = ||x||_0 \tag{4}$$

$$f_2(x) = n - ||x||_0 \tag{5}$$

where $||x||_0$ is the number of 1-bits in $x = (x_1, x_2, \ldots, x_n)$ with $x_i \in \{0, 1\}, i = 1, \ldots, n$. In the above COCZ problem, the two objectives contradict with each other and a simple relationship can be observed

$$f_1(x) + f_2(x) = n. \tag{6}$$

Note that the COCZ problem defined above is slightly different from [4] which adds restriction to avoid every solution to be Pareto optimal. As we are only interested in the time spent to cover the whole Pareto front, the above definition is nontrivial and more suitable to be an example of one-to-one match. Let $H = n$, the MOEA/D (seri-MOEA/D) algorithm decomposes the COCZ problem into $n+1$ subproblems and we have the following lemma.

Lemma 1: After the optima of both objectives are found and the reference point z^* has been updated accordingly, the optimum of each subproblem decomposed by MOEA/D (seri-MOEA/D) is a Pareto optimal solution of the COCZ problem defined by (4) and (5) and vice versa.

Proof: After the optima of both objectives are found and the reference point z^* has been updated accordingly, the kth subproblem to minimize is

$$F_k(x) = \max\left\{\lambda^k |f_1(x) - n|, \left(1-\lambda^k\right)|f_2(x) - n|\right\} \tag{7}$$

where $\lambda^k = k/n$.

As $f_2(x) = n - f_1(x)$, to simplify $F_k(x)$, let $\Delta(x)$ be the difference between the two items [the two functions to be judged which is larger, as in (7)] in $F_k(x)$, that is

$$\begin{aligned}\Delta(x) &= \lambda^k(n - f_1(x)) - \left(1-\lambda^k\right)|n - f_1(x) - n| \\ &= \lambda^k n - f_1(x).\end{aligned} \tag{8}$$

With $\lambda^k = k/n$ we have

$$\Delta(x) = k - f_1(x). \tag{9}$$

It is easy to see that when $f_1(x)$ increases from 0 to n, the first item of $F_k(x)$ is decreasing and the second item is increasing. To get the larger item of the two items in $F_k(x)$, we can just compute the cross point of the two items.

Set $\Delta(x) = 0$, we have that the cross point satisfies $f_1(x) = k$, which means if $f_1(x) < k$, the first item is larger and if $f_1(x) > k$, the second item is larger. Hence $F_k(x)$ can be simplified into the following piece-wisely defined function:

$$F_k(x) = \begin{cases} \lambda^k |f_1(x) - n|, & \text{if } f_1(x) \le k \\ (1-\lambda^k)|f_2(x) - n| & \text{if } f_1(x) > k. \end{cases} \tag{10}$$

From the definition of the two objective functions (4) and (5), it is obvious that when $f_1(x)$ falls in range $[0, k]$, $F_k(x)$ is a decreasing function; while in range $[k, n]$, $F_k(x)$ is an increasing function. Thus, the point with $f_1(x) = k$ minimize $F_k(x)$, i.e., the kth subproblem, and when $f_1(x) = k$, $(f_1, f_2) = (k, n-k)$, which is the kth point of the Pareto front of COCZ. As k varies from 0 to n, solutions of the subproblems cover the whole Pareto front of COCZ. ■

With the above lemma, we can now present some runtime bounds on the COCZ problem of different algorithms.

1) Runtime Bound Results on Simple MOEAs:

The following propositions show the runtime complexity of Seri-MOEA/D and SEMO on the COCZ problem.

Proposition 1: Seri-MOEA/D with 1-bit or bitwise mutation covers the whole Pareto front of the COCZ problem within an expected time $\Theta(n \log n)$.

To compare with seri-MOEA/D, the following result presents the expected optimization time of SEMO without/with the SEOF procedure to solve the COCZ problem.

Proposition 2: SEMO with 1-bit mutation and without SEOF procedure covers the whole Pareto front of the COCZ problem within expected time $\Theta(n^2 \log n)$.

Proposition 3: SEMO with 1-bit mutation and SEOF procedure covers the whole Pareto front of the COCZ problem within expected time $\Theta(n^2)$.

Due to page limitations, we only provide the main proof idea, the detailed proofs are provided in the supplementary material (http://www.ai.sysu.edu.cn/zhanzhh/materials/moead-ana.pdf). The key point to prove the runtime bound for seri-MOEA/D is to compute the runtime it needed to solve the $(k+1)$th subproblem based on the optimal solution of the kth subproblem ($k = 0, 1, \ldots, n-1$). The key point to prove the runtime bounds of the SEMO with/without SEOF is to study the probability of finding a new Pareto optimal solution based on current Pareto optimal set.

From the above propositions, we can see that seri-MOEA/D can run faster than SEMO. The reason is that seri-MOEA/D can solve each subproblem in an ordered way, which can be done very efficiently. Note that both seri-MOEA/D and SEMO with SEOF are ideal algorithms as they both need to find the optimum solution(s) of the objective function(s) in the initialization procedure.

2) Analysis on Standard MOEA/D: The above analysis shows the different runtime bounds of some simple MOEAs with only one individual in their population. Results show that the seri-MOEA/D can fun faster due to its neighborhood-based coevolution. However, as seri-MOEA/D is a rather ideal algorithm, the results can be different for the standard MOEA/D which runs with population with many individuals. Our analysis results prove that on COCZ, the standard MOEA/D can still benefit from neighborhood-based coevolution.

Proposition 4: For a standard MOEA/D with optimal decomposition as shown above with $n + 1$ subproblems and each subproblem is mapped to a different point in the Pareto front, its runtime on COCZ is upper bounded by $O(n^2 \log n)$.

Proof: First, we want to show that the MOEA/D can find the optima of both objectives in $O(n^2 \log n)$. It is clear that the 0th subproblem equals to the objective function $f_2(x)$ while the nth subproblem equals to the objective function $f_1(x)$. Both objective functions can be solved by an $(1 + 1)$EA with runtime bounds $\Theta(n \log n)$ as to solve them equals to solve the OneMax problem [6]. Note that the coevolution mechanism used in MOEA/D can only accelerate this process, so the MOEA/D can find the optimal reference point within $O(n^2 \log n)$ as the other $n - 1$ individuals are also searching at the same time.

Second, with the optimal reference point, subproblems are now defined by (10) and to cover the whole Pareto front we just need to solve each subproblem. Each subproblem equals to an OneMax problem and there are $n-1$ such subproblems to solve so the total runtime is also upper bounded by $O(n^2 \log n)$. Thus, overall the runtime of a standard MOEA/D to cover the whole Pareto front of the COCZ problem is upper bounded by $O(n^2 \log n)$. ■

Note that in the above analysis, the effects of the coevolution mechanism is omitted. With the coevolution mechanism, the runtime can be much less to cover the whole Pareto front. It is difficult to give a clear analysis on the coevolution process as the interaction of different individuals is hard to analyze; however, we have the following result that can show how MOEA/D benefits from its coevolution mechanism.

Suppose that for the MOEA/D, for any subproblem k ($k = 0, \ldots, n$), its neighbor is set to be subproblems $k - 1$ (if $k > 0$) and $k + 1$ (if $k < n$).

For COCZ, subproblem k has very similar shape with its neighborhood subproblem and their optima are only one mutation step away after the optimal reference point is found. Denote the individual corresponding to the subproblem k as x_k and one individual of its neighbor is x_k^n. According to Algorithm 1 and the above analysis, after the optimal reference point is found, whenever a new x_k' is generated, it will try to update both x_k and $x_k{}^n$. If it can update x_k, which means a better solution is found for subproblem k, then it is also a better solution for subproblem $k+1$ or $k-1$ unless x_k^n is already the optimal solution of its corresponding subproblem. Hence we have the following lemma.

Lemma 2: For the standard MOEA/D on the COCZ problem, after the optimal reference point is found, with x_k and $x_k{}^n$ defined above, every time x_k is updated, $x_k{}^n$ is also updated unless $x_k{}^n$ is already the optimal solution of its corresponding subproblem.

So in this sense, the standard MOEA/D runs much faster as there are at the same time at least two individuals are used to optimize the similar two subproblems. In this way, these two subproblems can actually be solved at the same time. So the runtime of MOEA/D can be reduced at least by 50% after the optimal reference point is found. Although this cannot change the upper bounds shown in Proposition 4, it is still very helpful in practice because the coevolution can work before the optimal reference point is found.

B. Case With No Polynomial-to-One Match

The above analysis shows that when a good match can be found, (seri-)MOEA/D works well. It should be noticed that the case analyzed above has evenly distributed Pareto front, which match well with the evenly distributed weight parameters λ^k. However, there can be cases that the evenly distributed weight parameters cannot give a satisfactory match. In the following analysis, we show an MOP instance for which an evenly distributed weight parameters-based polynomial sized decomposition cannot distinguish some points in the Pareto front.

This MOP instance is from [5], which is an extended version of the leading ones and trailing zeros (LOTZ) [4] problem called weighted leading positive ones trailing negative ones (LPTNO). The problem is defined as

$$\text{Weighted LPTNO}(x) = \left(\sum_{i=1}^{n} w_i \prod_{j=1}^{i}(1+x_j), \sum_{i=1}^{n} v_i \prod_{j=i}^{n}(1-x_j)\right) \quad (11)$$

where x is an assignment to variables $x_1, \ldots, x_n$ with values from $\{-1, 1\}, w_i, v_i > 0, i = 1, \ldots, n$. The Pareto front has $n+1$ points and the corresponding solutions are $1^{\#n}, 1^{\#(n-1)}(-1)^{\#1}, \ldots, 1^{\#1}(-1)^{\#(n-1)}, (-1)^{\#n}$.

Before we analyze the runtime complexity of MOEA/D on LPTNO problem, we first show that there is no polynomial-sized evenly distributed l^k-based decomposition for the LPTNO problem which can yield a good match, where "good match" indicates that for each point of the Pareto front, we have at least one subproblem, with its optimum corresponding to the point.

Lemma 3: For the LPTNO problem with $w_j = v_j = 1$, a predefined polynomial-sized evenly distributed Tchebycheff weight vector cannot guarantee to map each point in the Pareto front to the optimum of a subproblem when the problem scale n is sufficiently large. In another word, there exists at least one point in the Pareto front, for any $k = 0, \ldots, H$, the optimum of the kth subproblem has a fitness vector different from the point.

Proof: As there are $n+1$ points in the Pareto front, we assume that there is a perfect decomposition with special settings of λ^k such that the optimum of the kth subproblem is corresponding to the kth point of the Pareto front. The kth subproblem can be written as follows (with $w_j = v_j = 1$):

$$F_k(x) = \max\left\{\begin{array}{l} \lambda^k\left(\left|\sum_{j=1}^{l} 2^j - \sum_{j=1}^{n} 2^j\right|\right), \\ \left(1-\lambda^k\right)\left(\left|\sum_{j=n-m+1}^{n} 2^{n+1-j} - \sum_{j=1}^{n} 2^{n+1-j}\right|\right) \end{array}\right\} \quad (12)$$

where the optimum for each objective is used as the reference point; l, m indicate the number of leading ones and trailing negative ones of x, respectively. Apparently $l+m \le n$. Simplify $F_k(x)$ we have

$$F_k(x) = \max\left(\lambda^k \sum_{j=l+1}^{n} 2^j, \left(1-\lambda^k\right)\sum_{j=1}^{n-m} 2^{n+1-j}\right). \quad (13)$$

As a Pareto optimal solution must satisfy $l+m=n$, and we assume that the optimum of $F_k(x)$ corresponding to a Pareto optimal solution, so the optimal λ^k must guarantee that a solution with $l+m=n$ minimizes the corresponding $F_k(x)$. Consider the trends of $F_k(x)$ when l changes from 0 to n with $l+m=n$. Obviously, when $l=0$, the first item in (13) is larger than the second item and when $l=n$, the second item is larger than the first item. With l changes from 0 to n, the first item in (13) is decreasing and the second item in (13) is increasing. Thus, there is always a cross point of the two items in $F_k(x)$ and the cross point minimizes $F_k(x)$. The cross point then must satisfy (l aligned to k)

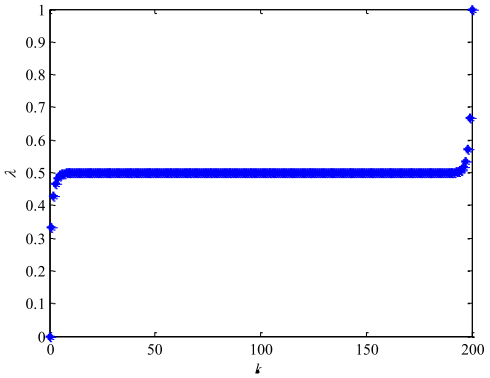


Fig. 1. λ^k changes from $k=0$ to $k=n$ ($n=200$ in this example).

$$\lambda^k \sum_{j=k+1}^{n} 2^j - \left(1-\lambda^k\right)\sum_{j=1}^{k} 2^{n+1-j} = 0$$
$$\Rightarrow \lambda^k = \frac{\sum_{j=1}^{k} 2^{n+1-j}}{\left(\sum_{j=k+1}^{n} 2^j + \sum_{j=1}^{k} 2^{n+1-j}\right)}. \quad (14)$$

Clearly $\lambda^k (k = 0, \ldots, n)$ are not evenly distributed. Fig. 1 shows the distribution of λ^k against k, from which we can see that most of these λ^k fall nearby 0.5. Considering the difference of $\lambda^{(n/2)}$ and $\lambda^{(n/2-1)}$, we have (assuming that $n/2$ is an integer)

$$\Delta = \lambda^{n/2} - \lambda^{n/2-1} = \frac{3}{2\left(2^{2+n/2}-5\right)}. \quad (15)$$

The difference is exponentially small, which means a predefined polynomial-sized, evenly distributed λ^k will not be able to distinguish the optimal $\lambda^{(n/2-1)}$ and $\lambda^{(n/2)}$ when n is sufficiently large. ■

As can be seen from the above lemma, asymptotically there is no polynomial sized evenly distributed weight parameters-based decomposition that can yield a good match. In the following analysis, we will analyze how the MOEA/D performs when a good match is available and not.

C. Runtime Bound Results on Simple MOEAs

Qian *et al.* [5] proved that the expected time to find the whole Pareto front of LPTNO by the SEMO with the SEOF procedure is $\Theta(n^3)$ (with 1-bit or bitwise mutation). Without the SEOF procedure, the runtime complexity is also $\Theta(n^3)$ [4].

The following results are the runtime bounds of seri-MOEA/D on the LPTNO problem. First, we present the runtime bound of seri-MOEA/D when the optimal λ^k shown in (14) is used.

Proposition 5: When the λ^k setting shown in (14) is used for seri-MOEA/D to solve the LPTNO problem, the expected runtime for seri-MOEA/D to cover the whole Pareto front is $\Theta(n^2)$.

Proof of the above proposition can be found in the supplementary material. The key idea of the proof is to prove that

the seri-MOEA/D can solve the next subproblem based on the optimal solution of the current subproblem [in $\Theta(n)$]. The proposition shows that seri-MOEA/D can solve the problem efficiently when the optimal decomposition is used. Without the optimal decomposition, the seri-MOEA/D cannot find each solution in the Pareto front from finding the optimum of each subproblem. However, we shall see in the following proposition that seri-MOEA/D can still cover the whole Pareto front.

Proposition 6: Seri-MOEA/D is used to solve the LPTNO problem with $H = 1$. In this case, only two subproblems are generated by decompostion, which are maximizing the two objectives of LPTNO, respectively. In such case the expected time for seri-MOEA/D to cover the whole Pareto front of the LPTNO problem is $\Theta(n^2)$.

Proof of the above proposition can be found in the supplementary material. The key idea of the proof is to prove that seri-MOEA/D can cover the whole Pareto front by evolving from the optimal solution of the second objective (0th subproblem) to the optimal solution of the first objective (1th subproblem). With this example we can see that solutions found during the search progress of the subproblems also play important roles in MOEA/D besides the optima of the subproblems.

It should be noted that in the above analysis H is set to 1 thus only two subproblems are generated. If H is set larger, such as $H = n$, many of the subproblems will be mapped to points of the two heads of the Pareto front and most of the points in the middle of the Pareto front are not mapped. A decomposition without good match can cause such problem for the seri-MOEA/D. A simple migration test (MT) procedure to deal with these duplicate subproblems can be used to solve this problem, where details can be found in the supplementary material. Analysis shows that the seri-MOEA/D can still efficiently solve the problem.

Proposition 7: Let $H = n$, the expected time for seri-MOEA/D with MT procedure to cover the whole Pareto front of LPTNO is $\Theta(n^2)$.

The above results show that seri-MOEA/D can run faster than the other simple MOEAs with/without optimal decomposition. It should be noticed that SEMO with SEOF is also an ideal algorithm, but it is slower than the seri-MOEA/D, which proves that seri-MOEA/D can run faster because of its decomposition-based search scheme.

1) Analysis on Standard MOEA/D: For the case with standard MOEA/D, according to Lemma 3, an evenly distributed weight vector cannot make sure that each Pareto point is mapped into a subproblem. We first show that the standard MOEA/D can cover the whole Pareto front of LPTNO efficiently if the optimal λ^k setting in (14) is used.

Proposition 8: For a standard MOEA/D with optimal decomposition parameters as shown in (14) with $n + 1$ subproblems and each subproblem mapped to a different point in the Pareto front, its runtime on LPTNO are upper bounded by $O(n^3)$.

Proof: As we have the optimal decomposition, to compute the time needed to cover the whole Pareto front, it is enough to analyze the time each subproblem used until it is solved and then add them up. Similar to the analysis on COCZ, the time used to find the optima of the two objective functions are bounded by $\Theta(n^3)$. After that, each subproblem can be solved by finding a solution with l leading 1-bits and m trailing -1-bits with $l + m = n$. Any solution with $l + m' < n$ is no better than solution with $l+m = n$ because for the case (l, m'), the second part of (13) becomes smaller, so their maximum is no worse than the case (l, m). In this sense, the subproblem k will try to find a solution with $l + m = n$ and with $l = k$. So each subproblem equals to the standard leading ones problem and can be solved in $O(n^2)$.

Note that neighborhood-based coevolution scheme will not slow down the optimization speed of each subproblem, so the above upper bound stays true. As there are $n - 1$ subproblems to be solved, the total time for a standard MOEA/D to cover the whole Pareto front of LPTNO are upper bounded by $O(n^3)$. ■

For the case without good match, we argue that in rare cases the algorithm needs a long time to solve the problem. When evenly distributed weight parameters are used, there are always some Pareto points not mapped to any subproblem, suppose these "missed" Pareto points corresponding to the Pareto optimal solutions with k leading 1-bits, where $k = l, \ldots, m$. It is then possible that in some generation of the evolution process, the current population has only solutions with less than l leading 1-bits and less than $n - m$ trailing -1-bits and the Pareto optimal solutions with leading 1-bits in range [l, m] not covered yet. To cover these points, individuals in the population with $l - 1$ leading 1-bits or $m + 1$ leading 1-bits are mostly likely to generate a new Pareto optimal solution. However, when such a new Pareto optimal solution is generated, it cannot stay in the current population as no subproblem prefers it. So, to cover all these Pareto solutions with leading 1-bits in range [l, m], we need to mutate multiple bits at the same time with bitwise mutation. The success probability to mutate multiple 1-bits into -1-bits at the same time is low, which means the algorithm will spend a long time to cover the whole Pareto front.

The above results show that the evenly distributed weight vector has its deficiency in certain cases. That is also why there has been many literatures trying to propose new methods to set weight parameters. Our results actually show that one can try to put most recently found new Pareto optimal solutions into the evolution process, especially if these individuals are not kept in the current population as they are not the optimal solution of any subproblems.

D. Summary

Overall, we summarize the runtime analysis results on COCZ and LPTNO (LOTZ) in Table I, with additional results of GEMO from [4]. Note that for standard MOEA/D, the optimal decomposition mentioned is used. Simulation results shown in Figs. 2 and 3 verify the above analysis results. Due to the numerical precision problem when n is large, which can affect the LOTZ problem, we only test n up to 50.

On one hand we can see that among all the single individual-based simple MOEAs, seri-MOEA/D shows its advantage

TABLE I
SUMMARY OF THE EXPECTED RUNTIME TO COVER THE WHOLE PARETO FRONT FOR DIFFERENT MOEAS. FOR MOEA/D THE OPTIMAL DECOMPOSITION PARAMETERS ARE USED

Algorithm / Problems	SEMO	SEMO with SEOF	GEMO	seri-MOEA/D	MOEA/D
COCZ	$\Theta(n^2\log n)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n\log n)$	$O(n^2\log n)$
LPTNO (LOTZ)	$\Theta(n^3)$	$\Theta(n^3)$	$O(n^2\log n)$	$\Theta(n^2)$	$O(n^3)$

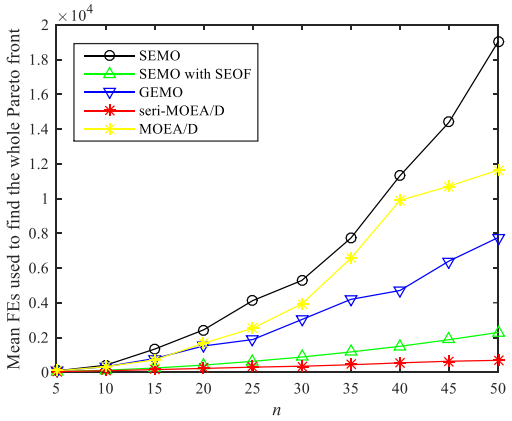


Fig. 2. Comparison on COCZ problem ($n = 0 \sim 50$).

in covering the whole Pareto front when a good neighborhood relationship of Pareto optimal solutions exists. On the other hand, the numerical simulation results corroborate that MOEA/D has a similar runtime complexity with SEMO (but not the same, as we only have upper bounds for MOEA/D). The standard MOEA/D runs slower than the seri-MOEA/D on this two simple MOP instances, due to the population-based search strategy of standard MOEA/D, which is slow here compared to seri-MOEA/D, but can be critical to solve some more difficult problems in practice.

V. ANALYZING MOEA/D ON TWO DIFFICULT MOP INSTANCES

In Section IV, we analyzed the runtime complexity of MOEA/D with respect to the decomposition parameters on two simple MOP instances. It should be noticed that the MOEA/D can work well when there are suitable nearby-relationships among the subproblems after the Tchebycheff decomposition. Apparently, the Tchebycheff decomposition is based on the fitness space of the objectives. However, the nearby-relationship in the fitness space cannot guarantee a nearby-relationship in the decision space. Thus, the scalar subproblem optimization-based search scheme used in MOEA/D may fail in certain cases. On the other hand, Berghammer *et al.* [7] showed that using purely Pareto dominance-based search scheme can also be inefficient in some cases. In the following analysis, we show cases that MOEA/D can work well actually because two kinds of search schemes are properly combined in MOEA/D.

A. Case With Deceptive Objective Functions

In single objective optimization, there are problems for which the decision space nearby-relationship does not match

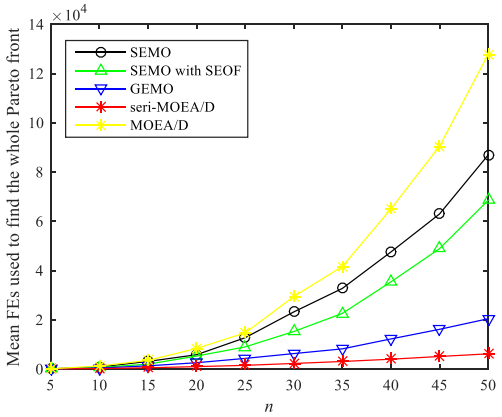


Fig. 3. Comparison on LPTNO (LOTZ) problem ($n = 0 \sim 50$).

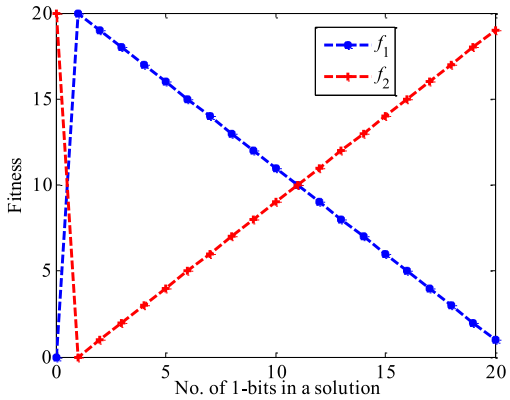


Fig. 4. Dec-obj-MOP ($n = 20$).

the fitness space, and these problems are termed as "deceptive" problems. For example, the pseudo Boolean function: $f(x) = n, 0, 1, \ldots, (n-1)$ when the number of 1-bits in $x = (x_1, x_2, \ldots, x_n)$ changes from 0 to n, is deceptive and hard to solve. To maximize $f(x)$, one need to find the solution with $||x||_0 = 0$. Usually we hope that the closer to the solution with $||x||_0 = 0$, the better fitness we get. However, for this function the fitness is decreasing when $||x||_0$ changes from n to 1, which will guide the solution in the opposite direction of the global optimum $||x||_0 = 0$.

In the following analysis, we consider the MOP instance with a deceptive objective named dec-obj-MOP (as shown in Fig. 4) defined on $\{0,1\}^n$:

$$f_1(x) = \mathrm{mod}(n + 1 - ||x||_0, n + 1)$$
$$f_2(x) = \mathrm{mod}(n + ||x||_0, n + 1). \quad (16)$$

Clearly for the above problem we have $f_1(x) + f_2(x) = n$, which shows that the Tchebycheff decomposition of this problem is exactly the same to the COCZ problem analyzed in Section IV. Set $H = n$ we have $n + 1$ subproblems and the fitness vector corresponding to the optimum of the kth subproblem is $(f_1, f_2) = (k, n - k)$, where $k = 0, \ldots, n$.

Although, the problem and COCZ have the same fitness space structure, they are quite different in the decision-fitness space map, and thus the runtime complexity, as shown by the following proposition.

Proposition 9: SEMO with SEOF along with bitwise mutation and seri-MOEA/D with bitwise mutation are expected to cover the whole Pareto front of the dec-obj-MOP problem in exponential time.

Proof: For the SEMO algorithm, as the SEOF procedure searches for the optimum of the second objective which is deceptive, the expected time cost by SEOF is exponential [48]. Thus the SEMO has to spend exponential time in initialization which makes the total time exponential. Similar problem exists for the seri-MOEA/D as seri-MOEA/D solves the 0th subproblem which equals to minimize $(n-f_2(x))$ first. Thus, both algorithms need exponential time to cover the whole Pareto front. Note that we use bitwise mutation for SEMO here so that it can find the optimum of the problem. ■

The above proposition shows that the deceptive objective function do misguide the search. However, we shall see that the dec-obj-MOP, which is constructed with a deceptive objective, can be solved easily in polynomial time for both SEMO without the SEOF procedure and standard MOEA/D.

Proposition 10: Both SEMO with 1-bit mutation but without SEOF and the standard MOEA/D with 1-bit mutation and with $H = n$ are expected to cover the whole Pareto front of the dec-obj-MOP problem in $O(n^2 \log n)$.

Proof: For the SEMO algorithm without SEOF, the initial solution is generated randomly with $||x||_0 = k, 0 \le k \le n$. The probability to add a new Pareto optimal solution to P when there are already i solutions in P is the same as Proposition 2, which is $(1/i)*((n-i+1)/n)$, and the expected time to cover the whole Pareto front is

$$\begin{aligned} n\sum_{i=1}^{n}\frac{i}{n-i+1} &= n\left(\sum_{i=1}^{n}\frac{n+1}{n-i+1}+\sum_{i=1}^{n}\frac{i-n-1}{n-i+1}\right) \\ &= n\left(\sum_{i=1}^{n}\frac{n+1}{n-i+1}-n\right) \\ &= O\left(n^2\log n\right). \end{aligned} \tag{17}$$

For MOEA/D, an initial population is randomly optimized to solve each subproblem. Regardless of the fact that the 0th subproblem is still hard to solve, the other subproblems can be solved in polynomial time. According to the analysis on COCZ problem, these subproblems can be solved in $O(n^2 \log n)$. Suppose the solution $||x||_0 = 1$ for the nth subproblem is found. In the following process, the nth problem continues to mutate $||x||_0 = 1$ and the probability to produce $||x||_0 = 0$ is $1/n$. Thus $||x||_0 = 0$ can be found in $O(n)$ expected time after $||x||_0 = 1$ is found. In total the MOEA/D can cover the whole Pareto front in polynomial time $O(n^2 \log n)$. ■

In a word, both SEMO without SEOF and MOEA/D solve the dec-obj-MOP problem in polynomial time. The reason is that when the two objectives are put together, the optimum is changed from a single solution to an optimal set. The solutions in the Pareto optimal set have good decision space nearby-relationship.

B. Deceptive Case With Nondeceptive Objectives

In opposition to the above example, an MOP which is constructed from nondeceptive objectives, but end up to be deceptive for SEMO without SEOF is proposed here. The two objectives of the MOP instance called Plateau-MOP defined on $\{0,1\}^n$ are as follows, where $1^{\#i}0^{\#n-i}$ denotes a solution with i 1-bits in a row and followed by $(n\text{-}i)$ 0-bits in a row:

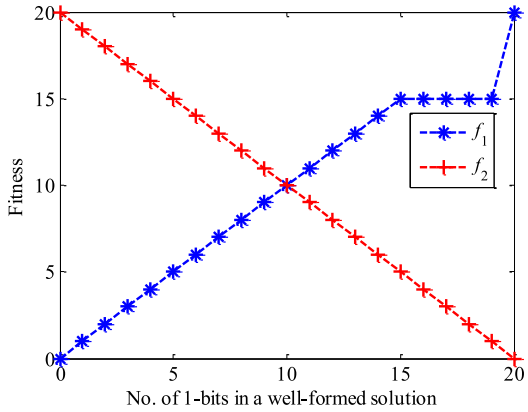


Fig. 5. Plateau-MOP ($n = 20$).

$$f_1(x) = \begin{cases} n & \text{if } x = 1^n \\ (3/4)n & \text{if } x = 1^{\#i}0^{\#n-i}, \\ & \quad i \in \{(3/4)n+1, \ldots, n-1\} \\ i & \text{if } x = 1^{\#i}0^{\#n-i}, i \in \{0,1,\ldots,(3/4)n\} \\ -\|x\|_0 & \text{otherwise} \end{cases} \tag{18}$$

$$f_2(x) = \begin{cases} n-i & \text{if } x = 1^{\#i}0^{\#n-i}, i = \{0,1,\ldots,n\} \\ -\|x\|_0 & \text{otherwise.} \end{cases} \tag{19}$$

The objective function f_1 is a modified version of the Plateau function [6]. f_1 is set to be below zero when x is not in the form of $1^{\#i}0^{\#(n-i)}$. When optimize f_1, any solution with negative fitness will try to minimize the 1-bits in it or directly jump to any solution with a good form $1^{\#i}0^{\#(n-i)}$, which we call "well-formed". Neither objective function is deceptive because for either objective a valid step can always be found to make a solution closer to the optimal solution. When the solution is well-formed, the solution falls on a path with different number of 1-bits, we call this path the well-formed solution path. The objective function f_2 has the same fitness to f_1 when x is not well-formed; when x is well-formed, f_2 is the inverse OneMax problem which makes it an opposite objective against f_1 along the well-formed solution path. Fig. 5 shows the fitness of f_1 and f_2 along this path. Note that there is a plateau of length $(1/4)n$ for f_1.

In Fig. 5, we only show the fitness of the two objectives when x is well-formed. As when x is not well-formed, both objectives have fitness below 0, these solutions are dominated by solutions in the well-formed solutions path. From Fig. 5 we can see that the Pareto optimal set is the same as the LOTZ problem if without the plateau of f_1. The plateau eliminates the LOTZ Pareto optimal solutions in range $[(3/4)n+1, n-1]$ because any solution in this range is dominated by the well-formed solution with $||x||_0 = (3/4)n$. The Pareto optimal set consists of solutions with $||x||_0 = 0, \ldots, (3/4)n$ along the well-formed solutions path and $||x||_0 = n$ which is an "outlier" because it cannot be directly generated by any other Pareto optimal solutions. Apparently, the outlier has no good nearby-relationship with the other Pareto optimal solutions in the

decision space. For this instance without good decision space nearby-relationship, we consider the difference of SEMO without SEOF and MOEA/D in their runtime complexity to cover the whole Pareto front.

Proposition 11: With overwhelming probability, the expected time for SEMO with bitwise mutation and without SEOF to cover the whole Pareto front of Plateau-MOP is exponential.

Proof: Recall that SEMO without SEOF will create a random solution at the beginning and then in each generation, a random solution is selected from the Pareto optimal set and then it goes through a mutation operation to find a better solution.

In the following, we show a typical run of the SEMO algorithm (a similar proof can be found in [42]). First, we show that with overwhelming probability $(1-2^{\Omega\ (-n)})$ the first well-formed solution found by SEMO satisfies $||x||_0 < (3/4)n$. Consider the initial solution, by Chernoff's bounds [6], the initial solution satisfies $||x||_0 < (2/3)n$ with overwhelming probability $(1-2^{(\Omega\ -n)})$. If the solution is well-formed, then we have the first well-formed solution satisfying $||x||_0 < (3/4)n$; If the initial solution is not well-formed, the probability it changes into a well-formed solution with $||x||_0 > (3/4)n$ from a solution with $||x||_0 \leq (2/3)n$ is at most

$$\left(\frac{1}{n}\right)^{n/12}\left(1-\frac{1}{n}\right)^{11n/12} < \left(\frac{1}{n}\right)^{n/12}. \tag{20}$$

This is exponentially small. If it cannot mutate to a well-formed solution, it will evolve toward well-formed solution with $||x||_0 = 0$, which obviously satisfies $||x||_0 \leq (3/4)n$. Thus, with overwhelming probability, the first well-formed solution satisfies $||x||_0 \leq (3/4)n$.

Second, consider the case when a well-formed solution is found and of course it is put into the Pareto optimal set and all the nonwell-formed solutions are eliminated. From the first well-formed solution, SEMO is able to find the well-formed solution with $||x||_0 \leq (3/4)n$ in polynomial time. Consider the time the algorithm needed to find the outlier $||x||_0 = n$. As all solutions with $||x||_0$ in range $[(3/4)n+1, n-1]$ are dominated by the well-formed solution $||x||_0 = (3/4)n$ thus these solutions cannot join the Pareto optimal set because the $1^{\#(3/4)n}0^{\#n/4}$ already exists. And all the other solutions in the Pareto optimal set are well-formed, and thus have a further distance to the outlier than $1^{\#(3/4)n}0^{\#n/4}$, thus the largest probability to generate the outlier is

$$\left(\frac{1}{n}\right)^{n/4}\left(1-\frac{1}{n}\right)^{(3/4)n} < \left(\frac{1}{n}\right)^{n/4} \tag{21}$$

which is exponentially small and thus the time needed to find the outlier is exponential. Thus, with overwhelming probability, the SEMO needs exponential time to find the whole Pareto front, which proves the proposition. ■

Proposition 12: The expected time to cover the whole Pareto front of Plateau-MOP for MOEA/D with $H = n$ is $O(n^3)$.

Proof: First we show that each Pareto optimal solution is mapped to a subproblem of MOEA/D.

The kth subproblem after decomposition is

$$F_k(x) = \max\left\{\lambda^k(n-f_1(x)), \left(1-\lambda^k\right)(n-f_2(x))\right\}. \tag{22}$$

To minimize $F_k(x)$, x must be well-formed. Otherwise assuming that x is not well-formed, for any well-formed solution x', we have $F_k(x') < F_k(x)$ because $f_i(x') > 0$ and $f_i(x) < 0, i = 1, 2$. Thus the optimum of $F_k(x)$ must be well-formed. Consider the case when x is well-formed. Denote the difference of the two items in $F_k(x)$ as $\Delta(x)$ and assume that n can be divided by 4. As $H = n$ we have $\lambda^k = k/n$

$$\Delta(x) = \begin{cases} k - \|x\|_0, & \text{if } |x| \in \left[0,(3/4)n\right]\bigcup\{n\} \\ \dfrac{k}{4} - \left(1-\dfrac{k}{n}\right)\|x\|_0, & \text{otherwise.} \end{cases} \tag{23}$$

Discuss when k varies from 0 to n: when $k = 0, 1, \ldots, (3/4)n$, if $\Delta(x) \geq 0$, i.e., $||x||_0 \leq k, F_k(x)$ equals to its first item in (22); otherwise if $||x||_0 > k, F_k(x)$ equals to its second item in (22). Consider $F_k(x)$ when $||x||_0$ is increasing, the first item in (22) is nonincreasing and the second item in (22) is nondecreasing, thus the optimal solution to $F_k(x)$ satisfies $||x||_0 = k$, with fitness vector $(k, n-k)$, which is a Pareto optimal solution.

Similarly, when $k = (3/4)n+1, \ldots, n-1, F_k(x)$ equals to its first item in (22) when $||x||_0 \leq n(4l+3n)(4n-16l)$ and equals to its second item otherwise, where $l = k-(3/4)n$. The optimal solution to $F_k(x)$ satisfies $||x||_0 = \lfloor n(4l+3n)/(4n-16l)\rfloor$, which is not Pareto optimal or same to $||x||_0 = (3/4)n$.

When $k = n, F_k(x)$ equals to its first item in (22) when $||x||_0 \leq n$. The optimal solution to $F_k(x)$ then satisfies $||x||_0 = n$, which is a Pareto optimal solution with fitness vector $(n, 0)$.

Overall, for each Pareto optimal solution there is a subproblem of MOEA/D matching it. As shown above, subproblems $F_k(x)$ with $k = 0, \ldots, (3/4)n$, and $k = n$ correspond to all the Pareto optimal solutions. Solving these subproblems is thus enough to cover the whole Pareto optimal front. For $k = 0, \ldots, (3/4)n, F_k(x)$ can be solved within the same time complexity of the LPTNO problem, which is $O(n^2)$. The overall time complexity to solve all $F_k(x)$ with $k = 0, \ldots, (3/4)n$ is thus $O(n^3)$. For case $k = n$, i.e., $F_n(x) = n - f_1(x)$, which has a plateau of length $n/4$, according to the results on the expected runtime analysis of the plateau function [6, p. 131], the expected time to cross the plateau is $O(n^3)$. With the time to become a well-formed solution $O(n \log n)$ and time to get to the plateau $O(n^2)$, the overall expected time to find the optimum of $F_n(x)$ is $O(n^3)$. Thus the total expected time to cover the whole Pareto front is $O(n^3)$, which means MOEA/D can solve Plateau-MOP in polynomial time. ■

C. Summary

We show that SEMO without SEOF can solve the dec-obj-MOP problem efficiently but cannot efficiently solve the Plateau-MOP problem. On the other hand, SEMO with SEOF can solve the Plateau-MOP problem efficiently but fails on the dec-obj-MOP problem. SEMO with SEOF can solve the

Plateau-MOP because the SEOF procedure can find the outlier Pareto optimal solution by minimizing f_2 in polynomial time like MOEA/D. For both problems, MOEA/D can solve them efficiently. Thus, MOEA/D shows more versatility in these two cases.

In a deeper view, the SEOF procedure can be taken as a scalar subproblem-based search scheme. SEMO without SEOF only uses Pareto domination-based search scheme. In MOEA/D, the scalar subproblem-based search scheme is used, and the Pareto domination-based search scheme is also used, which is concealed in Step "Update P" of algorithm MOEA/D. It should be noticed that the Pareto domination relation is a weak relation, which may cause inefficiency as shown above while the scalar-based search scheme has a strong but narrow targeting. The SEMO with SEOF shows a rather raw way to combine both kinds of search schemes; while MOEA/D shows a better way to combine both kinds of search schemes, which shows better adaptation ability.

VI. Conclusion

In this paper, we present the very basic theoretical study on decomposition-based MOEAs. Runtime analysis is done on simple MOP instances COCZ and LPTNO (LOTZ). For the first COCZ instance, the MOEA/D can enjoy a good decomposition while for the LPTNO (LOTZ) instance, we prove that evenly distributed weight parameters cannot give a good decomposition. With these different decomposition situations, we analyze the runtime of standard MOEA/D, and show that the algorithm can benefit from its neighborhood-based coevolution mechanism. We also study and compare simple MOEAs to show how the decomposition-based seri-MOEA/D can run faster than other single individual MOEAs on these simple MOP instances.

We also conduct further analysis on more difficult MOP instances to show that both the Pareto domination-based and the scalar subproblem-based search schemes play important roles in MOEA/D. Two MOP instances are developed to test different kinds of search schemes, one with deceptive objective while the other one with an objective function containing a plateau. Analysis results show that MOEA/D can solve both instances in polynomial time while algorithms using pure Pareto dominance-based or scalar subproblem-based search scheme needs exponential time to solve one of the two problems. The results prove that MOEA/D combines these two kinds of search schemes in a proper way.

For the future work, we believe that the theoretical aspects of decomposed-based MOEAs can be further studied. For example, various real-world MOP instances such as in cloud computing environment [72], [73] shall be studied with MOEA/D, as we only analyzed very simple MOP instances here. General runtime complexity studies of MOEA/D on certain categories of MOPs are also very interesting. Moreover, accommodate the MOEA/D framework to enhanced particle swarm optimization [67]–[69] or differential evolution [70], [71] algorithms and analyze the run time are also significant. At last, instead of studying the cover time of the Pareto front, one may also try to analyze the approximating rate of the Pareto front when MOEA/D is used to solve MOP instances with continuous or exponential-sized Pareto front.

References

[1] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, no. 1, pp. 1–16, 1995.

[2] K. Deb, A. Pratap, S. Agarwal, 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.

[3] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[4] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 170–182, Apr. 2004.

[5] C. Qian, Y. Yu, and Z.-H. Zhou, "An analysis on recombination in multiobjective evolutionary optimization," *Artif. Intell.*, vol. 204, pp. 99–119, Nov. 2013.

[6] T. Jansen, *Analyzing Evolutionary Algorithms the Computer Science Perspective*. Berlin, Germany: Springer, 2013.

[7] R. Berghammer, T. Friedrich, and F. Neumann, "Convergence of set-based multiobjective optimization, indicators and deteriorative cycles," *Theor. Comput. Sci.*, vol. 456, pp. 2–17, Oct. 2012.

[8] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization—-Algorithms and Their Computational Complexity*. Berlin, Germany: Springer, 2010.

[9] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artif. Intell.*, vol. 127, no. 1, pp. 57–85, Mar. 2001.

[10] B. Doerr, D. Johannsen, and C. Winzen, "Multiplicative drift analysis," *Algorithmica*, vol. 64, no. 4, pp. 673–697, Dec. 2012.

[11] Y. R. Zhou, J. He, and Q. Nie, "A comparative runtime analysis of heuristic algorithms for satisfiability problems," *Artif. Intell.*, vol. 173, no. 2, pp. 240–257, Feb. 2009.

[12] C. Witt, "Tight bounds on the optimization time of a randomized search heuristic on linear functions," *Comb. Probab. Comput.*, vol. 22, no. 2, pp. 294–318, Mar. 2013.

[13] B. Doerr, T. Kötzing, J. Lengler, and C. Winzen, "Black-box complexities of combinatorial problems," *Theor. Comput. Sci.*, vol. 471, pp. 84–106, Feb. 2013.

[14] T. Kötzing, F. Neumann, H. Röeglin, and C. Witt, "Theoretical analysis of two ACO approaches for the traveling salesman problem," *Swarm Intell.*, vol. 6, no. 1, pp. 1–21, Mar. 2012.

[15] A. M. Sutton and F. Neumann, "A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling," in *Parallel Problem Solving From Nature—PPSN XII*. Berlin, Germany: Springer, 2012, pp. 52–61.

[16] Y. Yu, X. Yao, and Z.-H. Zhou, "On the approximation ability of evolutionary optimization with application to minimum set cover," *Artif. Intell.,* vols. 180–181, pp. 20–33, Apr. 2012.

[17] S. Droste, "Analysis of the (1+1) EA for a noisy ONEMAX," in *Proc. Genet. Evol. Comput. Conf.*, Seattle, WA, USA, 2004, pp. 1088–1099.

[18] P. K. Lehre and X. Yao, "On the impact of mutation-selection balance on the runtime of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 225–241, Apr. 2012.

[19] T. Jansen and C. Zarges, "Fixed budget computations: A different perspective on run time analysis," in *Proc. Genet. Evol. Comput. Conf.*, Philadelphia, PA, USA, 2012, pp. 1325–1332.

[20] I. Karahan and M. Koeksalan, "A territory defining multiobjective evolutionary algorithms and preference incorporation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 636–664, Aug. 2010.

[21] C. K. Chow and S. Y. Yuen, "A multiobjective evolutionary algorithm that diversifies population by its density," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 149–172, Apr. 2012.

[22] Z.-H. Zhan *et al.*, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[23] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 6, pp. 565–588, Dec. 2001.

[24] Z. He, G. G. Yen, and J. Zhang, "Fuzzy-based Pareto optimality for many-objective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 269–285, Apr. 2014.

[25] P. A. N. Bosman, "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 51–69, Feb. 2012.

[26] K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius, "An interactive evolutionary multiobjective optimization method based on progressively approximated value functions," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 723–739, Oct. 2010.

[27] Z. Cai and Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 658–675, Dec. 2006.

[28] A. Zhou, Y. Jin, and Q. Zhang, "A population prediction strategy for evolutionary dynamic multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 40–53, Jan. 2014.

[29] M. A. Abido, "Multiobjective evolutionary algorithms for electric power dispatch problem," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 315–329, Jun. 2006.

[30] N. V. Venkatarayalu, T. Ray, and Y.-B. Gan, "Multilayer dielectric filter design using a multiobjective evolutionary algorithm," *IEEE Trans. Antennas Propag.*, vol. 53, no. 11, pp. 3625–3632, Nov. 2005.

[31] D. Martin, A. Rosete, J. Alcala-Fdez, and F. Herrera, "A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 54–69, Feb. 2014.

[32] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 4–19, Feb. 2014.

[33] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 321–344, Jun. 2013.

[34] R. E. Steuer and E.-U. Choo, "An interactive weighted Tchebycheff procedure for multiple objective programming," *Math. Program.*, vol. 26, no. 3, pp. 326–344, 1983.

[35] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[36] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Effects of using two neighborhood structures on the performance of cellular evolutionary algorithms for many-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 2508–2515.

[37] S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 442–446, Jun. 2012.

[38] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," School Comput. Sci. Electron. Eng., Univ. Essex, Colchester, U.K., Tech. Rep. CES-491, 2009.

[39] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function-based multiobjective evolutionary algorithm," in *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2009, pp. 438–452.

[40] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Simultaneous use of different scalarizing functions in MOEA/D," in *Proc. Genet. Evol. Comput. Conf.*, Portland, OR, USA, 2010, pp. 519–526.

[41] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 456–474, Jun. 2010.

[42] D. Sudholt, "On the analysis of the $(1+1)$ memetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, Seattle, WA, USA, 2010, pp. 493–500.

[43] S. Ghosh, S. Das, A. V. Vasilakos, and K. Suresh, "On convergence of differential evolution over a class of continuous functions with unique global optimum," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 1, pp. 107–124, Feb. 2012.

[44] T. Chen, K. Tang, G. Chen, and X. Yao, "Analysis of computational time of simple estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 1–22, Feb. 2010.

[45] Q. Zhang and H. Muehlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.

[46] C. Witt, "Runtime analysis of the $(\mu+1)$ EA on simple pseudo-Boolean functions," *Evol. Comput.*, vol. 14, no. 1, pp. 65–86, 2006.

[47] P. S. Oliveto and C. Witt, "Simplified drift analysis for proving lower bounds in evolutionary computation," *Algorithmica*, vol. 59, no. 3, pp. 369–386, 2011.

[48] A. Auger and B. Doerr, "Runtime analysis of evolutionary algorithm for discrete optimization," in *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. Hackensack, NJ, USA: World Scientific, 2011, pp. 21–52.

[49] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Adv. Appl. Probab.*, vol. 14, no. 3, pp. 502–525, 1982.

[50] B. Doerr, "Drift analysis," in *Proc. Genet. Evol. Comput. Conf.*, Dublin, Ireland, 2011, pp. 1311–1320.

[51] I. Wegener, "Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions," in *Evolutionary Optimization*. New York, NY, USA: Springer, 2002, pp. 349–369.

[52] D. Sudholt, "A new method for lower bounds on the running time of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 418–435, Jun. 2013.

[53] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem," *Theor. Comput. Sci.*, vol. 378, no. 1, pp. 32–40, 2007.

[54] M. Laumanns, L. Thiele, and E. Zitzler, "Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 170–182, Apr. 2004.

[55] L. Paquete and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," in *Proc. Evol. Multi-Criter. Optim.*, Faro, Portugal, 2003, pp. 479–493.

[56] Y. Yu, C. Qian, and Z. Zhou, "Switch analysis for running time analysis of evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 777–792, Dec. 2015.

[57] Y. Yu and C. Qian, "Running time analysis: Convergence-based analysis reduces to switch analysis," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 2603–2610.

[58] C. Qian, Y. Yu, Y. Jin, and Z.-H. Zhou, "On the effectiveness of sampling for evolutionary optimization in noisy environments," in *Parallel Problem Solving From Nature*. New York, NY, USA: Springer, 2014, pp. 302–311.

[59] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 20–35, Feb. 2014.

[60] J. He, T. Chen, and X. Yao, "On the easiest and hardest fitness functions," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 295–305, Apr. 2015.

[61] S. B. Gee, K. C. Tan, V. A. Shim, and N. R. Pal, "Online diversity assessment in evolutionary multiobjective optimization: A geometrical perspective," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 542–559, Aug. 2015.

[62] X. Y. Cai, Y. X. Li, Z. Fan, and Q. Zhang, "An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 508–523, Aug. 2015.

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

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

[65] H. D. Wang, L. C. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2015.

[66] J. X. Cheng, G. G. Yen, and G. X. Zhang, "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 592–605, Aug. 2015.

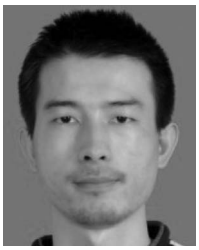
[67] Y. H. Li, Z.-H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Inf. Sci.*, vol. 293, pp. 370–382, Feb. 2015.

[68] M. Shen *et al.*, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7141–7151, Dec. 2014.

[69] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[70] Y.-L. Li *et al.*, "Fast micro-differential evolution for topological active net optimization," *IEEE Trans. Cybern.*, 2015. DOI: 10.1109/TCYB.2015.2437282.

[71] Y.-L. Li *et al.*, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.
[72] Z.-H. Zhan *et al.*, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surv.*, vol. 47, no. 4, Jul. 2015, Art. ID 63.
[73] H.-H. Li, Z.-G. Chen, Z.-H. Zhan, K.-J. Du, and J. Zhang, "Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment," in *Proc. Genet. Evol. Comput. Conf.*, Madrid, Spain, 2015, pp. 1419–1420.



Yuan-Long Li (S'11) received the B.Sc. degree in mathematics and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2009 and 2014, respectively.

He is a Research Fellow with Nanyang Technological University, Singapore. His research interests include global optimization and time series analysis and their applications in data center modeling.

Yu-Ren Zhou received the B.Sc. degree in mathematics from Peking University, Beijing, China, in 1988, and the M.Sc. degree in mathematics and the Ph.D. degree in computer science from Wuhan University, Wuhan, China, in 1991 and 2003, respectively.

He is a Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include design and analysis of algorithms, evolutionary computation, and social networks.

Zhi-Hui Zhan (M'13) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He has published over 50 international journal and conference papers, including about 20 papers in IEEE Transactions and IEEE Magazines. His research interests include evolutionary computation, cloud computing, and evolutionary computation applications in real-world problems.

Dr. Zhan was a recipient of the China Computer Federation Outstanding Dissertation in 2013 for his doctoral dissertation, the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China, in 2014, and the Pearl River New Star in Science and Technology in 2015. He is listed as one of the Most Cited Chinese Researchers in Computer Science.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from City University of Hong Kong, Hong Kong, in 2002, under the supervision of Prof. H. Chung.

Since 2004, he has been a Changjiang Scholars Professor with Sun Yat-sen University, Guangzhou, China. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. His research interests include computational intelligence, cloud computing, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and the ACM Guangzhou Chapter.