Running Time Analysis of MOEA/D with Crossover on Discrete Optimization Problem

Zhengxin Huang,[1] **Yuren Zhou,**[1,2,*] **Zefeng Chen,**[1] **Xiaoyu He**[1]
[1]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China
[2]Engineering Research Institute, Guangzhou College of South China, University of Technology, Guangzhou 510800, China
Email: thenewyi@gmail.com, zhouyuren@mail.sysu.edu.cn,
chzfeng@foxmail.com, hxyokokok@foxmail.com

Abstract

Decomposition-based multiobjective evolutionary algorithms (MOEAs) are a class of popular methods for solving multiobjective optimization problems (MOPs), and have been widely studied in numerical experiments and successfully applied in practice. However, we know little about these algorithms from the theoretical aspect. In this paper, we present a running time analysis of a simple MOEA with crossover based on the MOEA/D framework (MOEA/D-C) on four discrete optimization problems. Our rigorous theoretical analysis shows that the MOEA/D-C can obtain a set of Pareto optimal solutions to cover the Pareto front of these problems in expected running time apparently lower than the one without crossover. Moreover, the MOEA/D-C only needs to decompose an MOP into a few scalar optimization subproblems according to several simple weight vectors. This result suggests that the use of crossover in decomposition-based MOEA can simplify the setting of weight vector for different problems and make the algorithm more efficient. This study theoretically explains why some decomposition-based MOEAs work well in computational experiments and provides insights in design of MOEAs for MOPs in future research.

Introduction

Multiobjective optimization problem (MOP) involves the simultaneous optimization of multiple conflicting objectives. It exists widely in real-world applications (Zhou et al. 2011). The goal of multiobjective optimization is to find a set of the best eclectic solutions called the Pareto optimal solutions. Due to the advantage of population-based nature, evolutionary algorithms (EAs) are able to obtain multiple Pareto optimal solutions in a single run, and multiobjective EAs (MOEAs) have been very popular in solving MOPs. MOEAs are broadly classified into three categories, i.e., domination-based, indicator-based and decomposition-based (Trivedi et al. 2017). In this paper, we present a theoretical analysis of a simple decomposition-based MOEA with crossover (MOEA/D-C) on several discrete optimization problems.

The idea of decomposition for solving MOPs has been used in (Ishibuchi and Murata 1998; Murata and Gen 2002). However, it becomes popular after the MOEA/D framework is presented in (Zhang and Li 2007). In this framework, an MOP is first decomposed into some scalar optimization subproblems according to the decomposition approach and weight vector. Then all subproblems are solved simultaneously by employing an EA. In the past decade, there are vast studies that have contributed to improving and developing MOEAs based on the MOEA/D framework, e.g., studies on novel weight vector generation methods (Qi et al. 2014; Giagkiozis, Purshouse, and Fleming 2014), improved decomposition approaches (Ishibuchi et al. 2010; Wang, Zhang, and Zhang 2016) and selection mechanisms (Chiang and Lai 2011; Li et al. 2017). For further decomposition-based MOEAs, the readers are advised to (Trivedi et al. 2017).

In (Ishibuchi et al. 2017), the authors conducted an experimental analysis for MOEAs based on the MOEA/D framework on the DTLZ and WFG test problems. Their results show that the performance of the algorithm strongly depends on Pareto front shapes. In (Tanabe and Ishibuchi 2018), the authors presented a parameter study in the MOEA/D framework using the unbounded external archive (UEA). Their experimental results indicate that suitable settings of the three control parameters (population size, scalarizing functions and penalty parameter of the penalty-based boundary intersection (PBI) function) significantly depend on the choice of UEA scenarios. An experimental study on the effect of reference point in the MOEA/D framework for optimizing WFG test problems is presented in (Wang et al. 2017). Their experimental results show that different reference point specifications lead to different performance of exploitation and exploration, and the strategy of dynamic reference point specifications is recommended to use for unknown problems.

Running time analysis is one of the most powerful theory tools to understand the performance of EAs. The first theoretical study for decomposition-based MOEAs is presented in (Li et al. 2016). They presented a running time analysis of a simple MOEA with only mutation based on MOEA/D framework, denoted as MOEA/D-M, on four discrete optimization problems. Their analyzed results show that if the optimally decomposed weight vector is used, the expected running time of the MOEA/D-M on these problems are better than the simple evolutionary multiobjective optimizer (SEMO) (Laumanns et al. 2002), which has been theoretically analyzed on many MOPs in the literature. However, they also show that the optimally decomposed weight vector for an MOP depends on its properties and may not be

obtained by using the evenly distributed generation method. This explains why an optimally decomposed weight vector is hard to obtain for different MOPs in practical applications.

Crossover operator often plays an important role in the search process of EAs and is widely used in numerical experiments. However, comparing to the mutation operator, the understanding of the effect of the crossover operator in EAs from theoretical analysis aspect is limited (Sudholt 2017). There are several studies have contributed to this issue. For single optimization problems, these researches (Jansen and Wegener 2002; Kötzing, Sudholt, and Theile 2011; Lehre and Yao 2011; Doerr et al. 2013; Doerr, Doerr, and Ebel 2015; Sudholt 2017; Doerr and Doerr 2018; Dang et al. 2018) have shown that crossover operator can speed up the search. Specifically, they proved that EAs enabled crossover outperform the disabled ones on some benchmark discrete optimization problems in term of the expected running time. As far as we know, the first running time analysis to show that the helpfulness of crossover on MOP is presented in (Neumann and Theile 2010). In (Qian, Yu, and Zhou 2013), the authors presented a running time analysis to show that crossover leads to better upper bounds than previous known result for LPTNO (LOTZ) and COCZ problems. These results indicate that crossover is helpful for EAs on optimizing some problems. However, whether it does help for decomposition-based MOEAs on optimizing MOPs has not yet been investigated from theoretical aspect.

In this paper, we present a running time analysis of a simple decomposition-based MOEA with crossover (MOEA/D-C) on four benchmark discrete optimization problems, i.e., COCZ, LPTNO, Dec-obj-MOP and Plateau-MOP. The upper bounds of expected running time obtained by MOEA/D-C are lower than the MOEA/D-M an order of n, where n is the size of decision variable. These bounds are better than or at least the best known ones. Moreover, there is an interesting thing worth mentioning that the MOEA/D-C only needs to decompose these MOPs instances into several subproblems according to a set of simple weight vectors while the MOEA/D-M needs to find $\Theta(n)$ optimally decomposed weight vectors. This theoretical study reveals that the use of crossover operator in MOEA/D framework can simplify the setting of weight vectors for different problems and make the algorithm more efficient. It provides insights in the design of decomposition-based MOEAs for MOPs in future research.

Algorithm and Problem

Analyzed Algorithm

An MOP can be formally defined as follows:

$$\begin{aligned} &\max \ \ F(x) = \big(f_1(x), \cdots, f_m(x)\big) \\ &s.t. \ \ x \in X, \end{aligned} \tag{1}$$

where X is the decision space and $x = (x_1, \cdots, x_n)$ is the decision variable, $F : X \to R^m$ consists of m functions and R^m is the objective space. Since the m objectives of (1) are often mutually conflicting, there is no a solution x that can maximize all objectives simultaneously. Instead, these conflicting objectives give rise to a set of eclectic optimal solutions. For $x_1, x_2 \in X$, we say that x_1 dominates x_2, denoted as $x_1 \succ x_2$, if and only if $f_i(x_1) \geq f_i(x_2)$ for all $i = 1, \cdots, m$ and $f_i(x_1) > f_i(x_2)$ for at least one index i. A solution $x^* \in X$ is Pareto optimal if there is no solution x such that $x \succ x^*$. The set of all Pareto optimal solutions is called *Pareto optimal set* (PS) and the set of all objective vectors corresponding to PS is called the *Pareto front* (PF).

In the MOEA/D framework, an MOP is decomposed into some scalar optimization subproblems according to the decomposition approach and weight vectors. As in the first implemented algorithm of the MOEA/D framework in (Zhang and Li 2007), the Tchebycheff decomposition approach and simplex-lattice weight vector generation method are used in MOEA/D-C as well as in the MOEA/D-M. Given an MOP defined in (1), the scalar optimization subproblem generated by the Tchebycheff approach is

$$\begin{aligned} &\min \ \ g(x|\lambda) = \max_{1 \leq i \leq m} \{\lambda_i |f_i(x) - z_i^*|\} \\ &s.t. \ \ x \in X, \end{aligned} \tag{2}$$

where $\lambda = (\lambda_1, \cdots, \lambda_m)$ is the weight vector, i.e., $\lambda_i \geq 0$ for $i = 1, \cdots, m$ and $\sum_{i=1}^m \lambda_i = 1$, and $z^* = (z_1^*, \cdots, z_m^*)$ denotes the reference point, i.e., $z_i^* = \max\{f_i(x)|x \in X\}$.

By altering the weight vector, the Tchebycheff approach generates different scalar optimization subproblems in form of (2) for an MOP defined in (1). Let H be a positive integer. The simplex-lattice design method generates weight vectors by taking m values from $\{0/H, 1/H, \cdots, H/H\}$ such that $\sum_{i=1}^m \lambda_i = 1$. Thus, for an MOP with m objectives and integer H, there are $N = C_{H+m-1}^{m-1}$ weight vectors, and each one corresponds to a scalar optimization subproblem.

After decomposition, for each subproblem the MOEA/D framework first selects the T closest subproblems to form its neighbor set, where T is an input parameter called *neighbor size*. The distance are measured by the Euclidean distance between their weight vectors. It then controls a population of size N to cooperatively solve the N scalar optimization subproblems by using the neighborhood-based coevolution.

A description of the presented MOEA/D-C is given in Algorithm 1. Different from the MOEA/D-M only using mutation to create the offspring, the MOEA/D-C allows to use the crossover operator in Step 7. Observe that the important decomposition and neighborhood-based coevolution features of the MOEA/D framework are retained in Algorithm 1.

Analyzed Problems

We denote by $\|x\|_1$ the number of 1-bits in solution x. The four analyzed MOPs in this paper are defined as follows.

Definition 1 (COCZ)**.** *The pseudo-Boolean function $COCZ : \{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

$$COCZ(x) = \big(\|x\|_1, n - \|x\|_1\big).$$

This instance is an extension of the COUNTONES problem called count ones count zeroes (COCZ). The above definition of COCZ is used in (Li et al. 2016), which is lightly different from the definition in (Laumanns, Thiele, and Zitzler 2004). However, this change does not effect the running time analysis since it only extends the size of PF from $0.5n+1$ to $n+1$ and will not change the upper bound of the

Algorithm 1 MOEA/D-C

Input: An MOP with m objectives, stop criterion, parameter H, the number of scalar optimization subproblems N, weight vectors $\{\lambda^1, \cdots, \lambda^N\}$ and neighbor size T.

Output: A Pareto optimal solution set P.

Initialization: The Pareto optimal solution set $P = \emptyset$. For each subproblem $g(x|\lambda^k)$, $k = 1, \cdots, N$, select the T closest subproblems into its neighbor set B_k according to the Euclidean distance between their weight vectors. Generate a solution $x_k \in \{0,1\}^n$ uniformly at random for each subproblem $g(x|\lambda^k)$. Set the reference point $z = (z_1, \cdots, z_m)$, where $z_i = \max\{f_i(x_k)\}$ for $k = 1, \cdots, N$. Let S_k denote the set of solutions corresponding to subproblems in B_k.
while stop criterion is not satisfied **do**
  for each subproblem $g(x|\lambda^k)$, $k = 1, \cdots, N$ **do**
    if $rand > p_c$ **then**
      Create two new solutions x'_k, x''_k for the kth subproblem by using the mutation operator to x_k.
    else
      Create two new solutions x'_k, x''_k for the kth subproblem by using the crossover operator on x_k and a solution in $S_k \setminus x_k$.
    end if
    Update z: for each z_i, if $\max\{f_i(x'_k), f_i(x''_k)\} > z_i$, set $z_i = \max\{f_i(x'_k), f_i(x''_k)\}$, $i = 1, \cdots, m$.
    Update S_k: for each solution x_j in S_k, if $\min\{g(x'_k|\lambda^j), g(x''_k|\lambda^j)\} \leq g(x_j|\lambda^j)$, replace x_j with the better one in $\{x'_k, x''_k\}$.
    Update P: remove all solutions dominated by x'_k or x''_k from P. If x'_k or x''_k is not dominated by solutions in P, add it into P.
  end for
end while

expected running time. A similar definition of this instance called ONEMINMAX is presented in (Giel and Lehre 2010). All solutions of COCZ are Pareto optimal and the distribution of elements in the PF is shown in Figure 1 (red points).

Definition 2 (WLPTNO). *The pseudo-Boolean function* $WLPTNO : \{-1,1\}^n \to \mathbb{R}^2$ *is defined as follows:*

$$WLPTNO(x) = \Big(\sum_{i=1}^{n} w_i \prod_{j=1}^{i}(1+x_i), \sum_{i=1}^{n} v_i \prod_{j=i}^{n}(1-x_i)\Big),$$

where $w_i, v_i > 0$ *for* $i = 1, \cdots, n$.

This instance is defined in (Qian, Yu, and Zhou 2013). The abbreviation WLPTNO stands for *Weighted Leading Positive Ones Trailing Negative Ones*. It can be considered as an extension of LOTZ (Leading Ones Trailing Zeroes) by shifting the decision space from $\{0,1\}^n$ to $\{-1,1\}^n$, and adding weights w_i and v_i for each leading positive one and trailing negative one bits, respectively. However, it mostly has very different properties from LOTZ. As in analyzing the MOEA/D-M (Li et al. 2016), we set $w_i = v_i = 1$ and denote this case as LPTNO. Note that the obtained upper bound of the expected running time in this paper is also true for other setting of weights. The PS of LPTNO has $n+1$ elements. As shown in Figure 2, unlike the COCZ, the elements of the PF are not evenly distributed.

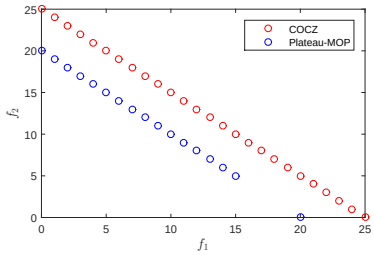

Figure 1: Pareto front of the COCZ with $n = 25$ and Plateau-MOP with $n = 20$.

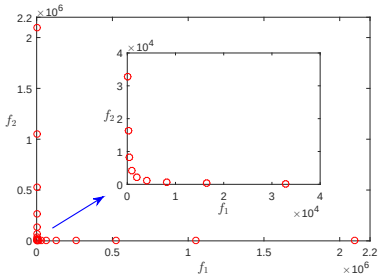

Figure 2: Pareto front of the LPTNO with $n = 20$.

Definition 3 (Dec-obj-MOP). *The pseudo-Boolean function* $Dec\text{–}obj\text{–}MOP : \{0,1\}^n \to \mathbb{N}^2$ *is defined as follows:*

$$Dec\text{–}obj\text{–}MOP(x) = \big(f_1(x), f_2(x)\big),$$

where

$$f_1(x) = n + 1 - \|x\|_1 \mod (n+1),$$
$$f_2(x) = n + \|x\|_1 \mod (n+1).$$

This instance is defined in (Li et al. 2016). There is deceptive property in the search space of f_2. The location of the global optimum is very far from the local optimum in the search space, and the fitness-based search will be mostly guided to the local optimum. Thus, these kinds of functions are deceptive and hard to solve. The expected running time to obtain the optimal solution for some subproblems of Dec-obj-MOP is $\Omega(n^n)$ (Li et al. 2016). The PS and the PF of Dec-obj-MOP are the same as the COCZ.

Definition 4 (Plateau-MOP). *Let* 1^i0^{n-i} *denote that there are* i *leading ones and* $n-i$ *trailing zeroes in the solution* x. *The pseudo-Boolean function* $Plateau\text{–}MOP : \{0,1\}^n \to \mathbb{N}^2$ *is defined as follows:*

$$Plateau\text{–}MOP(x) = \big(f_1(x), f_2(x)\big),$$

where

$$f_1(x) = \begin{cases} n & \text{if } x = 1^n, \\ 3n/4 & \text{if } x = 1^i0^{n-i}, 3n/4 < i < n, \\ i & \text{if } x = 1^i0^{n-i}, 0 \leq i \leq 3n/4, \\ -\|x\|_1 & otherwise, \end{cases}$$

$$f_2(x) = \begin{cases} n-i & \text{if } x = 1^i0^{n-i}, 0 \le i \le n, \\ -\|x\|_1 & otherwise. \end{cases}$$

This instance is defined in (Li et al. 2016). For convenience, we hereafter assume that $\frac{n}{4}$ is integral. Although neither objective functions in Plateau-MOP is deceptive, it is hard to solve for SEMO. The expected running time of SEMO on Plateau-MOP is $\Omega(n^{0.25n})$ (Li et al. 2016).

For f_1, there is a plateau which x is in form of 1^i0^{n-i} for $i \in [\frac{3}{4}n, n-1]$. So solutions of Plateau-MOP in this range are dominated by the solution $1^{3n/4}0^{n/4}$ since the fitness value of f_2 becomes worse with increasing $\|x\|_1$. The PS of Plateau-MOP has $\frac{3}{4}n+2$ elements, which are 1^i0^{n-i} for $i \in \{[0, \frac{3}{4}n]\} \cup \{n\}$. The elements on PF of Plateau-MOP is shown in Figure 1 (blue points). Note that the point $(n, 0)$ is an outlier in the objective space.

Running Time Analysis

In this section, we present the running time analysis of Algorithm 1 on the four above MOPs. For parameters in Algorithm 1, in this analysis we set $H = 2, T = m$ and $p_c = 0.5$. So the set of decomposed weight vectors is $\{\lambda^1 = (0,1), \lambda^2 = (0.5, 0.5), \lambda^3 = (1,0)\}$ and the number of scalar optimization subproblems N (also the population size) is 3. For the variation operators in Step 5 and 7, we use the standard bit mutation and one-point crossover, respectively. Thus Algorithm 1 requires six fitness evaluations in an iteration. For ease of expression, we assume that the optimal value of each objective for these problems have been known in the following analysis. Note that this assumption does not effect the expected running time of Algorithm 1 on these problems, because in each iteration any new solution with better fitness value for the objective will be updated immediately in Step 9 and can be accepted in Step 10.

Analysis on COCZ

For COCZ, a set of Pareto optimal solutions corresponding to the PF is $\{0^n, 0^{*n-1}1^*, \cdots, 1^n\}$, where $0^{*i}1^{*j}$ indicates that there are i 0-bits and j 1-bits randomly distributed in the solution. The best known upper bound of expected running time on COCZ is $O(n \log n)$ (Qian, Yu, and Zhou 2013). In (Li et al. 2016), the authors proved that the evenly distributed weight vectors, i.e., $\lambda^k = (\lambda_1^k, 1-\lambda_1^k), \lambda_1^k = \frac{k}{n}, k = 0, \cdots, n$, produce $n+1$ subproblems that are mapped one-to-one to the points in the PF. So the MOEA/D-M with $H = n$ obtains a set of solutions to cover the PF by solving the $n+1$ subproblems. Thus, the expected running time of the MOEA/D-M on COCZ is $O(n^2 \log n)$ since it solves each subproblem in expected running time $O(n \log n)$.

These decomposed subproblems for COCZ in Algorithm 1 (Tchebycheff approach) are listed as follows:

$$\min\ g(x|\lambda^1) = \|x\|_1, \tag{3}$$

$$\min\ g(x|\lambda^2) = \max\{0.5(n - \|x\|_1), 0.5\|x\|_1\}, \tag{4}$$

$$\min\ g(x|\lambda^3) = n - \|x\|_1. \tag{5}$$

In the following, we prove that the expected running time of Algorithm 1 on COCZ is $O(n \log n)$. The analysis consists of two phases. In the first phase, by neglecting the promotion of the crossover operator and coevolution, we prove that Algorithm 1 can find an optimal solution for each subproblem by using the mutation operator in expected running time $O(n \log n)$. In the second phase, we prove that by using the crossover operator to optimal solutions for the three subproblems, Algorithm 1 obtains a set of solutions to cover the PF in expected running time $O(n \log n)$.

Lemma 1. *For COCZ, Algorithm 1 finds an optimal solution for each subproblem in expected running time $O(n \log n)$.*

Proof. The optimization process of these functions is similar to optimize the well-known ONEMAX function. For (3) (or (5)), the fitness value becomes better when increasing the 0-bits (1-bits) in solution x and the optimal solution is 0^n (1^n). Let i denote the number of 0-bits (1-bits) in the current solution. For Algorithm 1, the mutation operator produces a better solution for (3) (or (5)) from the current one with probability $\frac{(n-i)}{2} \cdot \frac{1}{n} \cdot (1-\frac{1}{n})^{n-1} \ge \frac{n-i}{2en}$, since it happens if one of these 0-bits (1-bits) is flipped while keeping other bits unchanged. Recall that $p_c = 0.5$. Thus, according to the fitness level approach (Sudholt 2013; Jansen 2013), Algorithm 1 finds the optimal solution for (3) and (5) in expected running time $O(n \log n)$. For (4), the optimal solution $0^{*n/2}1^{*n/2}$ is close to the initial solution. If $\|x\|_1 < 0.5n$, the fitness value becomes better when increasing $\|x\|_1$. If $\|x\|_1 > 0.5n$, decreasing $\|x\|_1$ will lead to a better fitness value. Thus, Algorithm 1 finds a better solution for (4) in an iteration with probability at least $\frac{n-i}{2en}$ and the expected running time is $O(n \log n)$. □

Theorem 1. *For COCZ, Algorithm 1 obtains a set of solutions to cover the PF in expected running time $O(n \log n)$.*

Proof. Since each subproblem selects two closest subproblems into B_k according to the Euclidean distance between their weight vectors, we have $S_1 = \{0^n, 0^{*n/2}1^{*n/2}\}$, $S_2 = \{0^{*n/2}1^{*n/2}, 0^n \text{ or } 1^n\}$ and $S_3 = \{1^n, 0^{*n/2}1^{*n/2}\}$ after solutions $0^n, 0^{*n/2}1^{*n/2}$ and 1^n have been found for (3), (4) and (5), respectively. Thus, the rest of Pareto solutions corresponding to PF can be partitioned into two disjoint sets, i.e., $R_1 = \{0^{*n-1}1^*, 0^{*n-2}1^{*2}, \cdots, 0^{*n/2+1}1^{*n/2-1}\}$ and $R_2 = \{0^{*n/2-1}1^{*n/2+1}, 0^{*n/2-2}1^{*n/2+2}, \cdots, 0^*1^{*n-1}\}$. In the following, we show that those solutions produced by using crossover operator on solutions in S_1 and S_3 cover R_1 and R_2 in expected running time $O(n \log n)$, respectively. Note that solutions produced by using crossover operator on solutions in S_2 and mutation operator to the three solutions during this phase can only accelerate the search process.

In each iteration, Algorithm 1 applies crossover to solutions in S_1 and S_3 with probability $p_c = 0.5$. Note that there are $0.5n$ 1-bits in solution $0^{*n/2}1^{*n/2}$ and the crossover point is selected from 1 to $n-1$ uniformly. Let l_1 and l_2 denote the position of the first and the last 1-bit in solution $0^{*n/2}1^{*n/2}$. We have $l_2 - l_1 \ge 0.5n$. We call a crossover a success if it produces solutions in R_1 and at least one of them have not been produced in previous crossovers. For S_1, if the crossover point is located in (l_1, l_2), it produces two solutions in R_1 since the other solution in S_1 is 0^n. Hence, Algorithm 1 needs at most $0.25n$ successes to obtain a set

of solutions to cover R_1 because each success produces two new solutions in R_1 and $|R_1| = 0.5n-1$. Observe that there are one or two points in (l_1, l_2) corresponding to each success. Thus, a success happens in an iteration with probability at least $\frac{0.25n-i}{2(n-1)}$, where i denotes the number of successes so far. Therefore, the expected running time of Algorithm 1 obtained a set of solutions to cover R_1 is upper bounded by

$$\sum_{i=0}^{0.25n-1} \frac{2(n-1)}{0.25n-i} \leq 2n \sum_{j=1}^{0.25n} \frac{1}{j} = O(n \log n).$$

Similarly, we have that Algorithm 1 obtains a set of solutions to cover R_2 in expected running time $O(n \log n)$.

Therefore, combined with Lemma 1, Algorithm 1 obtains a set of solutions S to cover the whole PF, i.e., $PF \subseteq F(S)$, of COCZ in expected running time $O(n \log n)$. □

Analysis on LPTNO

For LPTNO, the set of Pareto optimal solutions corresponding to the PF is $\{1^n, 1^{n-1}(-1), \cdots, (-1)^n\}$, and the best known upper bound of expected running time is $O(n^2)$ (Qian, Yu, and Zhou 2013). In (Li et al. 2016), the authors proved that the weight vectors, $\lambda^k = (\lambda_1^k, 1-\lambda_1^k), \lambda_1^k = \frac{\sum_{j=1}^{k} 2^{n+1-j}}{\sum_{j=k+1}^{n} 2^j + \sum_{j=1}^{k} 2^{n+1-j}}, k = 0, \cdots, n$, produces a set of subproblems which are mapped one-to-one to the points in the PF. So the MOEA/D-M with $H = n$ obtains a set of solutions to cover the PF by solving the decomposed $n+1$ subproblems. Thus, the expected running time of the MOEA/D-M on LPTNO is $O(n^3)$ since it solves each subproblem in expected running time $O(n^2)$. Note that this set of weight vectors are not evenly distributed since most of λ^k fall nearby $(0.5, 0.5)$.

These decomposed subproblems for LPNTO in Algorithm 1 (Tchebycheff approach) are listed as follows:

$$\min\ g(x|\lambda^1) = 2^{n+1} - 2 - \sum_{i=1}^{n} \prod_{j=i}^{n} (1-x_i), \tag{6}$$

$$\min\ g(x|\lambda^2) = \max\big\{2^n - 1 - 0.5 \sum_{i=1}^{n} \prod_{j=1}^{i} (1-x_i), \; 2^n - 1 - 0.5 \sum_{i=1}^{n} \prod_{j=i}^{n} (1+x_i)\big\}, \tag{7}$$

$$\min\ g(x|\lambda^3) = 2^{n+1} - 2 - \sum_{i=1}^{n} \prod_{j=1}^{i} (1+x_i). \tag{8}$$

Lemma 2. *For LPTNO, Algorithm 1 finds the optimal solution for each subproblem in expected running time $O(n^2)$.*

Proof. The optimization process of functions (6), (7) and (8) is similar to optimize the well-known LEADINGONES function. For (8) (or (6)), the fitness value becomes better when increasing the leftmost 1-bits (rightmost (-1)-bits) in x and the optimal solution is 1^n ($(-1)^n$). So in Algorithm 1 the mutation operator produces a better solution for (8) (or (6)) from the current one with probability $\frac{1}{2} \cdot \frac{1}{n} \cdot (1-\frac{1}{n})^{n-1} \geq \frac{1}{2en}$, because its fitness value will decrease when the leftmost (-1)-bit (rightmost 1-bit) is flipped while keeping other bits unchanged. Thus, according to the fitness level approach, the expected running time of Algorithm 1 to find the optimal solution for (8) (or (6)) is $O(n^2)$ since there are at most n bits to be flipped. For (7), any new solution where the minimum number of leading 1-bits and trailing (-1)-bits is non-decreased is accepted in the iteration. If the optimal solution is not found, there must exist at least a bit which is flipped to produce an accepted solution, and thus a new solution is accepted with probability at least $\frac{1}{2en}$. Note that if the difference between the numbers of leading 1-bits and trilling (-1)-bits is 1, there is a small plateau, which consists of two points and needs at most additional expected time $O(n)$ to leave since it needs two random walks on the plateau in expectation. Therefore, Algorithm 1 also finds the optimal solution $1^{n/2}(-1)^{n/2}$ for (7) in expected running time $O(n^2)$. □

Theorem 2. *For LPTNO, Algorithm 1 obtains a set of solutions to cover the PF in expected running time $O(n^2)$.*

Proof. Because Algorithm 1 selects two closest subproblems into B_k according to the Euclidean distance between their weight vectors for each subproblem, we have that $S_1 = \{(-1)^n, 1^{n/2}(-1)^{n/2}\}$, $S_2 = \{1^{n/2}(-1)^{n/2}, (-1)^n \text{ or } 1^n\}$ and $S_3 = \{1^n, 1^{n/2}(-1)^{n/2}\}$ after solutions $(-1)^n, 1^{n/2}(-1)^{n/2}$ and 1^n have been found for subproblems (6), (7) and (8), respectively. The rest of Pareto optimal solutions corresponding to PF can be partitioned into two disjoint sets, that is, $R_1 = \{1(-1)^{n-1}, 1^2(-1)^{n-2}, \cdots, 1^{n/2-1}(-1)^{n/2+1}\}$ and $R_2 = \{1^{n/2+1}(-1)^{n/2-1}, 1^{n/2+2}(-1)^{n/2-2}, \cdots, 1^{n-1}(-1)\}$. We next show that those solutions produced by using crossover operator on solutions in S_1 and S_3 cover R_1 and R_2 in expected running time $O(n \log n)$, respectively.

For the two solutions in S_1, the leftmost $0.5n-1$ bits have different value. If and only if one of these bits is selected, the crossover creates a unique solution in R_1. Thus, to obtain a set of solutions to cover R_1, the crossover operator needs to select each of these bits at least once. Let i denote the number of these bits that have been selected, a new bit is selected by the crossover in a new iteration with probability $\frac{0.5n-1-i}{2(n-1)}$. Recall that $p_c = 0.5$. Thus, Algorithm 1 obtains a set of solutions to cover R_1 in expected running time

$$\sum_{i=0}^{0.5n-2} \frac{2(n-1)}{0.5n-1-i} \leq 2n \sum_{j=1}^{0.5n} \frac{1}{j} = O(n \log n).$$

Similarly, we have that Algorithm 1 obtains a set of solutions to cover R_2 in expected running time $O(n \log n)$.

Therefore, combined with Lemma 2, Algorithm 1 obtains a set of solutions to cover the whole PF of LPNTO in expected running time $O(n^2)$. □

Analysis on Dec-obj-MOP

For Dec-obj-MOP, the set of Pareto optimal solutions corresponding to the PF is $\{0^n, 1^n, 0^*1^{*n-1}, \cdots, 0^{*n-1}1^*\}$, and the best known upper bound of expected running time is

$O(n^2 \log n)$ (Li et al. 2016). The evenly distributed weight vectors, i.e., $\lambda^k = (\lambda_1^k, 1-\lambda_1^k), \lambda_1^k = \frac{k}{n}, k = 0, \cdots, n$, also produces a set of subproblems that are mapped one-to-one to the points in the PF. However, different from COCZ that every decomposed subproblems can be solved in expected running time $O(n \log n)$, the expected fitness evaluations to find the optimal solution for the subproblem corresponding to weight vector $(0, 1)$, i.e., (9), is $\Omega(n^n)$. But the expected running time of the MOEA/D-M on Dec-obj-MOP is $O(n^2 \log n)$, because the optimal solution for (9) can be obtained and added into P by other subproblems in expected running time $O(n \log n)$.

For Dec-obj-MOP, these decomposed subproblems in Algorithm 1 are listed as follows:

$$\min\ g(x|\lambda^1) = n - f_2(x), \tag{9}$$

$$\min\ g(x|\lambda^2) = \max \frac{1}{2}\{(n - f_1(x)), (n - f_2(x))\}, \tag{10}$$

$$\min\ g(x|\lambda^3) = n - f_1(x), \tag{11}$$

where $f_1(x)$ and $f_2(x)$ are the same as in the Definition 3.

Lemma 3. *For Dec-obj-MOP, Algorithm 1 finds an optimum for subproblems (10) and (11) and the local optimum for subproblem (9) in expected running time $O(n \log n)$.*

Proof. For (11), the fitness value becomes better when decreasing $\|x\|_1$ if $\|x\|_1 > 1$. Thus, similar to optimize the ONEMAX, starting from any uniformly random solution Algorithm 1 finds the optimal solution $0^{*n-1}1^*$ for (11) in expected running time $O(n \log n)$. For (10), the fitness value becomes better when increasing $\|x\|_1$ if $\|x\|_1 < 0.5n$ and decreasing $\|x\|_1$ if $\|x\|_1 > 0.5n$. Thus, starting from any initial solution Algorithm 1 finds the optimal solution $0^{*n/2}1^{*n/2}$ for (10) in expected running time $O(n \log n)$. For (9), the optimal solution is 0^n. There is deceptive property in the search space since the fitness value becomes better when increasing $\|x\|_1$ if $\|x\|_1 > 0$. Since the expected number of 1-bits in an initial solution is $\frac{n}{2} > 0$, by Chernoff bound, the search will be guided to the solution 1^n with probability $1 - e^{-\Omega(n)}$. Thus, Algorithm 1 finds the local optimum 1^n for (9) in expected running time $O(n \log n)$. □

Although the mutation operator cannot find the optimal solution for (9) in polynomial running time, we next show that Algorithm 1 still obtains a set of solutions to cover the PF of Dec-obj-MOP in expected running time $O(n \log n)$.

Theorem 3. *For Dec-obj-MOP, Algorithm 1 obtains a set of solutions to cover the PF in expected running time $O(n \log n)$.*

Proof. For Dec-obj-MOP, we have $S_1 = \{1^n, 0^{*n/2}1^{*n/2}\}$, $S_2 = \{0^{*n/2}1^{*n/2}, 1^n \text{ or } 0^{*n-1}1^*\}$ and $S_3 = \{0^{*n-1}1^*, 0^{*n/2}1^{*n/2}\}$ after solutions $1^n, 0^{*n/2}1^{*n/2}$ and $0^{*n-1}1^*$ have been found for subproblems (9), (10) and (11), respectively. We partition the rest of Pareto optimal solutions corresponding to PF into three disjoint sets, i.e., $R_1 = \{0^{*n/2-1}1^{*n/2+1}, 0^{*n/2-2}1^{*n/2+2}, \cdots, 0^{*1}1^{*n-1}\}$, $R_2 = \{0^{*n-2}1^{*2}, 0^{*n-3}1^{*3}, \cdots, 0^{*n/2+1}1^{*n/2-1}\}$ and $R_3 = \{0^n\}$.

Since solutions produced by using crossover on solutions in S_1 are the same as the case in COCZ that each success creates two new solutions in R_1, Algorithm 1 obtains a set of solutions to cover R_1 in expected running time $O(n \log n)$.

For R_2, there is a bit difference with the case in COCZ, because there is a 1-bit in the solution $0^{*n-1}1^* \in S_3$ and it will be copied into an offspring in the one-point crossover. In such way, some successes may only produce a new solution in R_2 since the other one is a duplication of previous solutions. Thus, for S_3 the crossover operator may take $0.5n - 2$ successes to obtain a set of solutions to cover R_2 since $|R_2| = 0.5n - 2$. Observe that there is at least one point in (l_1, l_2) corresponding to a success, where l_1 and l_2 denote the position of the first and the last 1-bit in the solution $0^{*n/2}1^{*n/2}$. For Algorithm 1, the probability of a success happening in an iteration is $\frac{0.5n-2-i}{2(n-1)}$, where i denotes the number of successes so far. Recall that $p_c = 0.5$. Thus, the expected running time of the crossover operator produced a set of solutions to cover R_2 is upper bounded by

$$\sum_{i=0}^{0.5n-3} \frac{2(n-1)}{0.5n-2-i} \leq 2n \sum_{j=1}^{0.5n} \frac{1}{j} = O(n \log n).$$

For the solution 0^n in R_3, it may be created by using crossover operator to solutions in S_2 and S_3. However, the probability of this event is complex to compute and the lower bound seems to be very small. We next show that it can still be found by Algorithm 1 in expected running time $O(n)$. Note that after the optimal solution $0^{*n-1}1^*$ has been found for (11), it will be kept by this subproblem forever. Thus, in later mutations it will be always used to create an offspring, and the offspring is 0^n with probability at least $\frac{1}{en}$ since it happens when the 1-bit is flipped and all 0-bits are kept unchanged. Thus, the solution 0^n is created and added into P by the mutation operator in expected running time $O(n)$.

Therefore, combined with Lemma 3, Algorithm 1 obtains a set of solutions to cover the PF of Dec-obj-MOP in expected running time $O(n \log n)$. □

Analysis on Plateau-MOP

The PF of Plateau-MOP is $\{(0, n), (1, n-1), \cdots, (3n/4, n/4), (n, 0)\}$ and the set of Pareto optimal solutions corresponding to PF is $\{0^n, 10^{n-1}, \cdots, 1^{3n/4}0^{n/4}, 1^n\}$. The best known upper bound of expected running time on Plateau-MOP is $O(n^3)$ (Li et al. 2016). The evenly distributed weight vector, i.e., $\lambda^k = (\lambda_1^k, 1-\lambda_1^k), \lambda_1^k = \frac{k}{n}, k = 0, \cdots, n$, produces a set of subproblems that are mapped one-to-one to the points on PF. So MOEA/D-M with $H = n$ finds a set of solutions to cover the PF by solving the decomposed $\Theta(n)$ subproblems. Thus, the expected running time of MOEA/D-M on Plateau-MOP (exclude the outlier point $(n, 0)$) is $O(n^3)$ since it solves each subproblem in expected running time $O(n^2)$ (Li et al. 2016).

For Plateau-MOP, these decomposed subproblems in Al-

gorithm 1 are listed as follows:

$$\min\ g(x|\lambda^1) = n - f_2(x), \tag{12}$$

$$\min\ g(x|\lambda^2) = \max \frac{1}{2}\{(n - f_1(x)), (n - f_2(x))\}, \tag{13}$$

$$\min\ g(x|\lambda^3) = n - f_1(x), \tag{14}$$

where $f_1(x)$ and $f_2(x)$ are the same as in the Definition 4.

Lemma 4. *For Plateau-MOP, Algorithm 1 finds the optimal solution for subproblems (12) and (13) and a suboptimal solution for subproblem (14) in expected running time $O(n^2)$.*

Proof. For (12), starting from any initial solution x, the optimization process first tries to minimize the 1-bits in it or directly jump to a solution with form of 1^i0^{n-i}, then it tries to flip the rightmost 1-bit into 0-bit until the optimal solution 0^n is found. The first part is equivalent to optimizing the ONEMAX ($O(n \log n)$) and the second part is equivalent to optimizing the LEADINGONES ($O(n^2)$). Thus, Algorithm 1 finds the optimal solution for (12) in expected running time $O(n^2)$. For (13), the optimal solution is $1^{n/2}0^{n/2}$. First, by Chernoff bound, we have that any initial solution turns into the form of 1^i0^{n-i} in expected running time $O(n \log n)$, where $i \leq \frac{3n}{4}$ with probability $1 - e^{-\Omega(n)}$. Second, from any solution in form of 1^i0^{n-i} for $i \leq \frac{3n}{4}$, the fitness becomes better when increasing i if $i < 0.5n$ and decreasing i if $i > 0.5n$, which happens with probability at least $\frac{1}{en}$. Thus, Algorithm 1 finds the optimal solution $1^{n/2}0^{n/2}$ for (13) in expected running time $O(n^2)$. For (14), in the objective space there is a plateau (suboptimal solutions), which is in form of 1^i0^{n-i} for $i \in [\frac{3}{4}n, n-1]$. Thus, Algorithm 1 finds the suboptimal solution $1^{3n/4}0^{n/4}$ in expected running time $O(n^2)$ from any solution in form of 1^i0^{n-i}. After finding the suboptimal solution $1^{3n/4}0^{n/4}$, it randomly walks on the plateau and accepts any new solution which is in form of 1^i0^{n-i} with $i \geq \frac{3}{4}n$. □

Theorem 4. *For Plateau-MOP, Algorithm 1 obtains a set of solutions to cover the PF (except the outlier point) in expected running time $O(n^2)$.*

Proof. After Algorithm 1 finds the optimal solution for subproblems (12) and (13) and a suboptimal solution for (14) (assume that is $1^{3n/4}0^{n/4}$ for convenience), we have $S_1 = \{0^n, 1^{n/2}0^{n/2}\}$, $S_2 = \{1^{n/2}0^{n/2}, 0^n \text{ or } 1^{3n/4}0^{n/4}\}$ and $S_3 = \{1^{3n/4}0^{n/4}, 1^{n/2}0^{n/2}\}$. Thus, except the outlier point, the rest of Pareto optimal solutions corresponding to PF can be partitioned into two disjoint sets, that is, $R_1 = \{1^{n/2-1}0^{n/2+1}, 1^{n/2-2}0^{n/2+2}, \cdots, 10^{n-1}\}$ and $R_2 = \{1^{n/2+1}0^{n/2-1}, 1^{n/2+2}0^{n/2-2}, \cdots, 1^{3n/4}0^{n/4}\}$.

For the two solutions in S_1, the leftmost $0.5n-1$ bits have different value. If and only if one of these bits is selected, the crossover creates a unique solution in R_1. Thus, to obtain a set of solutions to cover R_1, the crossover phase needs to select each of these bits once. Let i denote the number of these bits that have been selected, a new bit is selected by the crossover operator in the next iteration with probability $\frac{0.5n-1-i}{2(n-1)}$. Recall that $p_c = 0.5$. Thus, the expected running time of Algorithm 1 obtained a set of solutions to cover R_1 is upper bounded by

$$\sum_{i=0}^{0.5n-2} \frac{2(n-1)}{0.5n-1-i} \leq 2n \sum_{j=1}^{0.5n} \frac{1}{j} = O(n \log n).$$

Similarly, for R_2 there are $\frac{n}{4}$ Pareto optimal solutions, and a new solution is created in an iteration by applying crossover on solutions in S_3 with probability $\frac{0.25n-i}{2(n-1)}$. Thus, we also have that Algorithm 1 obtains a set of solutions to cover R_2 in expected running time $O(n \log n)$.

Therefore, combined with Lemma 4, Algorithm 1 obtains a set of solutions to cover the PF (except the outlier point) of Plateau-MOP in expected running time $O(n^2)$. □

Experimental Verification

As shown in Figure 3, the curves of average fitness evaluations of numerical experiments (dash lines) on four problems are approximately in the order of the corresponding theoretical bounds (solid lines), respectively. For ease of observation, we plot curves of $25n^2$ and $25n \log n$ for bounds $O(n^2)$ and $O(n \log n)$, respectively. This result confirms the correctness of theoretical bounds obtained in our analysis.

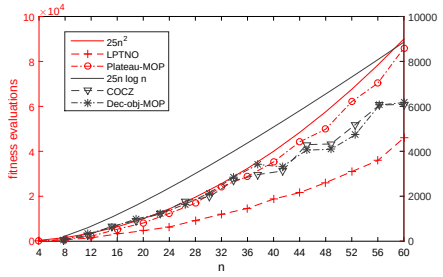

Figure 3: Curves of average fitness evaluations used in experiments, where MOEA/D-C runs 30 times for each n.

Conclusions

In this paper, we present a theoretical analysis for the effect of crossover in the MOEA/D framework. Our rigorous running time analysis shows that the MOEA/D-C working with several simple weight vectors can obtain a set of Pareto optimal solutions to cover the PF of four benchmark discrete optimization problems in expected running time better than the MOEA/D-M. This result suggests that the use of the crossover in the MOEA/D framework can simplify the complexity of the setting of the decomposed weight vectors and promote the efficiency of this kind of algorithms.

This study theoretically explains why some MOEAs based on decomposition in the literature work well in computational experiments and provides insights in design of decomposition-based MOEAs for MOPs in future research. In addition, it provides evidence for that crossover is helpful in design of EAs. To derive a better bound of MOEA/D-C on these problems, one needs to estimate the effect of the coevolution in the algorithm. This is an interesting and challenging study in future work.

References

Chiang, T. C., and Lai, Y. P. 2011. MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism. In *Evolutionary Computation*, 1473–1480.

Dang, D. C.; Friedrich, T.; Kötzing, T.; Krejca, M. S.; Lehre, P. K.; Oliveto, P. S.; Sudholt, D.; and Sutton, A. M. 2018. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation* 22(3):484–497.

Doerr, B., and Doerr, C. 2018. Optimal static and self-adjusting parameter choices for the (1+(λ,λ)) genetic algorithm. *Algorithmica* 80(5):1658–1709.

Doerr, B.; Johannsen, D.; Kötzing, T.; Neumann, F.; and Theile, M. 2013. More effective crossover operators for the all-pairs shortest path problem. *Theoretical Computer Science* 471:12–26.

Doerr, B.; Doerr, C.; and Ebel, F. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567:87–104.

Giagkiozis, I.; Purshouse, R. C.; and Fleming, P. J. 2014. Generalized decomposition and cross entropy methods for many-objective optimization. *Information Sciences* 282:363–387.

Giel, O., and Lehre, P. K. 2010. On the effect of populations in evolutionary multi-objective optimisation*. *Evolutionary Computation* 18(3):335.

Ishibuchi, H., and Murata, T. 1998. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28(3):392–403.

Ishibuchi, H.; Sakane, Y.; Tsukamoto, N.; and Nojima, Y. 2010. Simultaneous use of different scalarizing functions in MOEA/D. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, Usa, July*, 519–526.

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

Jansen, T., and Wegener, I. 2002. On the analysis of evolutionary algorithms - a proof that crossover really can help. *Algorithmica* 34(1):47–66.

Jansen, T. 2013. *Analyzing evolutionary algorithms: The computer science perspective*. Springer Science & Business Media.

Kötzing, T.; Sudholt, D.; and Theile, M. 2011. How crossover helps in pseudo-boolean optimization. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 989–996. ACM.

Laumanns, M.; Thiele, L.; Zitzler, E.; Welzl, E.; and Deb, K. 2002. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *International Conference on Parallel Problem Solving from Nature*, 44–53. Springer.

Laumanns, M.; Thiele, L.; and Zitzler, E. 2004. Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Transactions on Evolutionary Computation* 8(2):170–182.

Lehre, P. K., and Yao, X. 2011. Crossover can be constructive when computing unique input–output sequences. *Soft Computing* 15(9):1675–1687.

Li, Y.-L.; Zhou, Y.-R.; Zhan, Z.-H.; and Zhang, J. 2016. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 20(4):563–576.

Li, K.; Kwong, S.; Zhang, Q.; and Deb, K. 2017. Interrelationship-based selection for decomposition multiobjective optimization. *IEEE Transactions on Cybernetics* 45(10):2076–2088.

Murata, T., and Gen, M. 2002. Cellular genetic algorithm for multi-objective optimization. In *Proc. of the 4th Asian Fuzzy System Symposium*, 538–542. Citeseer.

Neumann, F., and Theile, M. 2010. How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. In *International Conference on Parallel Problem Solving from Nature*, 667–676. Springer.

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

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2013. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence* 204:99–119.

Sudholt, D. 2013. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 17(3):418–435.

Sudholt, D. 2017. How crossover speeds up building-block assembly in genetic algorithms. *Evolutionary Computation* 25(2):237–274.

Tanabe, R., and Ishibuchi, H. 2018. An analysis of control parameters of MOEA/D under two different optimization scenarios. *Applied Soft Computing* 70:22–40.

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

Wang, R.; Xiong, J.; Ishibuchi, H.; Wu, G.; and Zhang, T. 2017. On the effect of reference point in MOEA/D for multiobjective optimization. *Applied Soft Computing* 58.

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

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

Zhou, A.; Qu, B.-Y.; Li, H.; Zhao, S.-Z.; Suganthan, P. N.; and Zhang, Q. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1(1):32–49.