Running Time Analysis of MOEA/D on Pseudo-Boolean Functions

Zhengxin Huang, Yuren Zhou, Zefeng Chen, Xiaoyu He, Xinsheng Lai, and Xiaoyun Xia

***Abstract*—Decomposition-based multiobjective evolutionary algorithms (MOEAs) are a class of popular methods for solving the 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 running time analysis of a simple MOEA with mutation and crossover based on the MOEA/D framework (MOEA/D-C) on five pseudo-Boolean functions. Our rigorous theoretical analysis shows that by properly setting the number of subproblems, the upper bounds of expected running time of MOEA/D-C obtaining a set of solutions to cover the Pareto fronts (PFs) of these problems are apparently lower than those of the one with mutation-only (MOEA/D-M). Moreover, to effectively obtain a set of solutions to cover the PFs of these problem, MOEA/D-C only needs to decompose these MOPs into several subproblems with a set of simple weight vectors while MOEA/D-M needs to find $\Omega(n)$ optimally decomposed weight vectors. This result suggests that the use of crossover in decomposition-based MOEA can simplify the setting of weight vectors for different problems and make the algorithm more efficient. This paper provides some insights into the working principles of MOEA/D and explains why some existing decomposition-based MOEAs work well in computational experiments.**

***Index Terms*—Crossover, MOEA/D, multiobjective optimization problem (MOP), running time analysis.**

Manuscript received March 1, 2019; revised May 30, 2019; accepted July 22, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61773410, Grant 61673403, Grant 61703183, and Grant 61562071, and in part by the Public Welfare Technology Application Research Plan Project of Zhejiang Province under Grant LGG19F030010. This paper was recommended by Associate Editor H. Ishibuchi. *(Corresponding author: Yuren Zhou.)*

Z. Huang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China, and also with the Department of Computer Science and Information Technology, Youjiang Medical University for Nationalities, Baise 533000, China (e-mail: thenewyi@gmail.com).

Y. Zhou, Z. Chen, and X. He are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China, and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, Guangzhou 510006, China (e-mail: zhouyuren@mail.sysu.edu.cn; chzfeng@mail2.sysu.edu.cn; hxyokokok@foxmail.com).

X. Lai is with the School of Mathematics and Computer Science, Shangrao Normal University, Shangrao 334001, China (e-mail: xsl2001_jx@163.com).

X. Xia is with the College of Mathematics Physics and Information Engineering, Jiaxing University, Jiaxing 314001, China (e-mail: scutxxy@gmail.com).

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

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/TCYB.2019.2930979

I. Introduction

Multiobjective optimization problems (MOPs) involve the simultaneous optimization of multiple conflicting objectives. MOPs exist widely in real-world applications [1], [2]. The goal of solving MOP is to find a set of tradeoff solutions called Pareto-optimal solutions. Due to the population-based nature, evolutionary algorithms (EAs) are able to obtain multiple solutions in a single run, and multiobjective EAs (MOEAs) have been very popular in solving MOPs.

MOEAs are broadly classified into three categories, that is, domination-based, indicator-based, and decomposition-based [3]. The idea of decomposition for solving MOPs has been used in [4] and [5]. It becomes popular after the MOEA/D framework is presented by Zhang and Li [6]. In MOEA/D, an MOP is first decomposed into some scalar subproblems according to the decomposition approach and weight vectors. Then, all subproblems are solved simultaneously by employing an EA. In the past decade, there are vast studies contributed to improving and developing MOEAs based on the MOEA/D framework. For example, studies on designing novel, adaptive, and dynamical weight vectors generation methods [7]–[10], improved decomposition approaches [11]–[14], and selection mechanisms [15], [16]. For further decomposition-based MOEAs, the readers are advised to refer to [3].

Ishibuchi *et al.* [17] 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 the Pareto front (PF) shapes. Tanabe and Ishibuchi [18] 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 the WFG test problems is presented by Wang *et al.* [19]. The 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 be used for unknown cases.

Running time analysis is one of the most powerful theory tools to understand the performance and work principles of EAs. The first theoretical study for decomposition-based

MOEAs is presented by Li *et al.* [20]. They presented a running time analysis of a simple MOEA with mutation-only based on the MOEA/D framework, denoted by MOEA/D-M, on four discrete optimization problems. Their analyzed results show that if the optimally decomposed weight vectors are used, the expected running time of the MOEA/D-M on these problems is better than the simple evolutionary multiobjective optimizer (SEMO) [21], which has been theoretically analyzed on many MOPs in [22]–[28]. However, they also pointed out that the optimally decomposed weight vectors 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, compared to the mutation operator, the understanding of the effect of the crossover operator in EAs from theoretical analysis aspect is limited [29]. There are several studies contributed to this topic. For single-objective optimization problems, researches in [29]–[38] have shown that crossover operator can speed up the optimization process. Specifically, they proved that some EAs with crossover outperform those algorithms with only mutation on some benchmark discrete optimization problems in terms of the expected running time. As far as we know, the first running time analysis to show that the helpfulness of crossover on optimizing MOP is presented by Neumann and Theile [39]. Qian *et al.* [40] presented a running time analysis to show that crossover leads to upper bounds better than previous known results for LPTNO [leading ones trailing zeros (LOTZ)] and count ones count zeroes (COCZ) functions. These results indicate that crossover operator is helpful for EAs on optimizing some multiobjective problems. However, whether it helps decomposition-based MOEAs to optimize MOPs has not yet been investigated from the theoretical analysis aspect.

In this paper, we present a running time analysis of a simple decomposition-based MOEA with mutation and crossover (MOEA/D-C) on five benchmark discrete optimization problems, that is, LOTZ, COCZ, LPTNO, Dec-obj-MOP, and Plateau-MOP. This paper is based upon our conference paper accepted by AAAI 2019 (Z. Huang, Y. Zhou, Z. Chen, and X. He, "Running Time Analysis of MOEA/D with Crossover on Discrete Optimization Problem"). In that work, the analysis was presented for the special case that the number of the decomposed subproblems is three on the last four problems. Here, we extend the analysis to the general case that the number of the decomposed subproblems can be any natural number larger than three, and give a detailed analysis on the LOTZ problem. Our analysis results show that by properly setting the number of decomposed subproblems, the upper bounds of expected running time of MOEA/D-C are lower than those of MOEA/D-M an order of n, where n is the length of decision variable (bit-string). The theoretical upper bounds (including the best known upper bounds and the upper bounds obtained by MOEA/D-M and MOEA/D-C) on the five problems are shown in Table I. Moreover, to effectively obtain a set of solutions to cover the PF, MOEA/D-C only needs to

TABLE I
UPPER BOUNDS ON THE FIVE PROBLEMS

	MOEA/D-M	Best known upper bound	MOEA/D-C
COCZ	$O(n^2 \log n)$	$O(n \log n)$ [40]	$O(n \log n)$
LPTNO	$O(n^3)$	$O(n^2)$ [40]	$O(n^2)$
LOTZ	$O(n^3)$	$O(n^2)$ [40]	$O(n^2)$
Dec-obj-MOP	$O(n^2 \log n)$	$O(n^2 \log n)$ [20]	$O(n \log n)$
Plateau-MOP	$O(n^3)$	$O(n^3)$ [20]	$O(n^2)$

decompose an MOP into several subproblems according to a set of simple weight vectors while MOEA/D-M needs to find $\Omega(n)$ optimally decomposed weight vectors. Finally, computational experiments are conducted to verify the theoretical upper bounds obtained in our analysis and compare the efficiency of MOEA/D-M and MOEA/D-C on the aforementioned five problems. This paper reveals that the use of crossover operator in MOEA/D framework can simplify the setting of weight vectors and make the algorithm more efficient on handling some discrete problems.

The main contributions of this paper are summarized as follows.

1) This paper shows that MOEA/D-C, a simple MOEA/D implementation equipped with both mutation and crossover operators, is better than MOEA/D-M, equipped with mutation-only, on optimizing five benchmark problems from theoretical aspect. This result indicates that crossover operator plays an important role in MOEA/D.
2) Our analyses explain why MOEA/D can effectively produce a set of solutions to cover the PF of some discrete MOPs by only solving several scalarization subproblems, that is, the crossover operator can quickly create other Pareto-optimal solutions by recombining Pareto-optimal solutions found by these subproblems. This paper provides some insights into the working principles of MOEA/D.
3) This paper provides an idea to analyze the upper bound of the expected running time of decomposition-based MOEAs on optimizing discrete MOPs, that is, considering solutions created by the mutation and crossover operators in two phases.

II. PRELIMINARIES

A. MOEA/D Framework

Formally, an MOP can be defined as follows:

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

where X is the decision space, $\mathbf{x}$ is the decision variable, and $\mathbf{F} : X \to \mathbf{R}^m$ consists of m functions and $\mathbf{R}^m$ is the objective space. Since the objectives are often mutually conflicting, there is no solution that can maximize all objectives simultaneously. Instead, these conflicting objectives give rise to a set of tradeoff optimal solutions. For $\mathbf{x}', \mathbf{x}'' \in X$, we say that $\mathbf{x}'$ dominates $\mathbf{x}''$, denoted as $\mathbf{x}' \succ \mathbf{x}''$, if and only if $f_i(\mathbf{x}') \geq f_i(\mathbf{x}'')$ for all $i = 1, \ldots, m$ and $f_i(\mathbf{x}') > f_i(\mathbf{x}'')$ for at least one index i.

Algorithm 1 Framework of MOEA/D [6]

```
Input: An MOP with $m$ objectives in Eq. (1), stopping criterion, the number of subproblems $N$, weight vectors $\{\lambda^1, \cdots, \lambda^N\}$ and neighbor size $T$.
Output: EP.
Step 1 (Initialization):
The Pareto optimal solution set $EP = \emptyset$. For each subproblem $g(\mathbf{x}|\lambda^k)$, $k = 1, \cdots, N$, select the $T$ closest subproblems to form its neighbor set $B_k$ according to the Euclidean distance between their weight vectors. Generate a solution $\mathbf{x}_k$ for each subproblem $g(\mathbf{x}|\lambda^k)$. Set the reference point $\mathbf{z} = (z_1, \cdots, z_m)$, where $z_j = \max\{f_j(\mathbf{x}_k), k = 1, \cdots, N\}$ for $j = 1, \cdots, m$.
Step 2 (Update):
For $i = 1, \cdots, N$ do
    1) Reproduction: Randomly select two indices $k$ and $l$ from $B_i$, and generate a new solution $\mathbf{x}'$ from parents $\mathbf{x}_k$ and $\mathbf{x}_l$ by applying genetic operators.
    2) Repair: Repair $\mathbf{x}'$ by applying problem-specific repair/improvement heuristic.
    3) Ftiness Evaluation: Compute $\mathbf{F}(\mathbf{x}')$.
    4) Update z: For each $j = 1, \cdots, m$, if $z_j < f_j(\mathbf{x}')$, then set $z_j = f_j(\mathbf{x}')$.
    5) Update Solutions: For each index $j \in B_i$, if $g(\mathbf{x}'|\lambda^j, \mathbf{z}) \leq g(\mathbf{x}_j|\lambda^j, \mathbf{z})$, then set $\mathbf{x}_j = \mathbf{x}'$ and $\mathbf{F}(\mathbf{x}_j) = \mathbf{F}(\mathbf{x}')$.
    6) Update EP: Remove all solutions dominated by $\mathbf{x}'$ from EP. If no solution in EP dominates $\mathbf{x}'$, add it into EP.
Step 3 (Stopping Criterion):
If stopping criterion is satisfied, output EP. Otherwise go to Step 2.
```

A solution $\mathbf{x}^* \in X$ is Pareto optimal if there is no solution $\mathbf{x} \in X$ such that $\mathbf{x} \succ \mathbf{x}^*$. The set of all Pareto-optimal solutions is called Pareto-optimal set (PS) and the set of objective vectors corresponding to the PS is called Pareto front (PF).

In MOEA/D framework, an MOP is first decomposed into several scalar optimization subproblems according to the decomposition approach (e.g., PBI approach [6], weighted sum approach, and Tchebycheff approach [13], [41]) and weight vectors. Then, all subproblems are solved simultaneously. The original MOEA/D framework presented by Zhang and Li [6] is summarized in Algorithm 1.

B. Analyzed Algorithm

Similar to the original MOEA/D in [6], the Tchebycheff approach and the Das and Dennis's systematic [42], [43] weight vector (evenly distributed) generation method are used in MOEA/D-C. For an MOP defined in (1), the scalar optimization subproblem generated by the Tchebycheff approach is

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

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ is the weight vector, $\lambda_i \geq 0$ for $i = 1, \ldots, m$ and $\sum_{i=1}^m \lambda_i = 1$, and $\mathbf{z}^* = (z_1^*, \ldots, z_m^*)$ denotes the reference point, that is, $z_i^* = \max\{f_i(\mathbf{x})|\mathbf{x} \in X\}$. Note that when implementing a decomposition-based algorithm z_i^* is generally set to the best (maximum) value for the ith objective found so far since the maximum value for each objective in the search space X is unknown.

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 Das and Dennis's systematic approach [42], [43] generates the weight vectors by taking m values from $\{0/H, 1/H, \ldots, H/H\}$ 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 between any two subproblems is measured by the Euclidean distance between their weight vectors. It then controls a population of size N to cooperatively solve the N subproblems by using the neighborhood-based coevolution.

The analyzed MOEA/D-C is described in Algorithm 2. Different from the MOEA/D-M only using the mutation operator to create offspring, the MOEA/D-C allows to use the crossover operator in step 8. Observe that the important decomposition and neighborhood-based coevolution features of the MOEA/D framework are retained in Algorithm 2. For the specific variation operators, we consider the standard bit mutation (flipping each bit in the solution with independent probability $p_m = [1/n]$) and one-point crossover in this paper. Note that the standard bit mutation is also used as the variation operators in running time analysis of MOEA/D-M [20]. For easy of comparison, the MOEA/D-M presented in [20] is described in the supplementary material since it is similar to Algorithm 2, that is, it is equivalent to setting $p_c = 0$.

C. Analyzed Problems

Given a solution $\mathbf{x}$, we denote by $|\mathbf{x}|_1$, the number of 1-bits, and by x_i, the value of the ith bit. The five analyzed problems in this paper are defined as follows.

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

$$\text{LOTZ}(\mathbf{x}) = \left(\sum_{i=1}^{n}\prod_{j=1}^{i} x_j, \sum_{i=1}^{n}\prod_{j=i}^{n}(1 - x_j)\right).$$

This instance is defined in [22]. It is an extension of the LEADINGONES problem, and is called LOTZ. The PF of LOTZ is shown in Fig. 1. Since we view the bit-string as a vector, the PS of the problem is a high-dimensional solution set. We adopt a visualization approach called parallel coordinates [44], [45] to visualize them, which can be found in the supplementary material.

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

$$\text{COCZ}(\mathbf{x}) = (|\mathbf{x}|_1, n - |\mathbf{x}|_1).$$

This instance is an extension of the COUNTONES problem called COCZ. The above definition of COCZ is used in [20], which is lightly different from the definition in [21]. However,

```
Algorithm 2 MOEA/D-C
Input: An MOP with $m$ objectives in Eq. (1), stopping criterion, parameter $H$, the number of subproblems $N$, weight vectors $\{\lambda^1, \cdots, \lambda^N\}$ and neighbor size $T$.
Output: A Pareto optimal solution set $EP$.
1: Initialization: The Pareto-optimal solution set $EP = \emptyset$. For each subproblem $g(\mathbf{x}|\lambda^k)$, $k = 1, \cdots, N$, select the $T$ closest subproblems to form its neighbor set $B_k$ according to the Euclidean distance between their weight vectors. Generate a solution $\mathbf{x}_k \in \{0,1\}^n$ uniformly at random for each subproblem $g(\mathbf{x}|\lambda^k)$. Set the reference point $\mathbf{z}^* = (z_1^*, \cdots, z_m^*)$, where $z_i^* = \max\{f_i(\mathbf{x}_k), k = 1, \cdots, N\}$ for $i = 1, \cdots, m$. Let $S_k$ denote the set of solutions corresponding to subproblems in $B_k$.
2: while stopping criterion is not satisfied do
3:   for $k = 1, \cdots, N$ do
4:     if $rand > p_c$ then
5:       Create two new solutions $\mathbf{x}_k', \mathbf{x}_k''$ for $g(\mathbf{x}|\lambda^k)$ by using mutation operator to solution $\mathbf{x}_k$.
6:     else
7:       $C = \{\mathbf{x}_k\} \cup \{$a random solution in $S_k \setminus \mathbf{x}_k\}$.
8:       Create two new solutions $\mathbf{x}_k', \mathbf{x}_k''$ for $g(\mathbf{x}|\lambda^k)$ by using crossover operator on solutions in $C$.
9:     end if
10:    Update $\mathbf{z}^*$: For each $z_i^*$, if $\max\{f_i(\mathbf{x}_k'), f_i(\mathbf{x}_k'')\} > z_i^*$, set $z_i^* = \max\{f_i(\mathbf{x}_k'), f_i(\mathbf{x}_k'')\}$ for $i = 1, \cdots, m$.
11:    Update $S_k$: For each solution $\mathbf{x}_j$ in $S_k$, if $\min\{g(\mathbf{x}_k'|\lambda^j), g(\mathbf{x}_k''|\lambda^j)\} \le g(\mathbf{x}_j|\lambda^j)$, replace $\mathbf{x}_j$ with the better one in $\{\mathbf{x}_k', \mathbf{x}_k''\}$.
12:    Update $EP$: Remove all solutions dominated by $\mathbf{x}_k'$ or $\mathbf{x}_k''$ from $EP$. If $\mathbf{x}_k'$ or $\mathbf{x}_k''$ is not dominated by solutions in $EP$, add it into $EP$.
13:  end for
14: end while
```

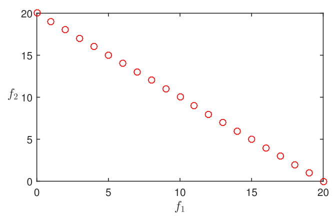


Fig. 1. Pareto fronts of LOTZ and COCZ with $n = 20$.

it only extends the size of PF from $0.5n+1$ to $n+1$ and will not change the upper bound of the expected running time. A similar definition for this instance called ONEMINMAX is presented in [25]. The PF of COCZ is shown in Fig. 1.

Definition 3 (WLPTNO): The pseudo-Boolean function WLPTNO : $\{-1,1\}^n \to \mathbb{R}^2$ is defined as follows:

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

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

This instance is defined in [40]. The abbreviation WLPTNO stands for *weighted leading positive ones trailing negative ones*. It can be considered as an extension of LOTZ 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 [20], 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. As shown in Fig. 2, unlike the LOTZ, the elements of the PF are not evenly distributed.

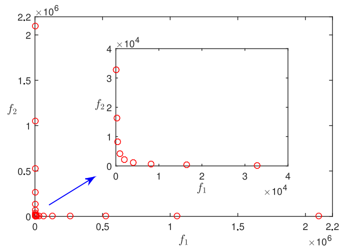


Fig. 2. Pareto front of LPTNO with $n = 20$.

Definition 4 (Dec-obj-MOP): The pseudo-Boolean function Dec-obj-MOP : $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:

$$\text{Dec-obj-MOP}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$$

where

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

This instance is defined in [20]. There exists 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 solutions are prone to be trapped into the local optimum if the fitness-based search scheme is used. 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 $\Omega(n^n)$ [20]. The PS and the PF of Dec-obj-MOP are the same as those of COCZ.

Definition 5 (Plateau-MOP): Let 1^i0^{n-i} denote a bit-string starting with i ones and ending with $n-i$ zeros. The pseudo-Boolean function Plateau-MOP : $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:

$$\text{Plateau-MOP}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$$

where

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

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

This instance is defined in [20]. For ease of expression, we hereafter assume that $(n/4)$ is an integer in this paper. 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 $\Omega(n^{n/4})$ [20].

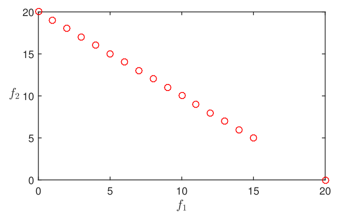


Fig. 3. Pareto front of the Plateau-MOP with $n = 20$.

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

III. Running Time Analysis

In this section, we present the running time analysis of Algorithm 2 on the above five MOPs. For parameters in Algorithm 2, in this analysis, we set $T = m + 1$ and $p_c = 0.5$. Since $m = 2$ for all problems, we have that the number of scalar optimization subproblems (i.e., the population size) $N = C_{H+m-1}^{m-1} = H + 1$. Thus, we have $N \geq 2$ because H is a positive integer. We concern on $N \geq 4$ in this paper.

Recall that $\lambda_i^k \in \{0, (1/H), (2/H), \dots, 1\}$ and $\sum_{i=1}^{m} \lambda_i^k = 1$ for all $k = 1, 2, \dots, N$. For convenience, we hereafter number the elements of the set of weight vectors with ascending order according to the value of λ_1^k, that is, $\lambda^1 = (0, 1)$, $\lambda^2 = ([1/H], 1 - [1/H]), \dots, \lambda^{N-1} = ([H-1]/H, [1/H])$, $\lambda^N = (1, 0)$.

For each problem, the analysis mainly consists of two phases. In the first phase, we neglect the promotion of the crossover operator and neighborhood-based coevolution, and prove the expected upper bound of the mutation operator in Algorithm 2 obtaining a Pareto-optimal solution for every subproblem. In the second phase, we prove that by using crossover operator on these solutions obtained in the first phase for the N subproblems, Algorithm 2 obtains a set of solutions to cover the PF quickly ($O(N \cdot n \log n)$). This analyzed process is inspired by the proof in [40], where an MOEA with crossover is analyzed, which is initialized with the optimum of each single objective. The main difference is that our analyzed algorithm is based on the MOEA/D framework, and individuals are randomly initialized and the population size is an input parameter of the algorithm.

A. Analysis on LOTZ

The PF of LOTZ is $\{(0, n), (1, n-1), \dots, (n, 0)\}$ and the PS is $\{0^n, 1^10^{n-1}, \dots, 1^n\}$. The best known upper bound of the expected running time on LOTZ is $O(n^2)$ [40]. Li *et al.* [20] showed that the expected running time of MOEA/D-M on LOTZ is $O(n^3)$. In the following lemma, we prove that the expected running time of Algorithm 2 on LOTZ is $O(N \cdot n^2)$.

Lemma 1: For LOTZ, the Pareto-optimal solutions 0^n and 1^n are found by Algorithm 2 for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n^2)$.

To prove this lemma, we first show that the optimizations of subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ are equivalent to maximizing the numbers of trailing 0-bits and leading 1-bits in the solution, respectively. Then, we estimate the upper bound of the expected running time of Algorithm 2 finding the optimal solutions 0^n and 1^n for them.

Proof: The subproblems corresponding to weight vectors $\lambda^1 = (0, 1)$ and $\lambda^N = (1, 0)$ of LOTZ in Algorithm 2 are

$$\min g\left(\mathbf{x}|\lambda^1\right) = \left|\sum_{i=1}^{n}\prod_{j=i}^{n}(1 - x_j) - z_2^*\right| \tag{3}$$

$$\min g\left(\mathbf{x}|\lambda^N\right) = \left|\sum_{i=1}^{n}\prod_{j=1}^{i}x_j - z_1^*\right|. \tag{4}$$

Note that when Algorithm 1 handles an MOP, the best values of z_1^* and z_2^* are often unknown previously and are dynamically updated during the evolutionary process. Since the value of z_2^* is set to the best (maximum) value for the second objective so far (line 10 in Algorithm 2), for all solutions $\mathbf{x}$ have been created for $g(\mathbf{x}|\lambda^1)$, (3) can be rewritten as

$$\min g\left(\mathbf{x}|\lambda^1\right) = z_2^* - \sum_{i=1}^{n}\prod_{j=i}^{n}\left(1 - x_j\right) = z_2^* - f_2(\mathbf{x}).$$

For the two new solutions $\mathbf{x}_1'$ and $\mathbf{x}_1''$ created for (3), if $\max\{f_2(\mathbf{x}_1'), f_2(\mathbf{x}_1'')\} \geq f_2(\mathbf{x}_1)$, according to the update principle in line 11 of Algorithm 2, the better one of them will be accepted to replace $\mathbf{x}_1$ in that iteration. Thus, the optimization of (3) is equivalent to maximize the number of trailing 0-bits in the solution.

If the optimal solution 0^n has not been found, in an iteration, the mutation operator in Algorithm 2 creates a better solution from the current one with probability at least $(1/n) \cdot (1 - [1/n])^{n-1} \geq (1/en)$, since it happens if the rightmost 1-bit is flipped while keeping other bits unchanged. Recall that $p_c = 0.5$ and two new solutions are created for each subproblem in an iteration. Note that other solutions that are created during this phase can only accelerate the search process. Therefore, according to the fitness-level approach [46], [47], the expected fitness evaluations of Algorithm 2 to find the optimal solution 0^n for subproblem $g(\mathbf{x}|\lambda^1)$ (3) are upper bounded by

$$2\sum_{i=k}^{n-1} en \leq 2en\sum_{i=0}^{n-1} 1 = 2en^2 = O\left(n^2\right) \tag{5}$$

where k is the number of trailing 0-bits in the initial solution.

Similarly, we have that the optimization of (4) is equivalent to maximize the number of leading 1-bits in the solution. So Algorithm 2 creates a better solution from the current one with probability at least $(1/n) \cdot (1 - [1/n])^{n-1} \geq (1/en)$ if the optimal solution 1^n has not been found. Thus, the upper bound of the expected fitness evaluations of Algorithm 2 to find the optimal solution 1^n for (4) are $O(n^2)$.

Note that Algorithm 2 optimizes the N subproblems in parallel since it creates and evaluates two new solutions for each subproblem in each generation. Therefore, Algorithm 2 finds the Pareto-optimal solutions 0^n and 1^n for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n^2)$. ■

We next prove that Algorithm 2 obtains a Pareto-optimal solution for all subproblems in expected running time $O(N \cdot n^2)$ after 0^n and 1^n have been found for $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$, respectively.

Lemma 2: For LOTZ, Algorithm 2 obtains a Pareto-optimal solution for every subproblem $g(\mathbf{x}|\lambda^i), i = 2, \ldots, N-1$, in expected running time $O(N \cdot n^2)$.

The sketch of proof for this lemma is as follows. We first determinate the form of the optimal solution for any $g(\mathbf{x}|\lambda^i)$. Then the expected running time of Algorithm 2 finding the optimal solution from an arbitrary solution is estimated.

Proof: After Algorithm 2 has found the optimal solutions 0^n and 1^n for (3) and (4), the values of z_1^* and z_2^* are set to n. Thus, these subproblems corresponding to weight vectors $\lambda^i, i = 2, \ldots, N-1$, are

$$\begin{aligned}\min g\big(\mathbf{x}|\lambda^i\big) &= \max\big\{\lambda_1^i|f_1(\mathbf{x}) - n|, \lambda_2^i|f_2(\mathbf{x}) - n|\big\}\\ &= \max\big\{\lambda_1^i(n - f_1(\mathbf{x})), \lambda_2^i(n - f_2(\mathbf{x}))\big\} \quad (6)\end{aligned}$$

where $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are the number of leading 1-bits and trailing 0-bits in $\mathbf{x}$, respectively. So we have $f_1(\mathbf{x}) + f_2(\mathbf{x}) \le n$.

We argue that for any $g(\mathbf{x}|\lambda^i)$ in (6), the optimal solution $\mathbf{x}$ must satisfy that $f_1(\mathbf{x}) + f_2(\mathbf{x}) = n$. Observe that for a given $g(\mathbf{x}|\lambda^i)$, the value of $\lambda_1^i(n - f_1(\mathbf{x}))$ decreases when increasing the value of $f_1(\mathbf{x})$, and the value of $\lambda_2^i(n - f_2(\mathbf{x}))$ decreases when increasing the value of $f_2(\mathbf{x})$. To minimize the maximum value of them, we should simultaneously increase the values of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ as much as possible. First, we claim that $f_1(\mathbf{x}) + f_2(\mathbf{x}) \ge n - 1$ holds for any optimal solution $\mathbf{x}$ of $g(\mathbf{x}|\lambda^i)$. Otherwise, if $f_1(\mathbf{x}) + f_2(\mathbf{x}) \le n - 2$, we can increase both $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ by one, which will simultaneously decrease the values of the two items in the right hand of $g(\mathbf{x}|\lambda^i)$. Thus, a smaller value is found for $g(\mathbf{x}|\lambda^i)$. This contradicts the assumption that $\mathbf{x}$ is an optimal solution. Second, we claim that $f_1(\mathbf{x}) + f_2(\mathbf{x}) \ne n - 1$ for any solution $\mathbf{x}$. If $f_1(\mathbf{x}) + f_2(\mathbf{x}) = n - 1$, there is exactly one bit between them. Since this bit is either 1 or 0, it will increase a 1-bit to the leading 1-bits or a 0-bit to the trailing 0-bits. Thus, in this case, it must be that $f_1(\mathbf{x}) + f_2(\mathbf{x}) = n$ for the solution $\mathbf{x}$.

We now determine the exact values of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ in the optimal solution for a given $g(\mathbf{x}|\lambda^i)$. Since $f_1(\mathbf{x}) + f_2(\mathbf{x}) = n$ holds for any optimal solution $\mathbf{x}$, by (6), we have

$$\min g\big(\mathbf{x}|\lambda^i\big) = \max\big\{\lambda_1^i(n - f_1(\mathbf{x})), \lambda_2^i f_1(\mathbf{x})\big\} \quad (7)$$

Recall that $f_1(\mathbf{x})$ is the number of leading 1-bits in $\mathbf{x}$. Observe the trend of two items in (7) when $f_1(\mathbf{x})$ increases from 0 to n. It is easy to see that the value of the first item is decreasing from $\lambda_1^i n > 0$ to 0 and the second item is increasing from 0 to $\lambda_2^i n > 0$. Thus, there must exist a real number $y \in [0, n]$ such that

$$\lambda_1^i(n - y) - \lambda_2^i y = 0 \Rightarrow y = \lambda_1^i n. \quad (8)$$

To get the minimum value for $g(\mathbf{x}|\lambda^i)$, we should set $f_1(\mathbf{x})$ close to y as much as possible. Note that there may have two consecutive natural numbers for $f_1(\mathbf{x})$ to obtain the minimum value since $f_1(\mathbf{x})$ is a natural number while y may be a real.

For ease of expression, we assume that y is a natural number. This means that the optimal solution of $g(\mathbf{x}|\lambda^i)$ is 1^y0^{n-y}. In the following text, we prove that starting from an arbitrary solution, Algorithm 2 finds the optimal solution 1^y0^{n-y} for $g(\mathbf{x}|\lambda^i)$ in expected running time $O(n^2)$. Note that this result is also true for the case that there are two consecutive natural numbers for $f_1(\mathbf{x})$ to obtain the minimum value of $g(\mathbf{x}|\lambda^i)$ since any one of them is an optimal solution.

For convenience, we use the shorthands $f_{1,\mathbf{x}}$ and $f_{2,\mathbf{x}}$ for $\lambda_1^i(n - f_1(\mathbf{x}))$ and $\lambda_2^i(n - f_2(\mathbf{x}))$, respectively. So $f_{1,\mathbf{x}}$ decreases when increasing the number of leading 1-bits and $f_{2,\mathbf{x}}$ decreases when increasing the number of trailing 0-bits. Let f_0 denote the optimal function value of $g(\mathbf{x}|\lambda^i)$, that is, $f_0 = \lambda_1^i(n - y) = \lambda_2^i y$. We assume that at the beginning $f_{1,\mathbf{x}}$ and $f_{2,\mathbf{x}}$ are not equal to f_0. Otherwise, the optimal solution has been found for $g(\mathbf{x}|\lambda^i)$ and there has nothing to be proved.

If $f_{1,\mathbf{x}} \ne f_{2,\mathbf{x}}$, without loss of generality, we assume that $f_{1,\mathbf{x}} < f_{2,\mathbf{x}}$. In the generation, if the mutation operator flips the rightmost 1-bit in $\mathbf{x}$ and keeps the values of all other bits unchanged, an improved solution is created for $g(\mathbf{x}|\lambda^i)$ since it decreases $f_{2,\mathbf{x}}$ (denoted by $f_{2,\mathbf{x}}^-$, the new value) while keeping $f_{1,\mathbf{x}}$ unchanged. This event happens with probability $(1/n)\cdot(1-[1/n])^{n-1} \ge (1/en)$. Note that before obtaining such an improvement, the algorithm may accept some "degenerate" solutions, which keep $f_{2,\mathbf{x}}$ unchanged and limitedly increase $f_{1,\mathbf{x}}$ (denoted by $f_{1,\mathbf{x}}^+$, the new value). In worst case, $f_{1,\mathbf{x}}^+$ can equal to $f_{2,\mathbf{x}}$. The probability of such an event is at most $(1/n)$ since the standard bit mutation needs to flip at least a 1-bits in the solution. If $f_{1,\mathbf{x}} = f_{2,\mathbf{x}} > f_0$, in the generation it turns into the case $f_{1,\mathbf{x}} \ne f_{2,\mathbf{x}}$ with probability at least $C_2^1 \cdot (1/n) \cdot (1 - [1/n])^{n-1} \ge (2/en)$, since it happens if the mutation operator flips the leftmost 0-bit or rightmost 1-bit in $\mathbf{x}$ and keeps all other bits unchanged.

To help understand the aforementioned analysis process, we give a concrete example as follows. Let $n = 18$ and $\lambda^i = (1/3, 2/3)$. From (8), we have $y = \lambda_1^i \cdot n = 6$ and thus $f_0 = \lambda_1^i \cdot (n - y) = 4$. So the optimal solution for this subproblem is 1^60^{12}. Assume that the current solution is $\mathbf{x} = 1^30 * 10^9$, we have $f_{1,\mathbf{x}} = 5$ and $f_{2,\mathbf{x}} = 6$, that is, $f_{1,\mathbf{x}} < f_{2,\mathbf{x}}$. If the mutation operator flips the rightmost 1-bit in $\mathbf{x}$ and keeps all other bits unchanged, the offspring (new solution) is $\mathbf{x}' = 1^30 * 0^{10}$. This event happens with probability $(1/n) \cdot (1 - [1/n])^{n-1} \ge (1/en)$. For the subproblem, $\mathbf{x}'$ is better than $\mathbf{x}$ since $f_{1,\mathbf{x}'} = 5$ and $f_{2,\mathbf{x}'} = (16/3)$. If the mutation operator flips the first 1-bit in $\mathbf{x}$ and keeps all trailing 0-bits unchanged, the new solution is $\mathbf{x}'' = 0110 * 10^9$. This event happens with probability $(1/n)\cdot(1-[1/n])^9 \le (1/n)$. For the subproblem, $\mathbf{x}''$ has the same fitness with $\mathbf{x}$, since $f_{1,\mathbf{x}''} = 6$ and $f_{2,\mathbf{x}''} = 6$. The event that turns $f_{1,\mathbf{x}} = f_{2,\mathbf{x}} > f_0$ into $f_{1,\mathbf{x}} \ne f_{2,\mathbf{x}}$ is equivalent to creating $\mathbf{x} = 1^30 * 10^9$ or $0110 * 0^{10}$ from $\mathbf{x}''$. The probability is $C_2^1 \cdot (1/n) \cdot (1 - [1/n])^{n-1} \ge (2/en)$ since it happens if the mutation operator flips the leftmost 0-bit or rightmost 1-bit in $\mathbf{x}''$ and keeps all other bits unchanged.

For the above discussion, in the worst case, the process of the mutation operator in Algorithm 2 finding a better solution for $g(\mathbf{x}|\lambda^i)$ from the current one is equivalent to reaching

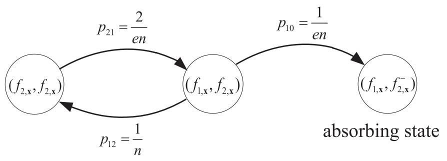

$p_{21} = \frac{2}{en}$
$p_{10} = \frac{1}{en}$
$(f_{2,\mathbf{x}}, f_{2,\mathbf{x}})$
$(f_{1,\mathbf{x}}, f_{2,\mathbf{x}})$
$(f_{1,\mathbf{x}}, f_{2,\mathbf{x}}^{-})$
absorbing state
$p_{12} = \frac{1}{n}$

Fig. 4. Markov chain of finding an improved solution for $g(\mathbf{x}|\lambda^i)$.

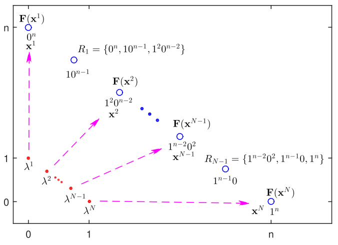

$\mathbf{F}(\mathbf{x}^1)$
0^n
$\mathbf{x}^1$
$R_1 = \{0^n, 10^{n-1}, 1^20^{n-2}\}$
10^{n-1}
$\mathbf{F}(\mathbf{x}^2)$
1^20^{n-2}
$\mathbf{x}^2$
$\mathbf{F}(\mathbf{x}^{N-1})$
$1^{n-2}0^2$
$\mathbf{x}^{N-1}$
$R_{N-1} = \{1^{n-2}0^2, 1^{n-1}0, 1^n\}$
$1^{n-1}0$
$\mathbf{F}(\mathbf{x}^N)$
$\mathbf{x}^N$
1^n
λ^1
λ^2
λ^{N-1}
λ^N
n
1
0
0
1
n

Fig. 5. Partition the PS into $N-1$ subsets according to the position of $\mathbf{x}^i$.

						d_1				d_2										
$n = 20$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\mathbf{X}^j = 1^50^{15}$	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{X}^{j+1} = 1^{10}0^{10}$	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

Fig. 6. Example of $\mathbf{x}^j = 1^50^{15}$, $\mathbf{x}^{j+1} = 1^{10}0^{10}$, $k = 4$, and $R_j = \{1^50^{15}, 1^60^{14}, 1^70^{13}, 1^80^{12}, 1^90^{11}, 1^{10}0^{10}\}$. If the crossover position is in $[d_1, d_2)$, a solution between $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ in R_j is created.

the absorbing state of the Markov chain shown in Fig. 4. From [48, Corollary 2], the expected running time to reach the absorbing state is at most $(1/p_{21}) + (1/p_{10}) + (1/p_{21}) \cdot (p_{12}/p_{10}) \leq 7.78n$. Therefore, the expected fitness evaluations of Algorithm 2 finding the optimal solution for $g(\mathbf{x}|\lambda^i)$ are upper bounded by

$$2\sum_{j=0}^{n/2-1} 7.78n = 7.78n^2 = O\left(n^2\right) \tag{9}$$

since the number of solutions that $f_{1,\mathbf{x}} = f_{2,\mathbf{x}}$ is at most $n/2$.

Note that any optimal solution for $g(\mathbf{x}|\lambda^i)$, $i = 2, \ldots, N-1$, is Pareto optimal for LOTZ. Therefore, for LOTZ, Algorithm 2 obtains a Pareto-optimal solution for subproblems $g(\mathbf{x}|\lambda^i)$, $i = 2, \ldots, N-1$, in expected running time $O(N \cdot n^2)$. ■

We next prove that Algorithm 2 obtains a set of solutions to cover the PF of LOTZ in expected running time $O(N \cdot n^2)$.

Theorem 1: For LOTZ, Algorithm 2 obtains a set of solutions to cover the PF in expected running time $O(N \cdot n^2)$.

To prove this theorem, we first show that the entire PS of the problem can be partitioned into multiple subsets with the extreme points of the Pareto-optimal solutions obtained by these subproblems. Then, we show that all solutions in any subset can be created by using the one-point crossover to its two extreme points in expected running time $O(n \log n)$.

Proof: By Lemmas 1 and 2, we know that Algorithm 2 finds N Pareto-optimal solutions for LOTZ in expected running time $O(N \cdot n^2)$. Note that these N objective vectors may have not covered the entire PF since N is a variable here, e.g., for any value of N such that $N < |PF|$. Moreover, there may be some solutions corresponding to the same point on the PF. We next assume that these N objective vectors have not covered the PF; otherwise, there has nothing to be proved.

We denote by $\mathbf{x}^i$, the solution found by Algorithm 2 for subproblem $g(\mathbf{x}|\lambda^i)$, $i = 1, 2, \ldots, N$ after $c \cdot N \cdot n^2$ fitness evaluations, where c is a large enough constant. Recall that, for LOTZ, the PF is $\{(0, n), (1, n-1), \ldots, (n, 0)\}$, and the PS is $\{0^n, 1^10^{n-1}, \ldots, 1^n\}$. Note that we have $\mathbf{x}^1 = 0^n$ and $\mathbf{x}^N = 1^n$ by Lemma 1. As shown in Fig. 5, we partition the PS into $N-1$ subsets according to the position of $\mathbf{x}^i$, $i = 1, 2, \ldots, N$, in it, that is, $R_j = \{\mathbf{x}^j, \ldots, \mathbf{x}^{j+1}\}$ for $1 \leq j \leq N-1$. It is not difficult to see that $\cup_{j=1}^{N-1} R_j = \{0^n, 1^10^{n-1}, \ldots, 1^n\}$. Thus, to find a set of solutions to cover the PF is equivalent to find a set of solutions to cover each R_j.

Recall that Algorithm 2 constructs the neighbor set B_i for each subproblem $g(\mathbf{x}|\lambda^i)$, $1 \leq i \leq N$, according to the Euclidean distance between their weight vectors. So we have $S_1 = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}$, $S_N = \{\mathbf{x}^{N-2}, \mathbf{x}^{N-1}, \mathbf{x}^N\}$, and $S_j = \{\mathbf{x}^{j-1}, \mathbf{x}^j, \mathbf{x}^{j+1}\}$ for $2 \leq j \leq N-1$. Observe that solutions in R_j can be created by using the crossover on S_j and S_{j+1}, because they contain the two endpoints of R_j, that is, $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$. For a given R_j, let k denote the number of solutions between $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ in the PS. If $k = 0$, then R_j have been covered. Otherwise, the Hamming distance between $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ is $k + 1 \in [2, n]$. Let d_1 and d_2 denote the minimum and the maximum indices, where the values between $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ are different. We have $d_2 - d_1 = k \leq n$. If and only if one of positions in $[d_1, d_2)$ is selected, the crossover creates a unique solution in R_j. A concrete example for this analysis is shown in Fig. 6. Thus, to obtain a set of solutions to cover R_j, the crossover operator needs to select each position in $[d_1, d_2)$ at least once. Let i denote the number of these positions that have been selected in the previous generation, one of the rest $k - i$ positions is selected by the crossover operator in the current generation with probability $2 \cdot (1/4) \cdot ([k-i]/n)$. Recall that $p_c = 0.5$ and in an iteration solutions $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ are selected from S_j or S_{j+1} with probability 0.5. Thus, the expected fitness evaluations of Algorithm 2 obtaining a set of solutions to cover R_j are upper bounded by

$$2\sum_{i=0}^{k-1} \frac{2n}{k-i} \leq 4n \sum_{j=1}^{n-1} \frac{1}{j} = 4n \log n = O(n \log n) \tag{10}$$

where the inequality follows from that $k \leq n - 1$. Thus, we have that Algorithm 2 finds a set of solutions to cover all R_j for $1 \leq j \leq N-1$ in expected running time $O(N \cdot n \log n)$.

Therefore, Algorithm 2 obtains a set of solutions S to cover the PF, that is, $PF \subseteq \mathbf{F}(S)$ of LOTZ in expected running time $O(N \cdot n^2) + O(N \cdot n \log n) = O(N \cdot n^2)$. ■

B. Analysis on COCZ

For COCZ, the PF is $\{(0, n), (1, n-1), \ldots, (n, 0)\}$, and a set of corresponding solutions is $\{0^n, 0^{*n-1}1^{*1}, \ldots, 1^n\}$, where $0^{*i}1^{*j}$ indicates that there are exactly i 0-bits and j 1-bits in the solution. The best known upper bound of the expected running time on COCZ is $O(n \log n)$ [40]. Li *et al.* [20] proved that the evenly distributed weight vectors, that is, $\lambda^k = (\lambda_1^k, 1 - \lambda_1^k)$, $\lambda_1^k = (k/n)$, $k = 0, \ldots, n$, produce $n + 1$ subproblems that are mapped one-to-one to the points on the PF. So the MOEA/D-M with parameter $H = n$ 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 $O(n \log n)$. In the following lemma, we prove that MOEA/D-C obtains a set of solutions to cover the PF of COCZ in expected running time $O(N \cdot n \log n)$.

Lemma 3: For COCZ, Algorithm 2 finds the optimal solutions 0^n and 1^n for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n \log n)$.

Proof: By similar argument in the beginning of the proof of Lemma 1, we have that, for COCZ, the subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ are as follows:

$$\min g\left(\mathbf{x}|\lambda^1\right) = z_2^* - (n - |\mathbf{x}|_1) \tag{11}$$

$$\min g\left(\mathbf{x}|\lambda^N\right) = z_1^* - |\mathbf{x}|_1. \tag{12}$$

Thus, the optimizations of $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ are equivalent to maximize the numbers of 0-bits and 1-bits in the solution, respectively.

For $g(\mathbf{x}|\lambda^1)$, let i denote the number of 0-bits in the current solution. So there are $(n-i)$ 1-bits in it. In the generation, if the mutation operator flips one of the $(n-i)$ 1-bits in the solution and keeps all other bits unchanged, a better solution is created for $g(\mathbf{x}|\lambda^1)$. Thus, the mutation operator creates a better solution from the current one with probability $C_{n-i}^1 \cdot (1/n) \cdot (1-[1/n])^{n-1} \geq ([n-i]/en)$. Therefore, the expected fitness evaluations of Algorithm 2 finding the optimal solution 0^n for $g(\mathbf{x}|\lambda^1)$ (11) are upper bounded by

$$2\sum_{i=0}^{n-1} \frac{en}{n-i} = 2en \sum_{j=1}^{n} \frac{1}{j} = 2en \log n = O(n \log n). \tag{13}$$

Similarly, we have that the upper bound of the expected fitness evaluations of Algorithm 2 to find the optimal solution 1^n for $g(\mathbf{x}|\lambda^N)$ (12) is $O(n \log n)$.

Therefore, Algorithm 2 finds the optimal solutions 0^n and 1^n for $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n \log n)$ since it optimizes the N subproblems in parallel. ■

Lemma 3 implies that Algorithm 2 finds and keeps the Pareto-optimal solutions 0^n and 1^n for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected fitness evaluations $O(N \cdot n \log n)$. Note that, for COCZ, all solutions are Pareto optimal. So these solutions kept by Algorithm 2 for other $N-2$ subproblems are also Pareto optimal. We next assume that these objective vectors of the N Pareto-optimal solutions have not covered the PF, and show that Algorithm 2 obtains a set of solutions to cover the rest elements in expected running time $O(N \cdot n \log n)$.

Theorem 2: For COCZ, Algorithm 2 obtains a set of solutions to cover the PF in expected running time $O(N \cdot n \log n)$.

Proof: The main idea of proof for this theorem is similar to the one used in Theorem 1. However, there are some differences in detail. Different from LOTZ, all 2^n solutions of COCZ are Pareto optimal. Thus, we only need to bound the expected running time to obtain a solution for each point in the PF.

We list all solutions of COCZ according to the ascending order of the number of 1-bits, that is, $S_{\text{cocz}} = \{0^n, 0^{*n-1}1^{*1}, \ldots, 0^{*1}1^{*n-1}, 1^n\}$, where $0^{*i}1^{*j}$ denotes a group

$\mathbf{X}^j = 1^{*3}0^{*17}$	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
$\mathbf{X}^{j+1} = 1^{*8}0^{*12}$	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	0	0	1	0
$h = 1,2,\cdots,20$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\lvert\mathbf{X}^{j+1}\rvert_{1\leq h} - \lvert\mathbf{X}^j\rvert_{1\leq h}$	0	1	1	2	1	2	2	3	2	3	3	4	3	3	4	4	4	4	5	5

Fig. 7. Example of $\mathbf{x}^j = 1^{*3}0^{*17}$, $\mathbf{x}^{j+1} = 1^{*8}0^{*12}$, $k = 4$, and $R_j = \{1^{*3}0^{*17}, 1^{*4}0^{*16}, 1^{*5}0^{*15}, 1^{*6}0^{*14}, 1^{*7}0^{*13}, 1^{*8}0^{*12}\}$. The values of $|\mathbf{x}^{j+1}|_{1<h} - |\mathbf{x}^j|_{1<h}$ are shown in the last row, where $|\mathbf{x}^j|_{1<h}$ denotes the number of 1-bits in $\mathbf{x}^j$ from position 1 to h. If the crossover position is $h = 2, 3$, or 5 ($l = 1$), solutions $1^{*4}0^{*16}$ and $1^{*7}0^{*13}$ are created.

of solutions that contain exactly i 0-bits and j 1-bits in them. Thus, we can also partition S_{cocz} into $N-1$ subsets according to the position of $\mathbf{x}^i$ in it such that $\cup_{j=1}^{N-1} R_j = S_{\text{cocz}}$, where $R_j = \{\mathbf{x}^j, \ldots, \mathbf{x}^{j+1}\}$ for $j = 1, \ldots, N-1$. This means that R_j includes all solutions listed in S_{cocz} between $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$, namely, the number of 1-bits in them is in $[|\mathbf{x}^j|_1, |\mathbf{x}^{j+1}|_1]$. Note that, by Lemma 3, we have $\mathbf{x}^1 = 0^n$ and $\mathbf{x}^N = 1^n$.

For a given R_j, let k be the number of group solutions in it except $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$. If $k = 0$, then R_j have been covered. Otherwise, the difference of the numbers of 1-bits in $\mathbf{x}^j$ and $\mathbf{x}^{j+1}$ is $k+1 \geq 2$. Without loss of generality, we assume that $|\mathbf{x}^{j+1}|_1 > |\mathbf{x}^j|_1$. Let $|\mathbf{x}^j|_{1\leq h}$ denote the number of 1-bits in $\mathbf{x}^j$ from position 1 to h. For each $l \in [1, k]$, there must exist at least a position $h \in [1, n]$ such that $|\mathbf{x}^{j+1}|_{1\leq h} - |\mathbf{x}^j|_{1\leq h} = l$. If one of these positions is selected, the one-point crossover creates two specific solutions in R_j. Fig. 7 shows a concrete example for this analysis. Thus, if the crossover operator selects these positions at least once, it must create a set of solutions to cover R_j. Let t denote the number of these positions that have been selected in the previous iterations, a new position is selected by the crossover operator in a new iteration with probability $2 \cdot (1/4) \cdot ([k-t]/n)$. Thus, the expected fitness evaluation of Algorithm 2 obtaining a set of solutions to cover R_j is upper bounded by

$$2\sum_{t=0}^{k-1} \frac{2n}{k-t} \leq 4n \sum_{j=1}^{n-1} \frac{1}{j} = 4n \log n = O(n \log n). \tag{14}$$

Therefore, Algorithm 2 obtains a set of solutions to cover the PF of COCZ in expected running time $O(N \cdot n \log n) + O(N \cdot n \log n) = O(N \cdot n \log n)$. ■

C. Analysis on LPTNO

For LPTNO, the PF is $\{(0, 2^{n+1}-2), (2^2-2, 2^n-2), \ldots, (2^{n+1}-2, 0)\}$, and the set of Pareto-optimal solutions corresponding to the PF is $\{(-1)^n, 1^1(-1)^{n-1}, \ldots, 1^n\}$.The best known upper bound for LPTNO is $O(n^2)$ [40]. Although LPTNO is a simple modification of LOTZ, there are dramatic difference between the distributions of their PFs. Li *et al.* [20] proved that the weight vectors, $\lambda^k = (\lambda_1^k, 1-\lambda_1^k), \lambda_1^k = ([\sum_{j=1}^k 2^{n+1-j}/[\sum_{j=k+1}^n 2^j + \sum_{j=1}^k 2^{n+1-j}]), k = 0, \ldots, n$, produce a set of subproblems that are mapped one-to-one to the points on the PF. So the MOEA/D-M with parameter $H = n$ 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 LPTNO is $O(n^3)$ since it solves each subproblem in expected running time $O(n^2)$. However, we notice that the distribution of this set of weight vectors is ill-condition since most of λ^k fall nearby $(0.5, 0.5)$, and it is hard to find in practical application. This implies that MOEA/D-M with even weight vector generation method can probably find some points on the PF even when setting H to a large number. The following theorem shows that the upper bound of the expected running time of MOEA/D-C with even weight vector generation method for this problem is $O(N \cdot n^2)$.

Theorem 3: For LPTNO, Algorithm 2 obtains a set of solutions to cover the PF in expected running time $O(N \cdot n^2)$.

The proof of this theorem is similar to prove the expected running time $O(N \cdot n^2)$ on LOTZ problem. Due to the limitation of pages, we only give the sketch of the proof and a detail can be found in the supplementary material. First, the mutation operator in Algorithm 2 creates an optimal solution for every subproblem in expected running time $O(N \cdot n^2)$. Note that, for LPTNO, the optimal solutions of some subproblems are mapped to the same point on the PF. Second, if some points on the PF have not been found after $cN \cdot n^2$ fitness evaluations, the crossover operator in Algorithm 2 will create a set of solutions to cover them in expected running time $O(N \cdot n \log n)$. Thus, in total, Algorithm 2 obtains a set of solutions to cover the PF of LPTNO in expected running time $O(N \cdot n^2) + O(N \cdot n \log n) = O(N \cdot n^2)$.

D. Analysis on Dec-obj-MOP

For Dec-obj-MOP problem, the PF is $\{(0, n), (1, n - 1), \ldots, (n, 0)\}$ and a set of corresponding solutions is $\{0^n, 1^n, 1^{*n-1}0^{*1}, \ldots, 1^{*1}0^{*n-1}\}$. The best known upper bound of expected running time is $O(n^2 \log n)$ [20]. The evenly distributed weight vectors, that is, $\lambda^k = (\lambda_1^k, 1 - \lambda_1^k), \lambda_1^k = (k/n), k = 0, \ldots, n$, produces a set of subproblems that are mapped one-to-one to the points in the PF. However, different from COCZ, that all 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 $(0, 1)$, that is, $g(\mathbf{x}|\lambda^1)$ is $\Omega(n^n)$ because of the deceive property in the search space. In the following theorem, we show that the expected running time of Algorithm 2 on this problem is $O(N \cdot n \log n)$.

Theorem 4: For Dec-obj-MOP, Algorithm 2 obtains a set of solutions to cover the PF in expected running time $O(N \cdot n \log n)$.

The proof of this result is similar to prove the expected running time $O(N \cdot n \log n)$ on COCZ problem. However, there are slight differences. First, the mutation operator in Algorithm 2 finds the Pareto-optimal solutions 1^n and $1^{*1}0^{*n-1}$ for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n \log n)$. Note that, for subproblem $g(\mathbf{x}|\lambda^1)$, the mutation operation in Algorithm 2 creates the optimum 0^n in $\Omega(n^n)$ since it needs to flip the n 1-bits in the local optimum 1^n simultaneously. Second, as argued in the proof of Theorem 2, we have that after $cN \cdot n \log n$ fitness evaluations, the crossover operator in Algorithm 2 creates a set of solutions to cover the PF of Dec-obj-MOP problem in expected running time $O(N \cdot n \log n)$ except for the point $(0, n)$ which corresponds to the solution 0^n. Third, after the solution $1^{*1}0^{*n-1}$ has been found for subproblem $g(\mathbf{x}|\lambda^N)$, it will be kept forever since it is the optimal solution for the subproblem. Thus, it will be selected to be a parent for reproduction in later mutation step, and the offspring is 0^n with probability at least $(1/en)$. Hence, the Pareto-optimal solution 0^n is created and added into EP in expected fitness evaluations $O(n)$ after the solution $1^{*1}0^{*n-1}$ has been found for $g(\mathbf{x}|\lambda^N)$. Therefore, in total, Algorithm 2 obtains a set of solutions to cover the PF in expected running time $O(N \cdot n \log n) + O(N \cdot n \log n) + O(n) = O(N \cdot n \log n)$. A complete proof can be found in the supplementary material.

E. Analysis on Plateau-MOP

The PF of Plateau-MOP is $\{(0, n), (1, n-1), \ldots, (3n/4, n/4), (n, 0)\}$ and the set of Pareto-optimal solutions corresponding to PF is $\{0^n, 10^{n-1}, \ldots, 1^{3n/4}0^{n/4}, 1^n\}$. Notice that the point $(n, 0)$ is an outlier in the objective space. In the following analysis, we neglect the effect of this point since one point often affects very limited in computing the performance metrics (e.g., the popular IGD [49], [50]) in computational experiments for MOPs. Li *et al.* [20] proved that the evenly distributed weight vectors, that is, $\lambda^k = (\lambda_1^k, 1 - \lambda_1^k), \lambda_1^k = (k/n), k = 0, \ldots, n$, produce a set of subproblems that are mapped one-to-one to the points in the PF. So the MOEA/D-M with $H = n$ can obtain a set of solutions to cover the PF (expect the outlier point) by solving the decomposed $(3n/4) + 1$ subproblems. Thus, the expected running time of the MOEA/D-M on Plateau-MOP is $O(n^3)$ since it solves each subproblem in expected running time $O(n^2)$ [20]. In the following theorem, we show that Algorithm 2 obtains a set of solutions to cover the PF of Plateau-MOP (except the outlier point) in expected running time $O(N \cdot n^2)$.

Theorem 5: For Plateau-MOP, Algorithm 2 obtains a set of solutions to cover the PF (except the outlier point) in expected running time $O(N \cdot n^2)$.

The proof of this result is similar to prove the expected running time $O(N \cdot n^2)$ on LOTZ problem. The main difference is that in the first phase, the mutation operator in Algorithm 2 first creates a solution in the form of $1^i0^{n-i}, 0 \le i < n$, (Pareto-optimal solution) for each subproblem in expected running time $O(N \cdot n \log n)$. Then, as argued in the proof of Lemma 1, it creates the solutions 0^n and $1^i0^{n-i}, i \ge (3n/4)$, for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$ in expected running time $O(N \cdot n^2)$. If some points in $\{(0, n), (1, n-1), \ldots, (3n/4, n/4)\}$ have not been found after $cN \cdot n^2$ fitness evaluations, as argued in the proof of Theorem 1, in the second phase, the crossover operator in Algorithm 2 will creates a set of solutions to them in expected running time $O(N \cdot n \log n)$. Thus, in total, Algorithm 2 obtains a set of solutions to cover the PF of Plateau-MOP (except the outlier point) in expected running time $O(N \cdot n \log n) + O(N \cdot n^2) + O(N \cdot n \log n) = O(N \cdot n^2)$. A detailed proof of this theorem can be found in the supplementary material.

Note that the value of N is a variable in the above analysis, from Theorems 1–5, we have the following theorem.

Theorem 6: By setting $N = O(1)$, Algorithm 2 obtains a set of solutions to cover the PF of LOTZ, WLPTN, and

Plateau-MOP in expected running time $O(n^2)$, and the PF of COCZ and Dec-obj-MOP in expected running time $O(n \log n)$.

Theorem 6 implies that if we set the number of the decomposed subproblems (also population size) small enough, the upper bounds of the expected running time of Algorithm 2 on the five problems are lower than those of MOEA/D-M an order of n. Note that this analysis result does not mean that the upper bounds of the expected running time of Algorithm 2 on these problems are proportionally increasing with the population size since our analyses neglect the promotion of the neighborhood-based coevolution, which is probably larger with increasing the population size.

IV. Experimental Study

In this section, we first conduct experiment to verify the correctness of expected upper bounds obtained in our analyses, then compare the efficiency of MOEA/D-C and MOEA/D-M on the five problems. The experiment is implemented in MATLAB R2014b and run on the PlatEMO platform [51].

A. Verification Experiment

To verify the correctness of theoretical bounds obtained in Theorem 6, we run numerical experiments on $n = 8, 16, 24, 32, 40, 52, 60, 70, 80, 90, 100$ while setting $N = 4$ on the five problems. For each problem, we independently run MOEA/D-C 30 times for each n. On the one hand, combined with (5), (9), and (10), we have that the exact upper bound obtained in our analyses for LOTZ, LPTNO, and Plateau-MOP is $N \cdot (2en^2 + 7.78n^2 + 4n \log n) \approx 53n^2$. On the other hand, we have that the exact upper bound obtained for COCZ and Dec-obj-MOP is $N \cdot (2en \log n + 4n \log n) \approx 38n \log n$ by combining (13) and (14). As shown in Figs. 8 and 9, not only are the curves of the average numbers of fitness evaluations in computational experiments on the five problems lower than the expected upper bounds obtained in our analyses but also the curves of the maximum numbers of fitness evaluations. Note that we have considered the worst-case input in the analyses. These results confirm the correctness of theoretical upper bounds of the expected running time obtained in our analyses.

We notice that compared with the gap between theoretical expected upper bounds and computational experiments on COCZ and Dec-obj-MOP (Fig. 8), the gap on LOTZ, LPTNO, and Plateau-MOP (Fig. 9) is a bit larger. The main reason is that when we prove the upper bound of Lemma 2, to tackle the worst case we need to neglect these progresses of individuals obtained in the first $2en^2$ fitness evaluations, that is, the expected running time to find the optimal solution for subproblems $g(\mathbf{x}|\lambda^1)$ and $g(\mathbf{x}|\lambda^N)$. And such a pessimistic assumption is not needed to prove the expected upper bound for COCZ and Dec-obj-MOP problems.

B. Comparison With MOEA/D-M

To further study the effect of the use of crossover operator in MOEA/D. We compare the average number of fitness evaluations of MOEA/D-C and MOEA/D-M on optimizing the five problems in computational experiments.

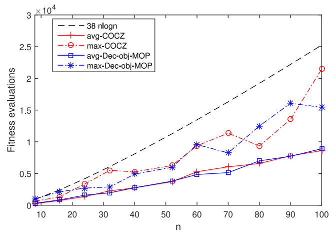


Fig. 8. Average and maximum numbers of fitness evaluations of MOEA/D-C require to cover the PFs of COCZ and Dec-obj-MOP.

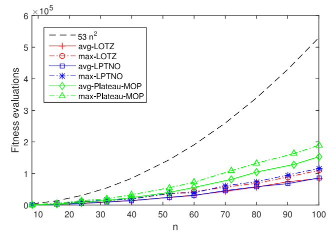


Fig. 9. Average and maximum numbers of fitness evaluations of MOEA/D-C require to cover the PFs of LOTZ, LPTNO, and Plateau-MOP.

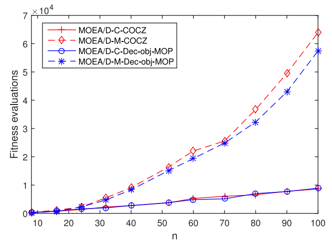


Fig. 10. Average number of fitness evaluations of MOEA/D-C and MOEA/D-M require to cover the PFs of COCZ and Dec-obj-MOP.

For all problems, the set of weight vectors of MOEA/D-C is $\{(0, 1), (1/3, 2/3), (2/3, 1/3), (1, 0)\}$. For MOEA/D-M, the set of weight vectors is set to the optimally decomposed weight vectors provided in [20]. More specifically, for LOTZ, COCZ, Dec-obj-MOP, and Plateau-MOP, the set of weight vectors is $\{\lambda^k = (\lambda_1^k, 1-\lambda_1^k), k = 0, \ldots, n\}$, where $\lambda_1^k = (k/n)$. The set of weight vectors for LPTNO is $\{\lambda^k = (\lambda_1^k, 1-\lambda_1^k), k = 0, \ldots, n\}$, where $\lambda_1^k = ([\sum_{j=1}^{k} 2^{n+1-j}]/[\sum_{j=k+1}^{n} 2^j + \sum_{j=1}^{k} 2^{n+1-j}])$. They proved that these weight vectors produce a set of subproblems that are mapped one-to-one to the points on the PFs. The expected upper bounds of MOEA/D-M with these weight vectors on optimizing the five problems are shown in Table I.

As shown in Figs. 10 and 11, the average number of fitness evaluations of MOEA/D-C to optimize the five problems are smaller than that of MOEA/D-M apparently. These results imply that crossover operator plays an important role in MOEA/D and make the algorithm more efficient on optimizing these problems. Another interesting observation on LPTNO is that MOEA/D-M always fails to obtain some points on the PF within two million fitness evaluations if $n \geq 60$. The reason is that the optimal values (2^{n+1}) of the two objectives increase rapidly with n such that lots of improved solutions increase objective value with a small number are neglected since the difference between them exceeds the accuracy of MATLAB. Thus, it often searches blindly in the optimization process.

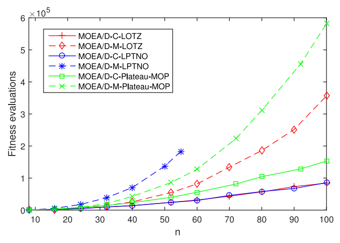


Fig. 11. Average number of fitness evaluations of MOEA/D-C and MOEA/D-M require to cover the PFs of LOTZ, LPTNO, and Plateau-MOP.

However, these situations do not affect the optimization process of MOEA/D-C and it still works effectively in the computational experiment. The probable reason for this phenomenon is that the crossover operator in MOEA/D-C helps individuals creating larger improvement.

V. Conclusion

In this paper, we present a theoretical analysis for the effect of crossover operator in the MOEA/D framework. Our rigorous running time analyses show that by properly setting the number of decomposed subproblems, the upper bounds of expected running time of the MOEA/D-C obtaining a set of solutions to cover the PFs of the five pseudo-Boolean functions are lower than those of MOEA/D-M an order of n. Computational experiments are also conducted to confirm the advantages of MOEA/D-C. Moreover, to effectively obtain a set of solution to cover the PF, MOEA/D-C just needs to decompose an MOP into several subproblems according to a set of simple weight vectors while MOEA/D-M needs to find $\Omega(n)$ optimally decomposed weight vectors. These results suggest 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 the algorithm on optimizing these problems.

This paper theoretically explains why some existing MOEAs based on decomposition work well in computational experiments and provides insights into the working principles of MOEA/D. In addition, it provides an idea to analyze the expected running time of MOEA/D on optimizing discrete optimization problems and evidence for that crossover is helpful and essential in the design of EAs.

To derive a lower bound of MOEA/D-C on these problems, one needs to estimate the promotion of the neighborhood-based coevolution. This is an interesting and challenging study in future work.

References

[1] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, 2011.

[2] Y. Xiang, Y. Zhou, M. Li, and Z. Chen, "A vector angle-based evolutionary algorithm for unconstrained many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 1, pp. 131–152, Feb. 2017.

[3] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 440–462, Jun. 2017.

[4] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.

[5] T. Murata and M. Gen, "Cellular genetic algorithm for multi-objective optimization," in *Proc. 4th Asian Fuzzy Syst. Symp.*, 2002, pp. 538–542.

[6] 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.

[7] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evol. Comput.*, vol. 22, no. 2, pp. 231–264, 2014.

[8] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Generalized decomposition and cross entropy methods for many-objective optimization," *Inf. Sci.*, vol. 282, pp. 363–387, Oct. 2014.

[9] X. He, Y. Zhou, Z. Chen, and Q. Zhang, "Evolutionary many-objective optimization based on dynamical decomposition," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 361–375, Jun. 2019.

[10] H. Li, J. Sun, Q. Zhang, and Y. Shui, "Adjustment of weight vectors of penalty-based boundary intersection method in MOEA/D," in *Evolutionary Multi-Criterion Optimization*, K. Deb *et al.*, Eds. Cham, Switzerland: Springer Int., 2019, pp. 91–100.

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

[12] R. Wang, Q. Zhang, and T. Zhang, "Decomposition-based algorithms using Pareto adaptive scalarizing methods," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 821–837, Dec. 2016.

[13] X. Ma, Q. Zhang, G. Tian, J. Yang, and Z. Zhu, "On Tchebycheff decomposition approaches for multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 226–244, Apr. 2018.

[14] D. Han, W. Du, W. Du, Y. Jin, and C. Wu, "An adaptive decomposition-based evolutionary algorithm for many-objective optimization," *Inf. Sci.*, vol. 491, pp. 204–222, Jul. 2019.

[15] T.-C. Chiang and Y.-P. Lai, "MOEA/D-AMS: Improving MOEA/D by an adaptive mating selection mechanism," in *Proc. Evol. Comput.*, 2011, pp. 1473–1480.

[16] K. Li, S. Kwong, Q. Zhang, and K. Deb, "Interrelationship-based selection for decomposition multiobjective optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2076–2088, Oct. 2015.

[17] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 169–190, Apr. 2017.

[18] R. Tanabe and H. Ishibuchi, "An analysis of control parameters of MOEA/D under two different optimization scenarios," *Appl. Soft Comput.*, vol. 70, pp. 22–40, Sep. 2018.

[19] R. Wang, J. Xiong, H. Ishibuchi, G. Wu, and T. Zhang, "On the effect of reference point in MOEA/D for multi-objective optimization," *Appl. Soft Comput.*, vol. 58, pp. 25–34, Sep. 2017.

[20] Y.-L. Li, Y.-R. Zhou, Z.-H. Zhan, and J. Zhang, "A primary theoretical study on decomposition-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 563–576, Aug. 2016.

[21] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb, "Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2002, pp. 44–53.

[22] 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.

[23] F. Neumann, "Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1620–1629, 2007.

[24] T. Friedrich, N. Hebbinghaus, F. Neumann, J. He, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," *Evol. Comput.*, vol. 18, no. 4, pp. 617–633, 2010.

[25] O. Giel and P. K. Lehre, "On the effect of populations in evolutionary multi-objective optimisation," *Evol. Comput.*, vol. 18, no. 3, p. 335, 2010.

[26] F. Neumann, J. Reichel, and M. Skutella, "Computing minimum cuts by randomized search heuristics," *Algorithmica*, vol. 59, no. 3, pp. 323–342, 2011.

[27] T. Friedrich and F. Neumann, "Maximizing submodular functions under matroid constraints by evolutionary algorithms," *Evol. Comput.*, vol. 23, no. 4, pp. 543–558, 2015.

[28] F. Daolio, A. Liefooghe, S. Verel, H. Aguirre, and K. Tanaka, "Problem features versus algorithm performance on rugged multiobjective combinatorial fitness landscapes," *Evol. Comput.*, vol. 25, no. 4, pp. 555–585, 2017.

[29] D. Sudholt, “How crossover speeds up building-block assembly in genetic algorithms,” *Evol. Comput.*, vol. 25, no. 2, pp. 237–274, 2017.

[30] W. Jansen, “The analysis of evolutionary algorithms—A proof that crossover really can help,” *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2002.

[31] T. Kötzing, D. Sudholt, and M. Theile, “How crossover helps in pseudo-Boolean optimization,” in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, 2011, pp. 989–996.

[32] P. K. Lehre and X. Yao, “Crossover can be constructive when computing unique input–output sequences,” *Soft Comput.*, vol. 15, no. 9, pp. 1675–1687, 2011.

[33] B. Doerr, E. Happ, and C. Klein, “Crossover can provably be useful in evolutionary computation,” *Theor. Comput. Sci.*, vol. 425, pp. 17–33, Mar. 2012.

[34] D. Sudholt, “Crossover speeds up building-block assembly,” in *Proc. 14th Annu. Conf. Genet. Evol. Comput.*, 2012, pp. 689–702.

[35] B. Doerr, D. Johannsen, T. Kötzing, F. Neumann, and M. Theile, “More effective crossover operators for the all-pairs shortest path problem,” *Theor. Comput. Sci.*, vol. 471, pp. 12–26, Feb. 2013.

[36] B. Doerr, C. Doerr, and F. Ebel, “From black-box complexity to designing new genetic algorithms,” *Theor. Comput. Sci.*, vol. 567, pp. 87–104, Feb. 2015.

[37] B. Doerr and C. Doerr, “Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm,” *Algorithmica*, vol. 80, no. 5, pp. 1658–1709, 2018.

[38] D.-C. Dang *et al.*, “Escaping local optima using crossover with emergent diversity,” *IEEE Trans. Evol. Comput.*, vol. 22, no. 3, pp. 484–497, Jun. 2018.

[39] F. Neumann and M. Theile, “How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem,” in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2010, pp. 667–676.

[40] C. Qian, Y. Yu, and Z.-H. Zhou, “An analysis on recombination in multi-objective evolutionary optimization,” *Artif. Intell.*, vol. 204, pp. 99–119, Nov. 2013.

[41] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12. New York, NY, USA: Springer, 1999.

[42] 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.

[43] I. Das and J. Dennis, “Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems,” *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[44] A. Inselberg, “The plane with parallel coordinates,” *Vis. Comput.*, vol. 1, no. 2, pp. 69–91, 1985.

[45] Z. He and G. G. Yen, “Visualization and performance metric in many-objective optimization,” *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 386–402, Jun. 2016.

[46] 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.

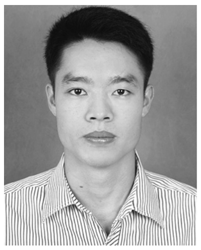
[47] T. Jansen, *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Heidelberg, Germany: Springer, 2013.

[48] Y. 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, 2009.

[49] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, “Multiobjective optimization test instances for the CEC 2009 special session and competition,” School Comput. Sci. Electron. Eng., Univ. Essex, Colchester, U.K., Rep. CES-487, 2008.

[50] D. A. Van, V. Gary, and B. Lamont, “Multiobjective evolutionary algorithm research: A history and analysis,” *Evol. Comput.*, vol. 8, no. 2, pp. 125–147, 1999.

[51] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum],” *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, Nov. 2017.

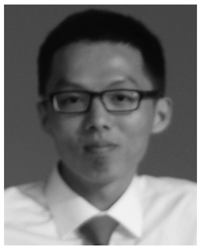


Zhengxin Huang received the M.Sc. degree in computational mathematics from the Guangxi University for Nationalities, Nanning, China, in 2011. He is currently pursuing the Ph.D. degree with Sun Yat-sen University, Guangzhou, China.

He is also a Lecturer with the Department of Computer Science and Information Technology, Youjiang Medical University for Nationalities, Baise, China. His current research interests include evolutionary algorithm design and theoretical analysis, and their applications in real world.

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



Zefeng Chen received the B.Sc. degree in information and computational science from Sun Yat-sen University, Guangzhou, China, in 2013, the M.Sc. degree in computer science and technology from the South China University of Technology, Guangzhou, in 2016, and the Ph.D. degree in computer science from Sun Yat-sen University in 2019.

His current research interests include evolutionary computation, multiobjective optimization, data mining, and machine learning.

Xiaoyu He received the B.Eng. degree in computer science and technology from Beijing Electronic Science and Technology Institute, Beijing, China, in 2010, the M.Sc. degree in public administration from the South China University of Technology, Guangzhou, China, in 2016, and the Ph.D. degree in computer science from Sun Yat-sen University, Guangzhou, in 2019.

His current research interests include evolutionary computation and data mining.

Xinsheng Lai received the M.Sc. degree in computer applied technology from Guizhou University, Guiyang, China, in 2004 and the Ph.D. degree in computer applied technology from the South China University of Technology, Guangzhou, China, in 2014.

He is currently a Professor with the School of Mathematics and Computer Science, Shangrao Normal University, Shangrao, China. His current research interests include evolutionary computation, neural computation, and their applications in real world.

Xiaoyun Xia received the M.Sc. degree in computer applied technology from the Jiangxi University of Science and Technology, Ganzhou, China, in 2007 and the Ph.D. degree in computer software and theory from the South China University of Technology, Guangzhou, China, in 2015.

He is currently an Associate Professor with the College of Mathematics Physics and Information Engineering, Jiaxing University, Jiaxing, China. His current research interests include evolutionary computation, intelligent computation, and global optimization.