A Runtime Analysis of Typical Decomposition Approaches in MOEA/D Framework for Many-objective Optimization Problems

Zhengxin Huang[1,2], **Yuren Zhou**[1*], **Chuan Luo**[3] and **Qingwei Lin**[3]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]Department of Computer Science, Youjiang Medical University for Nationalities, Baise, China
[3]Microsoft Research, China
huangzhx26@mail2.sysu.edu.cn, yurenzhou@mail.sysu.edu.cn, {chuan.luo, qlin}@microsoft.com

Abstract

Decomposition approach is an important component in multi-objective evolutionary algorithm based on decomposition (MOEA/D), which is a popular method for handing many-objective optimization problems (MaOPs). This paper presents a theoretical analysis on the convergence ability of using the typical weighted sum (WS), Tchebycheff (TCH) or penalty-based boundary intersection (PBI) approach in a basic MOEA/D for solving two benchmark MaOPs. The results show that using WS, the algorithm can always find an optimal solution for any subproblem in polynomial expected runtime. In contrast, the algorithm needs at least exponential expected runtime for some subproblems if using TCH or PBI. Moreover, our analyses discover an obvious shortcoming of using WS, that is, the optimal solutions of different subproblems are easily corresponding to the same solution. In addition, this analysis indicates that if using PBI, a small value of the penalty parameter is a good choice for faster converging to the Pareto front, but it may lose the diversity. This study reveals some optimization behaviors of using three typical decomposition approaches in the well-known MOEA/D framework for solving MaOPs.

1 Introduction

Multi-objective optimization problems (MOPs) involve optimizing multiple conflicting objective functions simultaneously. MOPs with at least four objectives widely exist in real-word applications and are referred to as many-objective optimization problems (MaOPs) [Li *et al.*, 2015a]. Multi-objective evolutionary algorithms (MOEAs) are popular methods for handling MOPs, and can be roughly classified into three categories, i.e., domination-based, indicator-based and decomposition-based [Trivedi *et al.*, 2017].

Domination-based MOEAs, such as NSGA-II [Deb *et al.*, 2002], are quite effective in solving low-dimensional MOPs. But they often suffer from dramatic performance degeneration when handling MaOPs, because of insufficient selection pressure toward the Pareto front [Li *et al.*, 2015b]. Due to the computation time of indicators rapidly increasing with the number of objectives, indicator-based MOEAs are largely restricted to solve MaOPs [Deng and Zhang, 2019]. Although these defects have been alleviated in some researches, (e.g., [Yuan *et al.*, 2016] and [Rostami and Neri, 2017]), decomposition-based MOEAs have attracted the most attention and are very popular for solving MaOPs in the past decade [Trivedi *et al.*, 2017].

Decomposition-based MOEAs have been implemented in [Ishibuchi and Murata, 1998]. They became popular after [Zhang and Li, 2007] proposed the classic MOEA/D framework. In the MOEA/D framework, an MaOP is first decomposed into a number of scalar subproblems according to the used decomposition approach and a set of weight vectors, then all subproblems are solved in parallel and collaboratively by using an evolutionary algorithm (EA). Decomposition approach is an important component in MOEA/D framework, since it determines the analytical expression formula of the subproblems and guides the evolutionary search [Wang *et al.*, 2020]. The typical decomposition approaches used in MOEA/D framework are weighted sum (WS), Tchebycheff (TCH) [Das and Dennis, 1998], and penalty-based boundary intersection (PBI) [Zhang and Li, 2007]. Many researches have developed novel decomposition approaches based on them, e.g., [Gómez and Coello, 2017; Luque *et al.*, 2020].

Running time or runtime analysis is an essential and powerful theory aspect to understand the work principles of EAs [Qian *et al.*, 2019]. Li *et al.* [2016] presented a runtime analysis of a simple algorithm based on MOEA/D framework, which only applies mutation operator to create offspring. Later, Huang *et al.* [2020] extended the analysis to the universal case that selects mutation or crossover to create offspring with a control parameter. Recently, a runtime analysis of a simple MOEA/D using somatic contiguous hypermutation operators [Jansen and Zarges, 2011] to create offspring was presented in [Huang and Zhou, 2020]. These theoretical studies have revealed some important optimization behaviors of decomposition-based MOEAs. However, they only consider the case of using TCH approach and the number of objectives in their analyzed problems is at most four.

This paper presents a runtime analysis for the convergence ability of using three typical decomposition approaches in a basic MOEA/D on optimizing benchmark MaOPs. Specifi-

*Corresponding author

cally, we analyze and compare the expected runtime of a basic decomposition-based MOEA with WS, TCH or PBI finding an optimal solution for any subproblem of two benchmark MaOPs, namely mLOTZ and mCOCZ [Laumanns *et al.*, 2004]. Note that in the two problems, m represents the number of the objectives and it is a variable. They have been served as benchmark problems for theoretical analyses of MOEAs, e.g., [Laumanns *et al.*, 2004; Bian *et al.*, 2018].

The theoretical results show that using WS, the algorithm can always find an optimal solution for any subproblem of mLOTZ and mCOCZ in polynomial expected runtime $O(\frac{n^2 \log n}{m})$ and $\Theta(n \log n)$, respectively. When using TCH, the expected lower bounds of the algorithm are $\Omega(\frac{n^2}{m})$ and $\Omega(n \log n)$, respectively. But the expected upper bounds are $O(n^2 m^{m/2})$ and $O(\frac{n^{m/2+1}}{m})$, respectively. When using PBI, the expected runtime is dependent on the setting of the penalty parameter θ. In the best case (i.e., $\theta = 0$), they are equal to that of using WS. In the worst case, the algorithm cannot find an optimal solution for some subproblems if θ is set too large. Moreover, our analyses discover an obvious shortcoming of using WS, i.e., the optimal solutions of different subproblems are easily corresponding to the same solution. In addition, our analyses indicate that increasing parameter θ is harmful for faster converging to the PF when using PBI, but it can alleviate the shortcoming of WS. This theoretical study reveals some important optimization behaviors of using the three typical decomposition approaches in MOEA/D for solving MaOPs, and may be helpful for designing more efficient MOEAs in future research.

2 Preliminaries

2.1 Analyzed Problems

Formally, a maximization MOP can be defined as follows:

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

where x is the decision variable, X is the decision space, $F : X \to R^m$ consists of m objective functions and R^m is the objective space. For $x_1, x_2 \in X$, we say that x_1 weakly dominates x_2, denoted as $x_1 \succeq x_2$, if $f_i(x_1) \geq f_i(x_2)$ for all $i = 1, \ldots, m$. If $x_1 \succeq x_2$ and $F(x_1) \neq F(x_2)$, we say that x_1 dominates x_2, denoted as $x_1 \succ x_2$. A solution $x^* \in X$ is Pareto optimal if there is no solution $x \in X$ such that $x \succ x^*$. The set of all Pareto optimal solutions is called *Pareto-optimal set* or *Pareto set* (PS) and the set of objective vectors corresponding to the PS is called *Pareto front* (PF).

Let x be a bit string of length n, i.e., $x \in \{0,1\}^n$, and $x[i]$ denote the value of the i-th bit of x for $i = 1, 2, \ldots, n$. The benchmark MaOPs (i.e., mLOTZ and mCOCZ [Laumanns *et al.*, 2004]) analyzed in this paper can be defined as follows.

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

$$mLOTZ(x) = \big(f_1(x), f_2(x), \ldots, f_m(x)\big),$$

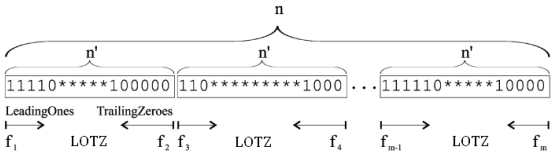

Figure 1: The schematic view of mLOTZ.

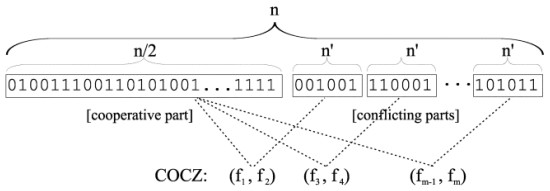

Figure 2: The schematic view of mCOCZ.

where

$$f_k(x) = \begin{cases} \sum_{i=1}^{n'} \prod_{j=1}^{i} x[j + n' \frac{(k-1)}{2}] & \textit{if k is odd,} \\ \sum_{i=1}^{n'} \prod_{j=i}^{n'} (1 - x[j + n' \frac{(k-2)}{2}]) & \textit{otherwise,} \end{cases}$$

m is an even number, n' is a natural number and $n' = \frac{2n}{m}$.

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

$$mCOCZ(x) = \big(f_1(x), f_2(x), \ldots, f_m(x)\big),$$

where

$$f_k(x) = \sum_{i=1}^{n/2} x[i] + \begin{cases} \sum_{j=1}^{n'} x[j + \frac{n + n'(k-1)}{2}] & \textit{if k is odd,} \\ \sum_{j=1}^{n'} (1 - x[j + \frac{n + n'(k-2)}{2}]) & \textit{otherwise,} \end{cases}$$

m is an even number, n' is a natural number and $n' = \frac{n}{m}$.

Note that all objective functions in mLOTZ and mCOCZ require to be maximized simultaneously. For mLOTZ, it is a concatenation of $\frac{m}{2}$ LOTZ (leading ones trailing zeros) [Laumanns *et al.*, 2004]. There are $\frac{m}{2}$ conflicting parts in the search space, as shown in Figure 1 [Laumanns *et al.*, 2004]. For mCOCZ, it is a combination of $\frac{m}{2}$ COCZ [Laumanns *et al.*, 2004]. There are cooperative and conflicting parts in the solution space, as shown in Figure 2 [Laumanns *et al.*, 2004].

2.2 Decomposition Approaches

In MOEA/D, an MOP is first decomposed into a number of scalar optimization subproblems according to the decomposition approach and a set of weight vectors. WS, TCH and PBI are classic decomposition approaches used in MOEA/D framework. Given a maximization MOP and a weight vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ satisfying $\lambda_i \geq 0$ for $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$, the subproblems generated by WS, TCH and PBI are shown in Eq. (2), Eq. (3) and Eq. (4), respectively.

$$\max g^{ws}(x|\boldsymbol{\lambda}) = \sum_{k=1}^{m} \lambda_k f_k(x) \tag{2}$$

Algorithm 1 A simple decomposition-based MOEA

Input: An MOP with m objectives, stop criterion, weight vectors $\{\lambda^1, \ldots, \lambda^N\}$ and neighbor size T.
Output: A candidate Pareto optimal solution set P.

1: *Initialization:* A candidate Pareto optimal solution set $P = \emptyset$. Decompose the original MOP into N scalar subproblems according to the used decomposition approach and weight vectors $\{\lambda^1, \ldots, \lambda^N\}$. For each subproblem $g(x|\lambda^k), k = 1, 2, \ldots, N$, select the T closest subproblems to form its neighbor set B_k according to the Euclidean distance between their weight vectors. Generate solution $x_k \in \{0,1\}^n$ uniformly at random for each $g(x|\lambda^k)$. Let S_k denote the set of solutions corresponding to the subproblems in B_k.
2: **while** stop criterion is not satisfied **do**
3: **for** each subproblem $g(x|\lambda^k), k = 1, 2, \ldots, N$ **do**
4: *Reproduction*: create offspring y_k by randomly flipping one bit in x_k.
5: *Update* S_k*:* for each solution x_j in S_k, if $g(y_k|\lambda^j)$ is not worse then $g(x_j|\lambda^j)$, replace x_j with y_k.
6: *Update* P*:* remove all solutions weakly dominated by y_k from P. If y_k is not dominated by any solution in P, add y_k into P.
7: **end for**
8: **end while**

$$\min g^{tch}(x|\boldsymbol{\lambda}, z^*) = \max_{1\le i\le m}\{\lambda_k|f_k(x) - z_i^*|\} \tag{3}$$

where $z^* = (z_1^*, \ldots, z_m^*)$ denotes the reference point, i.e., $z_i^* = \max\{f_i(x)|x \in X\}$.

$$\min\ g^{pbi}(x|\boldsymbol{\lambda}, z^*) = d_1 + \theta d_2 \tag{4}$$

where z^* has the same meaning with Eq. (3), $\theta > 0$ is the penalty parameter and

$$d_1 = \frac{\|(F(x)-z^*)^T\lambda\|}{\|\boldsymbol{\lambda}\|},$$
$$d_2 = \|F(x) - (z^* - d_1\boldsymbol{\lambda})\|.$$

As shown above, for a given weight vector, different decomposition approaches transform an MOP into different scalar subproblems and they largely affect the optimization.

2.3 Analyzed Algorithm

After decomposition, MOEA/D framework controls a population of size N to cooperatively solve all subproblems in parallel via the neighborhood-based coevolution. The analyzed algorithm in this paper is summarized in Algorithm 1. Note that Algorithm 1 can be instantiated by using different decomposition approaches in the initialization (line 1). This paper mainly concerns on the expected runtime of Algorithm 1 with WS, TCH or PBI approach converging to an optimal solution of any subproblem for mLOTZ and mCOCZ. To simplify the observation, the one-bit mutation operator [Zhou *et al.*, 2019], which flips a randomly chosen bit of the parent in a mutation, is used to create offspring in Algorithm 1.

3 Runtime Analysis

Since we mainly focus on the convergence speed of using the three decomposition approaches, we afterward assume that $\lambda_{i=1,2,\ldots,m} > 0$ holds for weight vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ and set the neighbor size $T = 1$ for the sake of simplicity.

3.1 Analysis on WS

Recall that mLOTZ is a concatenation of $\frac{m}{2}$ LOTZ problems (see Figure 1). For ease of expression, we call the j-th LOTZ as component/part j in the following analysis. Let 1^i0^{h-i} denote the bit-string of length h with i leading 1-bits and $h-i$ trailing 0-bits.

Lemma 1. *For mLOTZ, if Algorithm 1 uses WS decomposition approach, all components in the optimal solution for any subproblem are in the form of $1^i0^{\frac{2n}{m}-i}$.*

Proof. Given weight vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$, according to Eq. (2) the subproblem generated by WS for mLOTZ is

$$\max g^{ws}(x|\boldsymbol{\lambda}) = \sum_{k=1}^{m}\lambda_k f_k(x), \tag{5}$$

where $f_k(x)$ equals the number of leading 1-bits in component $\lceil\frac{k}{2}\rceil$ if k is odd, otherwise it equals the number of trailing 0-bits in this component (see Definition 1).

We first show that all components in any optimal solution are in the form of $1^i0^{\frac{2n}{m}-i}$. Otherwise there is at least a component j in it in the form of $1^{l_1}0\#10^{l_2}$, where $\#$ denotes the wildcard string. If the mutation operator flips the leftmost 0-bit or the rightmost 1-bit, it becomes $1^{l_1+1}0\#10^{l_2}$ or $1^{l_1}0\#10^{l_2+1}$. The function value of the offspring on objective f_{2j-1} or f_{2j} is larger than its parent, and its function values on other objectives are unchanged. Thus, Algorithm 1 finds a better function value for Eq. (5). This contradicts with the assumption that the current solution is an optimum.

Furthermore, we consider the two values in weight vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ related to component j, i.e., λ_{2j-1} and λ_{2j}. We have $\lambda_{2j-1} > \lambda_{2j}, \lambda_{2j-1} < \lambda_{2j}$ or $\lambda_{2j-1} = \lambda_{2j}$. Let $x^j = 1^i0^{\frac{2n}{m}-i}, i = 0\ldots, \frac{2n}{m} - 1$ denote component j in the current solution.

If $\lambda_{2j-1} > \lambda_{2j}$ and the mutation operator flips the leftmost 0-bit in x^j, it becomes $y^j = 1^{i+1}0^{\frac{2n}{m}-i-1}$. The offspring is an improved solution for Eq. (5), because of $\lambda_{2j-1}\cdot f_{2j-1}(y^j) + \lambda_{2j}\cdot f_{2j}(y^j) - \lambda_{2j-1}\cdot f_{2j-1}(x^j) - \lambda_{2j}\cdot f_{2j-1}(x^j) = \lambda_{2j-1} - \lambda_{2j} > 0$. By analogy, Algorithm 1 can repeatedly create an improved solution for Eq. (5) by flipping the leftmost 0-bit in x^j into 1-bit until it becomes the optimal form $1^{\frac{2n}{m}}$. Similarly, we know that if $\lambda_{2j-1} < \lambda_{2j}$, Algorithm 1 can repeatedly create an improved solution for Eq. (5) by flipping the rightmost 1-bit into 0-bit until it becomes $0^{\frac{2n}{m}}$.

If $\lambda_{2j-1} = \lambda_{2j}$, for any two bit-strings $s_1 = 1^{i_1}0^{\frac{2n}{m}-i_1}$ and $s_2 = 1^{i_2}0^{\frac{2n}{m}-i_2}$, we have $\lambda_{2j-1}\cdot f_{2j-1}(s_1) + \lambda_{2j}\cdot f_{2j}(s_1) - \lambda_{2j-1}\cdot f_{2j-1}(s_2) - \lambda_{2j}\cdot f_{2j}(s_2) = 0$. Thus, in this case all bit-strings in the form of $1^i0^{\frac{2n}{m}-i}$ have the same contribution to the function value of Eq. (5) and are optimal form. □

Theorem 1. *For mLOTZ, using WS decomposition approach, the lower bound and upper bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem are $\Omega(\max\{\frac{n^2}{m}, n\log n\})$ and $O(\frac{n^2\log m}{m})$, respectively.*

Proof. First of all, the expected Hamming distance of any initial solution from an optimal solution is $E(d) = n\cdot\frac{1}{2} = \frac{n}{2}$. Recall that all initial solutions are generated from $\{0,1\}^n$ uniformly at random, and all components in an optimal solution are in the form of $1^i0^{\frac{2n}{m}-i}$. By Chernoff bound, we have $\Pr(E(d) \leq \frac{n}{4}) \leq e^{-\Theta(n)}$. Thus, the expected number of bits in any initial solution needed to be corrected before it becomes an optimal one is at least $\frac{n}{4}$ with probability $1-e^{-\Omega(n)}$.

Moreover, given any component, if it is not in the form of $1^i0^{\frac{2n}{m}-i}$, Algorithm 1 can create an improved offspring by flipping the leftmost 0-bit or the rightmost 1-bit in them (refer to the proof of Lemma 1). If it is in the form of $1^i0^{\frac{2n}{m}-i}$ but not an optimal form, Algorithm 1 can obtain an improvement by flipping either the leftmost 0-bit or the rightmost 1-bit in them. Thus, in a generation the probability of Algorithm 1 creating an improved solution for Eq. (5) is at least $\frac{h}{n}$ and at most $\frac{2h}{n}$, where h denotes the number of components that are not in its optimal form in the current solution.

By using the optimistic assumption that the $\frac{n}{4}$ bits needed to be corrected are done with the highest possible probability, i.e., $\frac{2h}{n} = \frac{m}{n}$, the expected runtime of Algorithm 1 finding an optimal solution for Eq. (5) is lower bounded by $\frac{n}{4}\cdot\frac{n}{m} = \Omega(\frac{n^2}{m})$. In addition, according to the proof of Theorem 5.11 in [Jansen, 2013], we know that the $\frac{n}{4}$ bits needed to be corrected have not been done with constant probability within $(n-1)\log n$ fitness evaluations. Thus, the lower bound of expected runtime is $\Omega(\max\{\frac{n^2}{m}, n\log n\})$.

To derive the upper bound, we need to consider the worst case that all bits in the initial solution are needed to be corrected and they are done by using the maximum possible probability with minimum quantity. Observe that in this case Algorithm 1 can create improved offspring with the highest possible probability, namely $\frac{m}{n}$, until the number of components that are not in form of $1^i0^{\frac{2n}{m}-i}$ is less than $\frac{m}{2}$. This implies that at least all bits in a component are done with this probability and the number of Algorithm 1 creating improved solution with the highest probability is at least $\frac{2n}{m}$. Similarly, we know that in the worst case the number of Algorithm 1 creating improved solution with probability $\frac{m-1}{n}$ is also $\frac{2n}{m}$, and so on until $\frac{1}{n}$. Therefore, the expected runtime of Algorithm 1 finding an optimal solution for Eq. (5) is upper bounded by $\sum_{j=1}^{m/2}\frac{2n}{m}\cdot\frac{n}{j} \leq \sum_{j=1}^{m}\frac{2n}{m}\cdot\frac{n}{j} = O(\frac{n^2\log m}{m})$. □

Lemma 2. *For mCOCZ, using WS approach, the lower bound of expected runtime of Algorithm 1 finding an optimal solution for every subproblem is $\Omega(n\log n)$.*

For mCOCZ, the subproblem generated by WS for weight vector $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_m)$ is

$$\max g^{ws}(x|\boldsymbol{\lambda}) = \sum_{k=1}^{m}\lambda_k f_k(x), \tag{6}$$

where $f_k(x)$ is shown in Definition 2.

Observe that for Eq. (6), all $f_k(x)$ increase when increasing the number of 1-bits in the first half of x (the cooperative part). This implies that a necessary condition of an initial solution becoming an optimum is that the first $\frac{n}{2}$ bits are transformed into 1-bits. Thus, we can conclude this result according to the proof of Theorem 5.11 in [Jansen, 2013]. We omit the detailed proof for the sake of space limitation.

Lemma 3. *For mCOCZ, using WS approach, the upper bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem is $O(n\log n)$.*

Proof. Let $i < \frac{n}{2}$ denote the number of 1-bits in the first half of the initial solution. In a generation, if one of the i 0-bits is flipped, Algorithm 1 obtains an improvement for Eq. (6). This event happens with probability $\frac{i}{n}$. Thus, the expected runtime of Algorithm 1 transforming all bits in the cooperative part into 1-bits is upper bounded by $\sum_{i=1}^{n/2-1}\frac{n}{i} = O(n\log n)$.

We next consider the conflicting part. Given any component j, similar to analysis in the proof of Lemma 1, we know that its optimal form is $1^{\frac{n}{m}}$ if $\lambda_{2j-1} > \lambda_{2j}$ and is $0^{\frac{n}{m}}$ if $\lambda_{2j-1} < \lambda_{2j}$. Otherwise, it can be any bit-string with length $\frac{n}{m}$. Let d_j denote the Hamming distance of the current component j from its optimal form. Observe that in a generation Algorithm 1 can create an improved offspring for Eq. (6) by flipping one of the d_j wrong bits in this part. Thus, in a generation Algorithm 1 finds an improved solution for Eq. (6) with probability is $\frac{d_1+\cdots+d_{m/2}}{n}$. Note that $1 \leq \sum_{i=1}^{\frac{m}{2}} d_j \leq \frac{n}{2}$ holds if the current solution is not an optimum. Therefore, Algorithm 1 transforms the conflicting part into an optimal form in expected runtime $\sum_{i=1}^{\frac{n}{2}}\frac{n}{i} = O(n\log n)$. □

Combining Lemma 2 and Lemma 3, we have Theorem 2.

Theorem 2. *For mCOCZ, using WS decomposition approach, the expected runtime of Algorithm 1 finding an optimal solution for any subproblem is $\Theta(n\log n)$.*

3.2 Analysis on TCH

Lemma 4. *For mLOTZ, using TCH approach, the expected lower bound of Algorithm 1 finding an optimal solution for any subproblem is $\Omega(\frac{n^2}{m})$ with probability $1-e^{-\Omega(\frac{n}{m})}$.*

Proof. For mLOTZ, the reference point is $z^* = (\frac{2n}{m},\ldots,\frac{2n}{m})$. Given weight vector $\boldsymbol{\lambda} = (\lambda_1,\ldots,\lambda_m)$, according to Eq. (3) the subproblem produced by TCH is

$$\min g^{tch}(x|\boldsymbol{\lambda}, z^*) = \max_{1\leq k\leq m}\{\lambda_k\big(\frac{2n}{m} - f_k(x)\big)\}, \tag{7}$$

where $f_k(x)$ is the same as that in Eq (5).

Observe that Eq. (7) requires to find a x^* such that $x^* = \arg\min_{x\in X}\{\max_{k=1\ldots,m}\{\lambda_k\big(\frac{2n}{m} - f_k(x)\big)\}\}$. For any x, let $V_{max}(x) := \max_{k=1,\ldots,m}\{\lambda_k\big(\frac{2n}{m} - f_k(x)\big)\}$ and $I_{max}(x) = \{k|\lambda_k\big(\frac{2n}{m} - f_k(x)\big) \geq V_{max}(x)\}$. Note that for mLOTZ there is no solution that can increase multiple function objective values simultaneously. Thus, in a generation Algorithm 1 creates an improved offspring x' from parent x if and only if: (1) $|I_{max}(x)| = 1$ (denote by k' the element in $I_{max}(x)$); (2) the one-bit mutation flipping one bit in the component $\lceil\frac{k'}{2}\rceil$ of

x such that $f_{k'}(x') > f_{k'}(x)$, i.e., either the leftmost 0-bit or the rightmost 1-bit in the part $\lceil \frac{k'}{2} \rceil$ is flipped (denoted as event E_1). Thus, in a generation event E_1 happens with probability at most $\frac{2}{n}$ and the expected runtime of Algorithm 1 obtaining such an improvement is $\Omega(n)$.

We next show that the expected number of improvements for finding an optimal solution is $\Omega(\frac{n}{m})$. First, for any optimal solution of Eq. (7), the part $\lceil \frac{k'}{2} \rceil$ must be in the form of $1^i 0^{\frac{2n}{m}-i}$. Otherwise, it is $1^{l_1}0\#10^{l_2}$. If the rightmost 1-bit is flipped, the function value on objective $f_{k'}$ is increased. This means that a better solution is found, which contradicts with the assumption. Second, according to analysis in the proof of Theorem 1, we have that to transform any component into the form of $1^i 0^{\frac{2n}{m}-i}$, the number of bits that needed to be corrected is at least $\frac{n}{2m}$ with probability $1 - e^{\Omega(\frac{n}{m})}$.

Therefore, with probability $1 - e^{-\Omega(\frac{n}{m})}$ the expect runtime of Algorithm 1 using TCH approach finding an optimal solution for any subproblem is $\Omega(n) \cdot \Omega(\frac{n}{m}) = \Omega(\frac{n^2}{m})$. □

Lemma 5. *For mLOTZ, using TCH approach, the expected upper bound of Algorithm 1 finding an optimal solution for any subproblem is $O(n^2 m^{m/2-2})$.*

Proof. We consider the random process that Algorithm 1 obtains an improved offspring starting from solution x with $|I_{max}(x)| = i$, $i \in \{1, \ldots, m\}$. Let $p_{i,i-1}$ be the probability of Algorithm 1 creating an offspring x' from x such that $|I_{max}(x')| = i - 1$. Observe that to delete an element from $I_{max}(x)$, Algorithm 1 needs to flip either the leftmost 0-bit or the rightmost 1-bit in the related component. So there are exactly i bits which flipping one of them can decrease $|I_{max}(x)|$ by one. Thus, we have $p_{i,i-1} = \frac{i}{n}$. However, in this situation Algorithm 1 can also create an offspring y from x such that $|I_{max}(y)| = i + 1$, if it flips one of $m - i$ bits in x leading to the value of an objective not in $I_{max}(x)$ increase to $V_{max}(x)$ (denoted as event E_2). The probability of event E_2 is $\frac{m-i}{n}$. Note that y is also accepted to replace x in such a mutation, because it has the same fitness value with the parent. In other words, to obtain an improved solution for Eq. (7), Algorithm 1 needs to pass through a plateau including at most m points, which is equivalent to reaching the absorbing state of Markov chain shown in Figure 3. The transition probability matrix $\mathbf{P}$ of the Markov chain is

$$\mathbf{P} = \begin{pmatrix} 1-\frac{m}{n} & \frac{m}{n} & 0 & \cdots & 0 & 0 \\ \frac{1}{n} & 1-\frac{m}{n} & \frac{m-1}{n} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \frac{m-1}{n} & 1-\frac{m}{n} & \frac{1}{n} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}.$$

Let T_i denote the expected runtime of Algorithm 1 reaching the absorbing sate from state $i = 0, 1, \ldots, m$, and use p_i^- and p_i^+ as shorthand for $p_{i,i-1}$ and $p_{i,i+1}$ in $\mathbf{P}$, respectively. According to Corollary 2 in [Zhou *et al.*, 2009], we have

$$T_i = \begin{cases} 0, i = 0 \\ T_{i-1} + \frac{1}{p_i^-} + \sum_{j=0}^{m-1-i} \frac{1}{p_{i+j+1}^-} \prod_{h=0}^{j} \frac{p_{i+h}^+}{p_{i+h}^-}, 0 < i < m \\ T_{m-1} + \frac{1}{p_m^-}, i = m \end{cases} \quad (8)$$

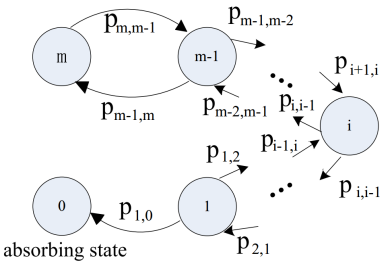

Figure 3: The Markov chain of finding an improved solution for Eq. (7) from the current one.

Let $l = m - 1 - i$. We have $i = m - 1 - l$ and $\sum_{j=0}^{m-1-i} \frac{1}{p_{i+j+1}^-} \prod_{h=0}^{j} \frac{p_{i+h}^+}{p_{i+h}^-} = \sum_{j=0}^{l} \frac{1}{p_{m-l+j}^-} \prod_{h=0}^{j} \frac{p_{m-1-l+h}^+}{p_{m-1-l+h}^-}$. Let $g_l := \sum_{j=0}^{l} \frac{1}{p_{m-l+j}^-} \prod_{h=0}^{j} \frac{p_{m-1-l+h}^+}{p_{m-1-l+h}^-}$. We have $g_0 = \frac{n}{m(m-1)}$ and $g_{l+1} = \frac{p_{m-1-(l+1)}^+}{p_{m-1-(l+1)}^-}\left(\frac{1}{p_{m-(l+1)}^-} + g_l\right)$. By Eq. (8), the expected runtime of Algorithm 1 obtaining an improved solution for Eq. (7) from the current one is upper bounded by $T_m = T_0 + \sum_{i=1}^{m} \frac{1}{p_i^-} + \sum_{j=0}^{m-2} g_j \leq O(nm^{m/2-1})$.

Note that for any weight vector, the function value of any objective contains at most $\frac{2n}{m}$ members, since there are $\frac{2n}{m}$ bits in any component. Thus, using TCH approach, the expect runtime of Algorithm 1 finding an optimal solution for any subproblem is $O(nm^{m/2-1}) \cdot O(\frac{n}{m}) = O(n^2 m^{m/2-2})$. □

Combining Lemma 4 and Lemma 5, we have Theorem 3.

Theorem 3. *For mLOTZ, using TCH decomposition approach, the lower bound and upper bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem are $\Omega(\frac{n^2}{m})$ and $O(n^2 m^{m/2-2})$, respectively.*

Lemma 6. *For mCOCZ, using TCH approach, the lower bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem is $\Omega(n \log n)$.*

For mCOCZ, the reference point is $z^* = (\frac{n}{2} + \frac{n}{m}, \ldots, \frac{n}{2} + \frac{n}{m})$. According to Eq. (3), the scalar subproblem produced by TCH for weight vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ is

$$\min g^{tch}(x|\boldsymbol{\lambda}, z^*) = \max_{1 \leq i \leq m} \{\lambda_i\left(\frac{n}{2} + \frac{n}{m} - f_i(x)\right)\}, \quad (9)$$

where $f_k(x)$ is shown in Definition 2. The main idea of proving this lemma is the same as that of Lemma 2, since Eq. (6) and Eq. (9) have the same property in the cooperative part.

Lemma 7. *For mCOCZ, using TCH approach, the upper bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem is $O(\frac{n^{m/2+1}}{m})$.*

Proof. First, similar to the proof of Lemma 3, we know that Algorithm 1 transforms the first $\frac{n}{2}$ bits into 1-bits in expected runtime $O(n \log n)$. We next consider the expected runtime to handle the conflicting part after this phase. We claim that for Eq. (9), $|I_{max}(x)| \leq \frac{m}{2}$ holds for any non-optimal solution x. Recall that there are $\frac{m}{2}$ components in the conflicting

part. If $|I_{max}(x)| > \frac{m}{2}$, there are at least two elements in the same component, i.e., existing a $j \in \{1, 2, \ldots, \frac{m}{2}\}$ such that $\lambda_{2j-1}\big(\frac{n}{2} + \frac{n}{m} - f_{2j-1}(x)\big) = \lambda_j\big(\frac{n}{2} + \frac{n}{m} - f_j(x)\big)$. Observe that $f_{2j-1}(x)$ and $f_{2j}(x)$ are conflicting. Thus, $f_{2j-1}(x)$ will be decreased when $f_{2j}(x)$ increasing and vice verse. Hence, the maximum of them cannot be decreased by flipping any bits in x. Therefore, the function value of Eq. (9) is at least $\lambda_{2j-1}\big(\frac{n}{2} + \frac{n}{m} - f_{2j-1}(x)\big)$ and x is an optimal solution.

Suppose that $|I_{max}(x)| = h$ holds for the current solution x. Let vector v_d denote the differences between 1-bits in all conflicting component of x and the optimal solution, and let i be the minimum value in v_d. Note that all values in v_d are integers. This means that for each component (or element) in $I_{max}(x)$, there are at least i 1-bits or 0-bits which flipping one of them can decrease the number of elements in $I_{max}(x)$ by one. And for any component not in $I_{max}(x)$, there are at most $(\frac{n}{m} - i)$ 1-bits or 0-bits which flipping one of them can increase the number of elements in $I_{max}(x)$ by one. Thus, in a generation, the probabilities of Algorithm 1 creating and accepting an offspring x' from x such that $|I_{max}(x')|$ is $h-1$ or $h+1$ are $\frac{h \cdot i}{n} \geq \frac{h}{n}$ and $\frac{(m/2-h)\cdot(n/m-i)}{n} \leq \frac{m-2h}{2m}$, respectively. Thus, the transition probability matrix $\mathbf{p}_{\frac{m}{2}+1,\frac{m}{2}+1}$ of finding an improved solution for the subproblem is

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{n} & p_{1,1} & \frac{m-2}{2m} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \frac{m/2-1}{n} & p_{m/2-1,m/2-1} & \frac{1}{m} \\ 0 & 0 & 0 & \cdots & \frac{m}{2n} & 1-\frac{m}{2n} \end{pmatrix},$$

where $p_{i,i} = 1 - p_i^+ - p_i^-$ for $i = 1, 2, \ldots, \frac{m}{2} - 1$.

By using the same argument in the proof of Lemma 5, we have $g_0 = \sum_{j=0}^{0} \frac{1}{p^-_{m/2+j}} \prod_{h=0}^{j} \frac{p^+_{m/2-1+h}}{p^-_{m/2-1+h}} = \frac{2n^2}{m^2(m/2-1)}$, and $g_{l+1} = \frac{p^+_{m/2-1-(l+1)}}{p^-_{m/2-1-(l+1)}}\big(\frac{1}{p^-_{m/2-(l+1)}} + g_l\big)$. Thus, we have $T_{m/2} = \sum_{i=1}^{m/2} \frac{1}{p_i^-} + \sum_{j=0}^{m/2-2} g_j \leq O(n^{m/2})$.

Note that for mCOCZ the number of 0-bits or 1-bits in any conflicting part is at most $\frac{n}{m}$. Thus, the expected runtime of Algorithm 1 finding an optimal solution for Eq. (9) is bounded by $O(n^{m/2}) \cdot O(\frac{n}{m}) = O(\frac{n^{m/2+1}}{m})$. □

Combining Lemma 6 with Lemma 7, we have Theorem 4.

Theorem 4. *For mCOCZ, using TCH decomposition approach, the lower bound and upper bound of expected runtime of Algorithm 1 finding an optimal solution for any subproblem are $\Omega(n \log n)$ and $O(\frac{n^{m/2+1}}{m})$, respectively.*

From the proofs of Lemmas 5 and 7, we know that the exponential expected upper bounds of using TCH are caused by the fact that it introduces many plateaus of size $\Theta(m)$ into the function landscape of some subproblems, and such a situation is likely to be encountered in handling other MaOPs.

3.3 Analysis on PBI

Theorem 5. *For mLOTZ and mCOCZ, Algorithm 1 using PBI finds an optimal solution for any subproblem in expected runtime $\Omega(\max\{\frac{n^2}{m}, n \log n\})$ and $\Omega(n \log n)$, respectively.*

Proof. First, if the penalty parameter θ is set to 0, Eq. (4) (subproblem generated by PBI for weight vector $\boldsymbol{\lambda}$) becomes

$$\min g^{pbi}(x|\boldsymbol{\lambda}, z^*) = \frac{\sum_{k=1}^{m} \lambda_k z_k^* - \sum_{k=1}^{m} \lambda_k f_k(x)}{\|\boldsymbol{\lambda}\|}. \quad (10)$$

Note that for any MOP and weight vector $\boldsymbol{\lambda}$, $\sum_{k=1}^{m} \lambda_k z_k^*$ and $\|\boldsymbol{\lambda}\|$ are fixed. Thus, the optimization of Eq. (10) is equivalent to finding $x^* = \arg\max \sum_{k=1}^{m} \lambda_k f_k(x)$, which equals Eq. (2) (subproblem generated by WS). Second, with increasing parameter θ, the penalty of individual deviating from weight vector $\boldsymbol{\lambda}$, namely the emphasis on diversity, is increased. As a result, some improved offspring for the case of using WS, which deviate from $\boldsymbol{\lambda}$ by too large an amount, will not be accepted if using PBI since their fitness values become worse (see Eq. (4)) in this situation. Thus, in any generation the probability of Algorithm 1 finding an improved offspring is at most that of using WS and the theorem is proven. □

Chen *et al.* [2019] proved that for any MOP and weight vector, the optimization of subproblem generated by WS is equivalent to that of by PBI with a specific setting of θ. Unfortunately, this does not hold in reverse. In fact, if θ is set too large, Algorithm 1 will never find an optimal solution for some subproblems, because it only accepts offspring that increase the function values of multiple objectives (i.e., flipping multiple bits in the parent). This is impossible for the one-bit mutation. If Algorithm 1 applies the standard bit mutation (SBM) [Doerr and Neumann, 2019] (also called bit-wise mutation [Zhou *et al.*, 2019]), which often flips each bit in the parent with independent probability $\frac{1}{n}$, it can flip multiple bits in a mutation. But the convergence speed is rather slow since SBM flips j bits in a mutation with probability $O(\frac{1}{n^j})$.

In summary, Algorithm 1 using WS can find an optimal solution for any subproblem of analyzed problems in polynomial expected runtime. But such a result may be not true for the case of using TCH or PBI. Moreover, the proofs of Lemmas 1 and 3 imply that the optimal solutions of different subproblems are easily corresponding to the same solution if using WS. And the proof of Theorem 5 indicates that increasing the parameter θ in PBI approach is harmful for faster converging to the PF, but it can increase the diversity.

4 Conclusions

This paper theoretically investigates the convergence speed of using the typical WS, TCH and PBI decomposition approaches in the MOEA/D framework for solving MaOPs. The results show that using WS, the algorithm can find an optimal solution for any subproblem of analyzed problems in polynomial expected runtime. When using TCH or PBI, the algorithm needs at least exponential expected runtime for some subproblems. Moreover, our analyses discover an obvious shortcoming of using WS, that is, the optimal solutions of different subproblems are easily corresponding to the same solution. And decreasing the penalty parameter in PBI is good for faster converging to the PF, but it may lose the diversity.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (61773410) and Microsoft Research Asia.

References

[Bian *et al.*, 2018] Chao Bian, Chao Qian, and Ke Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 1405–1411, Stockholm, Sweden, 2018.

[Chen *et al.*, 2019] Lei Chen, Hai-Lin Liu, Kay Chen Tan, Yiu-Ming Cheung, and Yuping Wang. Evolutionary many-objective algorithm using decomposition-based dominance relationship. *IEEE Transactions on Cybernetics*, 49(12):4129–4139, 2019.

[Das and Dennis, 1998] I. Das and J. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Deng and Zhang, 2019] Jingda Deng and Qingfu Zhang. Approximating hypervolume and hypervolume contributions using polar coordinate. *IEEE Transactions on Evolutionary Computation*, 23(5):913–918, 2019.

[Yuan *et al.*, 2016] Yuan Yuan, Hua Xu, Bo Wang, and Xin Yao. A new dominance relation-based evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 20(1):16–37, 2016.

[Rostami and Neri, 2017] Shahin Rostami and Ferrante Neri. A fast hypervolume driven selection mechanism for many-objective optimisation problems. *Swarm and Evolutionary Computation*, 34:50–67, 2017.

[Doerr and Neumann, 2019] Benjamin Doerr and Frank Neumann. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, 2019.

[Gómez and Coello, 2017] Raquel Hernández Gómez and Carlos A. Coello Coello. A hyper-heuristic of scalarizing functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 577–584, New York, NY, USA, 2017.

[Huang and Zhou, 2020] Zhengxin Huang and Yuren Zhou. Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA*, pages 2359–2366, 2020.

[Huang *et al.*, 2020] Zhengxin Huang, Yuren Zhou, Zefeng Chen, Xiaoyu He, Xinsheng Lai, and Xiaoyun Xia. Running time analysis of MOEA/D on pseudo-boolean functions. *IEEE Transactions on Cybernetics*, in press, 2020.

[Ishibuchi and Murata, 1998] Hisao Ishibuchi and Tadahiko Murata. 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, 1998.

[Jansen and Zarges, 2011] Thomas Jansen and Christine Zarges. Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theoretical Computer Science*, 412(6):517–533, 2011.

[Jansen, 2013] Thomas Jansen. *Analyzing evolutionary algorithms: The computer science perspective*. Springer, 2013.

[Laumanns *et al.*, 2004] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2):170–182, 2004.

[Li *et al.*, 2015a] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48(1):13:1–13:35, 2015.

[Li *et al.*, 2015b] Miqing Li, Shengxiang Yang, and Xiaohui Liu. Bi-goal evolution for many-objective optimization problems. *Artificial Intelligence*, 228:45–65, 2015.

[Li *et al.*, 2016] Yuan-Long Li, Yu-Ren Zhou, Zhi-Hui Zhan, and Jun Zhang. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20(4):563–576, 2016.

[Luque *et al.*, 2020] Mariano Luque, Sandra Gonzalez-Gallardo, Rubén Saborido, and Ana B Ruiz. Adaptive global WASF-GA to handle many-objective optimization problems. *Swarm and Evolutionary Computation*, 54:100644, 2020.

[Qian *et al.*, 2019] Chao Qian, Yang Yu, Ke Tang, Xin Yao, and Zhi-Hua Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence*, 275:279 – 294, 2019.

[Trivedi *et al.*, 2017] Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2017.

[Wang *et al.*, 2020] Jia Wang, Yuchao Su, Qiuzhen Lin, Lijia Ma, Dunwei Gong, Jianqiang Li, and Zhong Ming. A survey of decomposition approaches in multiobjective evolutionary algorithms. *Neurocomputing*, 408:308–330, 2020.

[Zhang and Li, 2007] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[Zhou *et al.*, 2009] Yuren Zhou, Jun He, and Qing Nie. A comparative runtime analysis of heuristic algorithms for satisfiability problems. *Artificial intelligence*, 173(2):240–257, 2009.

[Zhou *et al.*, 2019] Zhihua Zhou, Yang Yu, and Chao Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.