Runtime Analysis of Somatic Contiguous Hypermutation Operators in MOEA/D Framework

Zhengxin Huang,[1,3] **Yuren Zhou**[1,2,*]
[1]School of Data and Computer Scinece, Sun Yat-sen University, Guangzhou 510006, China
[2]Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, Sun Yat-sen University, Guangzhou 510006, China
[3]Department of Computer Science and Information Technology, Youjiang Medical University for Nationalities, Baise 533000, China
Email: thenewyi@gmail.com, zhouyuren@mail.sysu.edu.cn

Abstract

Somatic contiguous hypermutation (CHM) operators are important variation operators in artificial immune systems. The few existing theoretical studies are only concerned with understanding the optimization behavior of CHM operators on solving single-objective optimization problems. The MOEA/D framework is one of the most popular strategies for solving multi-objective optimization problems (MOPs). In this paper, we present a runtime analysis of using two CHM operators in MOEA/D framework for solving five benchmark MOPs, including four bi-objective and one many-objective problems. Our analyses show that the expected runtimes of CHM operators on the four bi-objective problems are better than or as good as that of the well-studied standard bit mutation operator. Moreover, using CHM operators in MOEA/D framework can improve the best known upper bound on the many-objective problem by a factor of n. This paper provides insight into understanding the optimization behavior of CHM operators in the well-known MOEA/D framework, and indicates that using the CHM operator in MOEA/D framework is a promising method for handling MOPs.

Introduction

Evolutionary algorithms (EAs) are a class of randomized search heuristics inspired by natural evolution. Among EAs, artificial immune systems (AISs) are inspired by the principles of immune systems of vertebrates (Corus, Oliveto, and Yazdani 2019). A main distinguishing feature of AISs to other EAs is that AISs use hypermutation operators. Different from the standard bit mutation (SBM, only flipping one bit in expectation) operator in classical EAs, hypermutation operators usually flip multiple bits in a mutation step.

AISs with hypermutation operators have achieved great success in solving complex optimization problems, including single-objective and multi-objective optimization, for example, (Shang et al. 2011; Yao et al. 2016; Alizadeh, Meskin, and Khorasani 2017; Dudek 2017; Lin et al. 2018; Xu et al. 2018; Geng et al. 2019). Compared to successes in application, the theoretical analysis on understanding the optimization behavior of hypermutation operators is underdeveloped, particularly on optimizing multi-objective optimization problems (MOPs). Somatic contiguous hypermutation (CHM) operators are important variation operators in AISs. Runtime analysis is an essential and powerful theoretical tool to understand the working principles of EAs (Doerr and Doerr 2018; Qian et al. 2019). A few theoretical studies have presented runtime analyses of EAs with CHM operators on solving single-objective optimization problems.

Jansen and Zarges (2011) analyzed the expected runtimes of a simple algorithm framework using three CHM operators on pseudo-Boolean functions. The results show that all considered CHM operators perform much better than SBM on the $\text{CLOB}_{b,k}$ problem. Jansen, Oliveto, and Zarges (2011) showed that the expected runtimes of a simple AIS with CHM operator are lower than that of SBM-based EAs on vertex cover instances that are known to be hard for randomized search heuristics. Corus et al. (2017) analyzed the expected runtimes of CHM and SBM on their easiest function classes, i.e., MINBLOCKS and ONEMAX. The results show that MINBLOCKS is exponentially hard for SBM, while CHM is only worse than SBM a factor of n on ONEMAX. Xia and Zhou (2018) presented runtime analyses of a simple algorithm with CHM, SBM or local mutation (LM) on some discrete optimization problems. The results show that CHM operators can solve all considered problems efficiently while SBM and LM need exponential expected runtime.

These studies indicate that CHM operators are very competitive operators for solving single-objective optimization problems, especially on instances that are hard for EAs with SBM or LM. In addition, analyses in (Jansen, Oliveto, and Zarges 2011) show that CHM operators can be an alternative to crossover without having to control population size and diversity. However, to the best of our knowledge, the optimization behavior of CHM operators on solving MOPs has not yet been studied from theoretical analysis aspect.

Multi-objective EAs (MOEAs) are popular methods for solving MOPs. MOEA based on decomposition (MOEA/D) framework (Zhang and Li 2007) is one of the most popular strategies in MOEAs. The decomposition idea has been used in (Ishibuchi and Murata 1998), and it becomes popular after Zhang and Li presented the MOEA/D framework in (2007).

*Corresponding author: Yuren Zhou.

In the past years, there are vast studies developing MOEAs based on the MOEA/D framework, e.g., (Qi et al. 2014; Jiang and Yang 2015; Li et al. 2017; He et al. 2019). Li et al. presented a runtime analysis of a simple MOEA/D with SBM on solving four benchmark MOPs in (2016).

In this paper, we present runtime analyses of a simple MOEA based on the MOEA/D framework with two typical CHM operators on optimizing five benchmark MOPs, which include four bi-objective and one many-objective problems. The results show that the expected runtimes of the MOEA using CHM operators on the four bi-objective problems, i.e., COCZ, LPTNO, Dec-obj-MOP and Plateau-MOP, are better than or as good as that of using the well-studied SBM operator. The expected runtimes of using CHM operators on the many-objective problem, namely mLOTZ ($m = 4$), are $O(n^2 \log n)$, which is lower than the best known upper bound by a factor of n. This analysis provides insight into understanding the optimization behavior of CHM operators in the well-known MOEA/D framework. The analysis results indicate that using CHM operator in MOEA/D framework is a promising method for solving MOPs.

Preliminaries

CHM Operators

We let $[a..b] := \{y \in \mathbb{Z} | a \leq y \leq b\}$. Given a bit string $x \in \{0,1\}^n$, we denote by $x[i], i \in [0..n-1]$ the value of the i-th bit in x. Note that we denote the first bit and the last bit in x as $x[0]$ and $x[n-1]$, respectively. The two CHM operators considered in this paper are defined as follows.

Definition 1 (CHM1 (Jansen and Zarges 2009; 2011; Xia and Zhou 2018))**.** *Given a parameter $r \in [0,1]$, the CHM1 operator mutates $x \in \{0,1\}^n$ in the following way.*

1: *Choose $p_1 \in [0..n-1]$ uniformly at random.*
2: *Choose $p_2 \in [0..n-1]$ uniformly at random.*
3: *For $i := \min\{p_1, p_2\}$ to $\max\{p_1, p_2\}$ do*
4: *Flip $x[i]$ with probability r.*

Definition 2 (CHM2 (Jansen and Zarges 2011; Xia and Zhou 2018))**.** *Given a parameter $r \in [0,1]$, the CHM2 operator mutates $x \in \{0,1\}^n$ in the following way.*

1: *Choose $p \in [0..n-1]$ uniformly at random.*
2: *Choose $l \in [0..n]$ uniformly at random.*
3: *For $i := 0$ to $l-1$ do*
4: *Flip $x[(p+i) \mod n]$ with probability r.*

As in the previous runtime analyses on CHM1 and CHM2 operators (Jansen and Zarges 2011; 2014; Xia and Zhou 2018), we only consider the case of $r = 1$ in this analysis.

Analyzed Algorithm

An MOP can be formally defined as follows:

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

where x is the decision variable, X is the decision space, $F : X \to R^m$ consists of m functions and R^m is the objective space. Generally, we call an MOP as many-objective optimization problem (MaOP) if $m \geq 4$. For $x', x'' \in X$, we say that x' weakly dominates x'', denoted as $x' \succeq x''$, if $f_i(x') \geq f_i(x'')$ for all $i \in [1..m]$. Furthermore, we say that x' dominates x'', denoted as $x' \succ x''$, if $x' \succeq x''$ and $F(x') \neq F(x'')$. A solution $x^* \in X$ is Pareto optimal if there is no $x \in X$ such that $x \succ x^*$. The set of all Pareto optimal solutions is called the *Pareto set* (PS) and the set of objective vectors of the PS is called the *Pareto front* (PF).

Algorithm 1 A simple decomposition-based MOEA

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

1: *Initialization:* The Pareto optimal solution set $P = \emptyset$. For each subproblem $g(x|\lambda^k), k \in [1..N]$, select the T closest subproblems to form its neighbor set B_k according to the Euclidean distance between their weight vectors. Generate a solution $x_k \in \{0,1\}^n$ uniformly at random for each subproblem $g(x|\lambda^k)$. Set the reference point $z = (z_1, \cdots, z_m)$, where $z_i = \max_{k \in [1..N]}\{f_i(x_k)\}$. Let S_k denote the set of solutions corresponding to subproblems in B_k.
2: **while** stop criterion is not satisfied **do**
3: **for** each subproblem $g(x|\lambda^k), k \in [1..N]$ **do**
4: *Reproduction*: create $y_k := mutate(x_k)$.
5: *Update z:* for each $i \in [1..m]$, if $f_i(y_k) > z_i$, set $z_i = f_i(y_k)$.
6: *Update S_k:* for each solution x_j in S_k, if $g(y_k|\lambda^j) \leq g(x_j|\lambda^j)$, replace x_j with y_k.
7: *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.
8: **end for**
9: **end while**

In MOEA/D framework (Zhang and Li 2007), an MOP is first decomposed into N scalar optimization subproblems according to the decomposition approach (e.g., the classical Weighted sum, Tchebycheff or Penalty-based boundary intersection (Zhang and Li 2007)) and weight vectors. Then, for each subproblem the framework selects the T closest subproblems to form its neighbor set. The distance is measured by the Euclidean distance between their weight vectors. At last, the framework controls a population of size N to cooperatively solve the N subproblems in parallel by using the neighborhood-based coevolution.

The analyzed algorithm based on the MOEA/D framework in this paper is summarized in Algorithm 1. It is not difficult to see that Algorithm 1 can be instantiated by using different mutation operators in line 4. If the SBM operator is used, Algorithm 1 is equivalent to the analyzed algorithm in (Li et al. 2016). In this paper, we are mainly concerned with analyzing the expected runtime of Algorithm 1 with CHM1 or CHM2 operator on optimizing five benchmark MOPs.

Similar to runtime analysis in (Li et al. 2016), Tchebycheff approach is used in this paper. Given an MOP in Eq. (1) and weight vector $\lambda = (\lambda_1, \cdots, \lambda_m)$, the subproblem

generated by Tchebycheff approach is

$$\min g(x|\lambda) = \max_{1\le i\le m}\{\lambda_i|f_i(x) - z_i^*|\}, \quad (2)$$

where $\lambda_i \geq 0$ for $i \in [1..m]$ and $\sum_{i=1}^{m}\lambda_i = 1$, and $z^* = (z_1^*, \cdots, z_m^*)$ denotes the reference point, i.e., $z_i^* = \max\{f_i(x)|x \in X\}$. By altering the weight vector, the Tchebycheff approach generates different subproblems in form of Eq. (2). For Algorithm 1, we use the classical Das and Dennis's approach (1998) to generate the N evenly distributed weight vectors for all considered problems, i.e., taking $\lambda_{i\in[1..m]}^{k\in[1..N]}$ from $\{\frac{0}{H}, \frac{1}{H}, \cdots, \frac{H}{H}\}$ such that $\sum_{i=1}^{m}\lambda_i^k = 1$, where H is a positive integer and $N = \binom{H+m-1}{m-1}$.

Analysis on Bi-objective Problems

In this section, we present a runtime analysis of Algorithm 1 with CHM1 or CHM2 on COCZ, LPTNO, Dec-obj-MOP and Plateau-MOP. They have been widely used in theoretical analyses of MOEAs, e.g., (Laumanns, Thiele, and Zitzler 2004; Qian, Yu, and Zhou 2013; Li et al. 2016; Bian, Qian, and Tang 2018; Huang et al. 2019). Let $|x|_1$ denote the number of 1-bits in solution x. The four analyzed problems in this section can be defined as follows.

Definition 3 (COCZ (Li et al. 2016))**.** *The pseudo-Boolean function $COCZ : \{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

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

For COCZ, it is to maximize the numbers of 1-bits and 0-bits in x simultaneously. A similar definition of this problem called ONEMINMAX is presented in (Giel and Lehre 2010).

Definition 4 (LPTNO (Li et al. 2016))**.** *The pseudo-Boolean function $LPTNO : \{-1,1\}^n \to \mathbb{R}^2$ is defined as follows:*

$$LPTNO(x) = \big(\sum_{i=1}^{n}\prod_{j=1}^{i}(1+x_j), \sum_{i=1}^{n}\prod_{j=i}^{n}(1-x_j)\big).$$

Weight LPTNO is first defined in (Qian, Yu, and Zhou 2013). It is an extension of LOTZ (where the numbers of leading 1-bits and trailing 0-bits are maximized simultaneously) obtained by shifting the decision space from $\{0,1\}^n$ to $\{-1,1\}^n$, and adding weights w_i and v_i for the i-th leading 1-bit and trailing (-1)-bit, respectively. Li et al. (2016) considered the case of $w_i = v_i = 1$ and denoted as LPTNO.

Definition 5 (Dec-obj-MOP)**.** *The pseudo-Boolean function $Dec\text{–}obj\text{–}MOP : \{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

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

where

$$f_1(x) = n + 1 - |x|_1 \mod (n+1), \quad (3)$$
$$f_2(x) = n + |x|_1 \mod (n+1). \quad (4)$$

Dec-obj-MOP is first defined in (Li et al. 2016). For $f_2(x)$, there is deceptive property in the search space. Specifically, the global optimum 0^n is far from the local optimum 1^n, and the search is guided to 1^n with overwhelming probability. Thus, these kinds of functions are hard to solve for algorithms with SBM. The expected runtime of Algorithm 1 with SBM finding the optimal solution for subproblem corresponding to $\lambda = (0,1)$ is $\Omega(n^n)$ (Li et al. 2016).

Definition 6 (Plateau-MOP (Li et al. 2016))**.** *The pseudo-Boolean function $Plateau\text{–}MOP : \{0,1\}^n \to \mathbb{N}^2$ is defined as follows:*

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

where

$$f_1(x) = \begin{cases} n & \text{if } x = 1^n, \\ 3n/4 & \text{if } x = 1^j0^{n-j}, 3n/4 < j < n, \\ j & \text{if } x = 1^j0^{n-j}, 0 \le j \le 3n/4, \\ -|x|_1 & otherwise, \end{cases} \quad (5)$$

$$f_2(x) = \begin{cases} n - j & \text{if } x = 1^j0^{n-j}, 0 \le j \le n, \\ -|x|_1 & otherwise. \end{cases} \quad (6)$$

For $f_1(x)$ of Plateau-MOP, there is a plateau in the search space, where x is in the form of $1^i0^{n-i}, i \in [\frac{3n}{4}..n-1]$. For convenience, we hereafter assume that $\frac{n}{4}$ is integral. Thus, solutions in this range are dominated by the solution $1^{3n/4}0^{n/4}$ since the fitness value on $f_2(x)$ becomes worse when increasing $|x|_1$. Before we present the main results, we first prove two helper lemmas for our analyses.

Lemma 1. *Given a solution $x \in \{0,1\}^n$, let p' and p'' be two positions of x satisfying $p' \le p''$. The probability of CHM1 or CHM2 only flipping all bits in the contiguous region from p' to p'' in x in an execution is $\Theta(\frac{1}{n^2})$.*

Proof. From Definition 1, we know that in an execution CHM1 only flips all bits in the contiguous region from p' to p'' in x if and only if $\{p_1 = p', p_2 = p''\}$ or $\{p_2 = p', p_1 = p''\}$. Note that we have assumed $r = 1$. Since p_1 and p_2 are selected from $[0..n-1]$, the total number of combinations is n^2. Thus, if $p' \neq p''$ the probability is $\frac{2}{n^2}$ otherwise $\frac{1}{n^2}$. For CHM2, in an execution it only flips all bits between p' and p'' in x if and only if $\{p = p', l = p'' - p' + 1\}$. Thus, the probability is $\frac{1}{n(n+1)}$ since the total number of combinations is $n(n+1)$. Therefore, the probability of CHM1 or CHM2 only flipping all bits in the contiguous region from p' to p'' in x in an execution is $\Theta(\frac{1}{n^2})$. □

Let 1^i0^{n-i} denote the bit string with i leading 1-bits and $(n-i)$ trailing 0-bits and $S := \{1^i0^{n-i}, i \in [0..n]\}$.

Lemma 2. *For $x_1, x_2 \in S$ satisfying $x_1 \neq x_2$, the probability of CHM1 or CHM2 creating x_2 from x_1 in an execution is $\Theta(\frac{1}{n^2})$.*

Proof. Let d_1 and d_2 denote the first and the last of bits that have different values between x_1 and x_2, respectively. By Lemma 1, in an execution CHM1 or CHM2 creates x_2 from x_1, i.e., only flipping all bits in the contiguous region from d_1 to d_2 in x_1, with probability $\Theta(\frac{1}{n^2})$. □

In the following, we present the runtime analysis of Algorithm 1 with CHM1 or CHM2 on the four problems (using the evenly distributed weight vectors for all problems). For each problem, our analysis consists of two phases. In the first phase, we estimate the expected runtime until a solution in the form of 1^i0^{n-i} is found for some subproblems for the

first time. In the second phase, we bound the expected runtime of obtaining a set of solutions to cover the whole PF of the problem after the first phase according to Lemma 2.

Similar to (Li et al. 2016), we assume that the optimal values of all objectives, i.e., the reference points, of all problems have been known in our analysis. Recall that for $k \in [1..N]$, $\lambda^k_{i\in[1..m]} \in \{0, 1/H, 2/H, \cdots, 1\}$ and $\sum_{i=1}^{m} \lambda_i^k = 1$ hold. For ease of expression, we number the elements in the set of weight vectors in ascending order according to λ_1^k, i.e., $\lambda^1 = (0, 1), \lambda^2 = (1/H, 1 - 1/H), \cdots, \lambda^{N-1} = (1 - 1/H, 1/H), \lambda^N = (1, 0)$.

For COCZ, all solutions are Pareto optimal. Thus, we only need to bound the runtime of finding a solution for each vector in the PF. Li et al. (2016) proved that using the evenly distributed weight vectors with $H = n$, Algorithm 1 with SBM can optimize the COCZ in expected runtime $O(n^2 \log n)$.

Lemma 3. *For COCZ, Algorithm 1 with CHM1 or CHM2 respectively finds the optimal solutions 0^n and 1^n for $g(x|\lambda^1)$ and $g(x|\lambda^N)$ in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. For COCZ, the reference point is (n, n). According to Eq. (2), these decomposed subproblems corresponding to weight vectors $\lambda^i, i \in [1..N]$ are as follows:

$$\min\ g(x|\lambda^i) = \max\{\lambda_1^i n - \lambda_1^i |x|_1, \lambda_2^i |x|_1\}, \qquad (7)$$

Observing the two terms in Eq. (7) when increasing $|x|_1$ from 0 to n, the value of $(\lambda_1^i n - \lambda_1^i |x|_1)$ decreases from $\lambda_1^i n$ to 0 and the value of $\lambda_2^i |x|_1$ increases from 0 to $\lambda_2^i n$. So there exists a real number $y_i \in [0, n]$ such that

$$\lambda_1^i n - \lambda_1^i y_i = \lambda_2^i y_i \Rightarrow y_i = \lambda_1^i n.$$

Thus, to get the minimal value of $g(x|\lambda^i)$ (in Eq. (7)), we should set $|x|_1$ close to y_i as much as possible. Note that there may be two consecutive integers for $|x|_1$ to obtain the minimal value of $g(x|\lambda^i)$, since $|x|_1$ is the number of 1-bits in solution x and y_i may be a real number.

For ease of expression, we assume that $a = \lambda_1^i n$ is an integer. This implies that the number of 1-bits in the optimal solution for $g(x|\lambda^i)$ is $\lambda_1^i n$. Since $\lambda^1 = (0, 1)$ and $\lambda^N = (1, 0)$, we have that the optimal solutions for $g(x|\lambda^1)$ and $g(x|\lambda^N)$ are 0^n and 1^n, respectively. Note that they are both in the form of $1^j 0^{n-j}$. We next estimate the expected runtime of Algorithm 1 with CHM1 or CHM2 finding the optimal solutions for $g(x|\lambda^1)$ and $g(x|\lambda^N)$.

For $g(x|\lambda^1)$, the optimal solution is 0^n and the function fitness value becomes better when the number of 1-bits in the solution decreases. Let t denote the number of 1-bits in the current solution. In the generation, if CHM1 or CHM2 only flips one of the $(n-t)$ 1-bits in the current solution into 0-bit, then an improved solution is created. From Lemma 1, we know that in a generation CHM1 or CHM2 only flips any bit in the solution with probability $\Theta(\frac{1}{n^2})$. Thus, the probability of Algorithm 1 with CHM1 or CHM2 creating an improved solution for $g(x|\lambda^1)$ from the current one is at least $\binom{n-t}{1} \cdot \Theta(\frac{1}{n^2}) = \Theta(\frac{n-t}{n^2})$. Therefore, the expected runtime of Algorithm 1 with CHM1 or CHM2 finding the optimal solution 0^n for $g(x|\lambda^1)$ is upper bounded by

$$N \sum_{t=0}^{n-1} \frac{n^2}{n-t} = N \cdot n^2 \sum_{t=1}^{n} \frac{1}{t} = O(N \cdot n^2 \log n).$$

Note that in a generation Algorithm 1 creates a new solution for each subproblem. Similarly, we have that Algorithm 1 with CHM1 or CHM2 finds the optimal solution 1^n for $g(x|\lambda^N)$ in expected runtime $O(N \cdot n^2 \log n)$.

In summary, Algorithm 1 with CHM1 or CHM2 finds the optimal solutions 0^n and 1^n for subproblems $g(x|\lambda^1)$ and $g(x|\lambda^N)$ in expected runtime $O(N \cdot n^2 \log n)$. □

We next bound the expected fitness evaluations until all solutions in $S = \{1^i 0^{n-i}, i \in [0..n]\}$ are created after 0^n and 1^n have been found for $g(x|\lambda^1)$ and $g(x|\lambda^N)$, respectively. Note that the objective vectors of S can cover the PF of COCZ and these vectors will be retained by the algorithm after they have been found (see line 7 in Algorithm 1).

Lemma 4. *For COCZ, after the optimal solutions 0^n and 1^n have been respectively found for $g(x|\lambda^1)$ and $g(x|\lambda^N)$, Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the PF in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. Note that the optimal solutions 0^n and 1^n will be respectively kept by $g(x|\lambda^1)$ and $g(x|\lambda^N)$ forever after they have been found for the first time. So they are always undergoing the variation operator in the later generations.

From Lemma 2, we know that by applying CHM1 or CHM2 to 0^n, any solution in S is created in the new generation with probability $\Theta(\frac{1}{n^2})$. Let j denote the number of solutions in S that have not been created up to the current generation. Thus, by applying CHM1 or CHM2 to 0^n, one of the j solutions is created in the new generation with probability $\binom{j}{1} \cdot \Theta(\frac{1}{n^2})$. Similarly, we have that a new solution in S is created in the new generation with probability $\Theta(\frac{j}{n^2})$ by applying CHM1 or CHM2 to 1^n. Hence, in the new generation a new solution in S is created with probability $\Omega(\frac{j}{n^2})$. Therefore, after 0^n and 1^n have been respectively found for $g(x|\lambda^1)$ and $g(x|\lambda^N)$, the expected runtime of Algorithm 1 with CHM1 or CHM2 obtaining a set of solutions to cover the whole PF of COCZ is upper bounded by

$$N \sum_{j=1}^{n-2} \frac{n^2}{j} \le N \cdot n^2 \sum_{j=1}^{n} \frac{1}{j} = O(N \cdot n^2 \log n).$$

Note that Algorithm 1 consumes N fitness evaluations in any generation. □

By Lemmas 3 and 4, we have the following theorem.

Theorem 1. *For COCZ, if $N = O(1)$, Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the PF in expected runtime $O(n^2 \log n)$.*

For LPTNO, the PS is $\{1^n, 1^{n-1}(-1), \cdots, (-1)^n\}$. Li et al. (2016) proved that using the weight vectors $\{(\lambda_1^k, 1 - \lambda_1^k) | \lambda_1^k = \frac{\sum_{j=1}^{k} 2^{n+1-j}}{\sum_{j=k+1}^{n} 2^j + \sum_{j=1}^{k} 2^{n+1-j}}, k \in [0..n]\}$, Algorithm 1 with SBM obtains a set of solutions to cover the PF in expected runtime $O(n^3)$. Note that Das and Dennis's approach can only generate these weight vectors in the case of $H = \Omega(2^n)$ (see Lemma 3 in (Li et al. 2016)).

Lemma 5. *For LPTNO, Algorithm 1 with CHM1 or CHM2 finds an optimal solution, which is in the form of $1^j(-1)^{n-j}$, for each subproblem in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. For LPTNO, the reference point is $(2^{n+1}-2, 2^{n+1}-2)$. According to Eq. (2), these subproblems corresponding to weight vectors $\lambda^i, i \in [1..N]$ are

$$\min g(x|\lambda^i) = \max\{\lambda_1^i(b-2^{l(x)+1}), \lambda_2^i(b-2^{t(x)+1})\}, \quad (8)$$

where $b = 2^{n+1}$, and $l(x)$ and $t(x)$ are the numbers of leading 1-bits and trailing (-1)-bits in x, respectively. Observe that the value of $\lambda_1^i(b-2^{l(x)+1})$ decreases when increasing $l(x)$, and the value of $\lambda_2^i(b-2^{t(x)+1})$ also decreases when increasing $t(x)$. Thus, to obtain the minimal value of $g(x|\lambda^i)$ (in Eq. (8)), we should simultaneously increase the values of $l(x)$ and $t(x)$ as much as possible. If $l(x)+t(x) \leq n-2$, we can increase each of them by at least one, and the values of the two terms in the right hand of $g(x|\lambda^i)$ are decreased simultaneously. Then a better function value is found for $g(x|\lambda^i)$ and x cannot be an optimal solution. Hence, we have that $l(x)+t(x) = n$ holds for any optimal solution x of $g(x|\lambda^i)$. Note that for any solution x, $l(x)+t(x) \neq n-1$ hold, since the rest bit between the leading 1-bits and trailing (-1)-bits must be 1 or -1. This implies that for any $g(x|\lambda^i)$, the optimal solutions are in the form of $1^j(\text{-}1)^{n-j}$.

We now determine the exact value of $l(x)$ and $t(x)$ in the optimal solution of each $g(x|\lambda^i)$. Since $l(x)+t(x) = n$ holds for any optimal solution x, by Eq. (8) we have

$$\min g(x|\lambda^i) = \max\{\lambda_1^i(b-2^{l(x)+1}), \lambda_2^i(b-2^{n-l(x)+1})\}. \quad (9)$$

Observe the trend of two terms in Eq. (9) when $l(x)$ increases from 0 to n. The value of the first term decreases from $\lambda_1^i(b-2)$ to 0 and the second term increases from 0 to $\lambda_2^i(b-2)$. So there exists a real number $y_i \in [0,n]$ satisfying

$$\lambda_1^i(b-2^{y_i+1}) - \lambda_2^i(b-2^{n-y_i+1}) = 0 \Rightarrow$$

$$y_i = \begin{cases} n-1+\log(\sqrt{(a-1)^2+\frac{a}{2^{n-2}}}-(a-1)), \text{if } \lambda_1^i \neq 0, \\ 0, \text{otherwise}, \end{cases}$$

where $a = \frac{\lambda_2^i}{\lambda_1^i}$. To get the optimal value of $g(x|\lambda^i)$, we should set $l(x)$ close to y_i as much as possible. For convenience, we again assume that y_i is an integer. This means that the optimal solution of $g(x|\lambda^i)$ is $1^{y_i}(\text{-}1)^{n-y_i}$. We next estimate the expected runtime of Algorithm 1 with CHM1 or CHM2 finding the optimal solution for each $g(x|\lambda^i)$.

We use the shorthands $f_1(x|\lambda^i)$ and $f_2(x|\lambda^i)$ for the values of the first term and the second term in $g(x|\lambda^i)$ (see Eq. (8)), respectively. Let $x_0 = 1^{n_l}(\text{-}1)\#1(\text{-}1)^{n_t}$ denote the current solution for $g(x|\lambda^i)$, where $\#$ denotes a wildcard string consisting of 1-bits and (-1)-bits. Without loss of generality, we assume that $n_l < y_i$, $n_t < n-y_i$ and $f_1(x_0|\lambda^i) = f_2(x_0|\lambda^i)$ hold. From Eq. (8), we know that for $g(x|\lambda^i)$ solutions $x_1 = 1^{n_l+1}\#1(\text{-}1)^{n_t}$ and $x_2 = 1^{n_l}(\text{-}1)\#(\text{-}1)^{n_t+1}$ are not worse than x_0, and the solution $x_3 = 1^{n_l+1}\#(\text{-}1)^{n_t+1}$ is strictly better than x_0, x_1 and x_2. Thus, if x_3 has been created for $g(x|\lambda^i)$, x_0, x_1 and x_2 will never be accepted since Algorithm 1 is elitist. For CHM1, in a mutation it creates x_1 or x_2 from x_0 if $p_1 \in \{n_l, n-n_t-1\}$ and $p_2 \in [n_l..n-n_t-1]$. For CHM2, in a mutation x_1 is created from x_0 if $p = n_l$ and $l \in [1..n-n_l-n_t]$, and x_2 is created from x_0 if $p \in [n_l..n-n_t-1]$ and $l = n-n_l-n_t-p$.

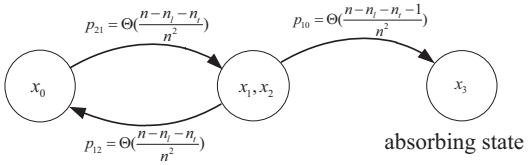

Figure 1: The Markov chain of finding an improved solution for $g(x|\lambda^i)$ from the current one.

Thus, by Lemma 1, the probability of CHM1 or CHM2 creating x_1 and x_2 from x_0 in a generation is $\Theta(\frac{n-n_l-n_t}{n^2})$. By analogy, we have that CHM1 or CHM2 creates x_3 from x_1 or x_2 in a generation with probability $\Theta(\frac{n-n_l-n_t-1}{n^2})$. Note that the probability of CHM1 or CHM2 creating x_0 from x_1 and x_2 is also $\Theta(\frac{n-n_l-n_t}{n^2})$.

Thus, in the worst case the process of Algorithm 1 with CMH1 or CHM2 finding an improved solution for $g(x|\lambda^i)$ from the current one is equivalent to reaching the absorbing state of the Markov chain shown in Figure 1. According to Corollary 2 in (Zhou, He, and Nie 2009), the expected runtime of reaching x_3 from x_0 is at most $\frac{1}{p_{21}}+\frac{1}{p_{10}}+\frac{1}{p_{21}}\cdot\frac{p_{12}}{p_{10}} \leq O(\frac{n^2}{n-n_l-n_t-1})$. Note that the expected runtime of creating x_3 from x_1 or x_2 is not larger than that of from x_0. Therefore, the expected fitness evaluations of Algorithm 1 with CMH1 or CHM2 finding the optimal solution for subproblem $g(x|\lambda^i)$ is upper bounded by

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

In summary, for LPTNO Algorithm 1 with CHM1 or CHM2 finds an optimal solution for each subproblem in expected runtime $O(N \cdot n^2 \log n)$ since it optimizes the N subproblems in parallel. □

Theorem 2. *For LPTNO, if $N = O(1)$, Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the PF in expected runtime $O(n^2 \log n)$.*

Proof. Note that the PS of LPTNO is $\{1^i(\text{-}1)^{n-i}, i \in [0..n]\}$ and all optimal solution found for all subproblems are in the form of $1^i(\text{-}1)^{n-i}$. Let j denote the number of solutions in the PS that have been not found so far. By using similar argument in the proof of Lemma 4, we have that in the new generation a new solution in S is created with probability $\Omega(\frac{j}{n^2})$. Thus, after the optimal solution has been found for each subproblem, the expected runtime of Algorithm 1 with CHM1 or CHM2 obtaining a set of solutions to cover the whole PF of LPTNO is upper bounded by

$$N \sum_{j=1}^{n-2} \frac{n^2}{j} \leq N \cdot n^2 \sum_{j=1}^{n} \frac{1}{j} = O(N \cdot n^2 \log n).$$

Therefore, combining Lemma 5, we have that Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the whole PF of LPTNO in expected runtime $O(n^2 \log n)$ if $N = O(1)$ (i.e., $H = O(1)$). □

The PS and PF of Dec-obj-MOP are the same as the COCZ. Li et al. (2016) proved that using the evenly distributed weight vectors with $H = n$, Algorithm 1 with SBM can find a set of solutions to cover the PF in expected runtime $O(n^2 \log n)$.

Lemma 6. *For Dec-obj-MOP, Algorithm 1 with CHM1 or CHM2 finds the optimal solution 0^n for subproblem $g(x|\lambda^1)$ in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. For Dec-obj-MOP, the reference point is (n, n). According to Eq. (2), these decomposed subproblems corresponding to weight vectors $\lambda^i, i \in [1..N]$ are

$$\min\ g(x|\lambda^i) = \max\{\lambda_1^i f_2(x), \lambda_2^i n - \lambda_2^i f_2(x)\}, \quad (10)$$

where $f_2(x)$ is shown in Eq. (4). Observe the trend of the two terms in the right hand of Eq. (10) when $f_2(x)$ increases from 0 to n. The value of $\lambda_1^i f_2(x)$ increases from 0 to $\lambda_2^i n \geq 0$ and the value of $(\lambda_2^i n - \lambda_2^i f_2(x))$ decreases from $\lambda_2^i n \geq 0$ to 0. So there must exist a real number $y_i \in [0, n]$ such that

$$\lambda_2^i n - \lambda_2^i y_i = \lambda_1^i y_i \Rightarrow y_i = \lambda_2^i n.$$

To get the minimal value of $g(x|\lambda^i)$, we should set $f_2(x)$ close to y_i as much as possible. Since $\lambda^1 = (0, 1)$, the minimal value of $g(x|\lambda^1)$ is obtained when $f_2(x) = n$. By Eq. (4), we know that the optimal solution of $g(x|\lambda^1)$ is 0^n.

For $g(x|\lambda^1)$, there is deceptive property in the search space since the function value becomes better when increasing $|x|_1$ if $|x|_1 > 0$. Since the expected number of 1-bits in the initial solution is $\frac{n}{2} > 0$, by Chernoff bound, the search will be guided to the local optimum 1^n with probability $1 - e^{-\Omega(n)}$. If 0^n has not been found, by Lemma 1, CHM1 or CHM2 increases the number of 1-bits by one in a generation with probability at least $\Theta(\frac{n-j}{n^2})$, where j denotes the number of 1-bits in the current solution. Thus, Algorithm 1 with CHM1 or CHM2 finds the local optimum 1^n for $g(x|\lambda^1)$ in expected runtime $O(n^2 \log n)$. By Lemma 2, CHM1 or CHM2 creates 0^n from 1^n in a generation with probability $\Theta(\frac{1}{n^2})$. Therefore, Algorithm 1 with CHM1 or CHM2 finds the global optimum 0^n for $g(x|\lambda^1)$ in expected runtime $O(N \cdot n^2 \log n) + \Theta(N \cdot n^2) = O(N \cdot n^2 \log n)$. □

Lemma 7. *For Dec-obj-MOP, after the optimal solution 0^n has been found for $g(x|\lambda^1)$, Algorithm 1 with CHM1 or CHM2 finds a set of solution to cover the PF in expected runtime $O(N \cdot n^2 \log n)$.*

We omit the detailed proof of this lemma since it is very similar to the proof of Lemma 4, where the optimal solution of $g(x|\lambda^1)$ is also 0^n. Thus, combining Lemma 6 and Lemma 7, we have the following theorem.

Theorem 3. *For Dec-obj-MOP, if $N = O(1)$, Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the PF in expected runtime $O(n^2 \log n)$.*

For Plateau-MOP, the well-known GSEMO is hard to optimize and the expected runtime is $\Omega(n^{0.25n})$ (Li et al. 2016). Li et al. (2016) proved that using the evenly distributed weight vectors with $H = n$, Algorithm 1 with SBM obtains a set of solutions to cover the PF of Plateau-MOP in expected runtime $O(n^3)$.

Lemma 8. *For Plateau-MOP, Algorithm 1 with CHM1 or CHM2 finds a solution in the form of $1^j 0^{n-j}$ for each subproblem in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. For Plateau-MOP, the reference point is $z^* = (n, n)$, and these subproblems corresponding to weight vectors $\lambda^i, i \in [1..N]$ are

$$\min g(x|\lambda^i) = \max\{\lambda_1^i(n - f_1(x)), \lambda_2^i(n - f_2(x))\}, \quad (11)$$

where $f_1(x)$ and $f_2(x)$ are shown in Eq. (5) and Eq. (6), respectively. Observing the two terms in the right hand of Eq. (11), we can find that the value of the first term decreases when increasing $f_1(x)$ and the value of the second term decreases when increasing $f_2(x)$. Thus, to obtain the minimal value of $g(x|\lambda^i)$, we should simultaneously increase the values of $f_1(x)$ and $f_2(x)$ as much as possible.

From Eq. (5) and Eq. (6), we know that if the current solution is not in the form of $1^j 0^{n-j}$, any new solution reducing the number of 1-bits will increase $f_1(x)$ and $f_2(x)$ simultaneously. Thus if the current solution is not in form of $1^j 0^{n-j}$, in a generation CHM1 or CHM2 creates a better solution with probability $\Theta(\frac{t}{n^2})$, where t denotes the number of 1-bits in the current solution. Recall that in an execution CHM1 or CHM2 only flips any bit in the solution with probability $\Theta(\frac{1}{n^2})$ (see Lemma 1). Therefore, the expected runtime of Algorithm 1 with CHM1 or CHM2 finding a solution in the form of $1^j 0^{n-j}$ for each $g(x|\lambda^i)$ is upper bounded by

$$N \sum_{t=1}^{n-2} \frac{n^2}{t} = N \cdot n^2 \sum_{t=1}^{n-2} \frac{1}{t} = O(N \cdot n^2 \log n).$$

Recall that Algorithm 1 with CHM1 or CHM2 consumes N fitness evaluations in any generation. □

Note that for all λ_i, we have $g(x|\lambda^i) < g(y|\lambda^i)$ for any $x \in S, y \notin S$ by Eq. (11), where $S := \{1^i 0^{n-i}, i \in [0..n]\}$. Thus, for any subproblem, if a solution in S has been found, all solutions not in the form of $1^i 0^{n-i}$ will never be accept.

Theorem 4. *For Plateau-MOP, if $N = O(1)$, Algorithm 1 with CHM1 or CHM2 finds a set of solutions to cover the whole PF in expected runtime $O(n^2 \log n)$.*

The proof of this theorem is very similar to that of Theorem 2 since all solutions kept by every subproblems are in the form of $1^i 0^{n-i}$ after the first phase and the PS of Plateau-MOP is a subset of S.

Analysis on Many-objective Problem

In this section, we present the runtime analysis of Algorithm 1 with CHM1 or CHM2 on a many-objective problem, i.e., mLOTZ for $m = 4$ (Laumanns, Thiele, and Zitzler 2004).

Definition 7 (mLOTZ)**.** *The pseudo-Boolean function $mLOTZ : \{0, 1\}^n \to \mathbb{N}^m$ is defined as follows:*

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

where

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

$m = 2 \cdot m'$, $n = m' \cdot n'$ and $m', n' \in \mathbb{N}$.

As shown in Definition 7, mLOTZ is the concatenation of $m' = m/2$ bi-objective LOTZ problems of $n' = 2n/m$ bits each. For mLOTZ, the size of the PF is $(2n/m+1)^{m/2}$ and thus the problem becomes complicated quickly when increasing m. For ease of observation, we only consider the case of $m = 4$ in this analysis. The expected runtime of the well-known SEMO on mLOTZ is $O(n^{m+1})$, and the best known upper bound for $m = 4$ is $O(n^3 \log n)$ (Laumanns, Thiele, and Zitzler 2004). For mLOTZ, if $m = 4$ (denoted as 4LOTZ), the PS is $\{1^i 0^{n/2-i} 1^j 0^{n/2-j}\}$, where $i, j \in [0..\frac{n}{2}]$. To simplify the analysis, we only consider these solutions created for the subproblem corresponding to weight vector $\lambda^0 = (0.5, 0, 0, 0.5)$ in the following proof.

Lemma 9. *For 4LOTZ, Algorithm 1 with CHM1 or CHM2 finds the optimal solution of subproblem corresponding to weight vector λ^0 in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. For 4LOTZ, the reference point is $(\frac{n}{2}, \frac{n}{2}, \frac{n}{2}, \frac{n}{2})$ and the subproblem corresponding to weight vector λ^0 is

$$\min g(x|\lambda^0) = \max\{n/4 - 0.5f_1(x), n/4 - 0.5f_4(x)\}. \tag{12}$$

Observe that the optimization of Eq. (12) is equivalent to maximizing $f_1(x)$ and $f_4(x)$ simultaneously. From Definition 7, $f_1(x)$ and $f_4(x)$ are equal to the numbers of leading 1-bits in the first half of x and trailing 0-bits in the second half of x, respectively. Thus, the optimal solution of Eq. (12) is $1^{n/2}0^{n/2}$. Let $n_l(x)$ and $n_t(x)$ denote the number of leading 1-bits and trailing 0-bits in the current solution x, respectively. We assume that $\min\{n_l(x), n_t(x)\} < n/2$ holds, otherwise there is nothing to prove. By using similar analysis in the proof of Lemma 5, we know that the process of Algorithm 1 creating a new solution x' from x such that $\min\{n_l(x'), n_t(x')\} > \min\{n_l(x), n_t(x)\}$ is equivalent to reaching the absorbing state of the Markov chain shown in Figure 1 ($p_{21} = p_{12} = \Omega(\frac{n-2j}{n^2})$ and $p_{10} = \Omega(\frac{n-2j-1}{n^2})$). Thus, the expected fitness evaluations of Algorithm 1 with CHM1 or CHM2 finding an improved solution from the current one is $O(\frac{n^2}{n-2j-1})$, where $j = \min\{n_l(x), n_t(x)\}$. Therefore, the expected runtime of Algorithm 1 with CMH1 or CHM2 finding the optimal solution $1^{n/2}0^{n/2}$ for subproblem $g(x|\lambda^0)$ is upper bounded by

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

Note that Algorithm 1 solves the N subproblems in parallel and creates a new solution for each one in a generation. □

Note that some elements in the PS of 4LOTZ are not in the form of $1^i 0^{n-i}$, e.g., $1^2 0^{n/2-2} 1^{n/2}$. Thus, we cannot use Lemma 2 to simply bound the expected runtime of Algorithm 1 with CHM1 or CHM2 on the problem as before.

Lemma 10. *For 4LOTZ, after the optimal solution $1^{n/2}0^{n/2}$ has been found for subproblem $g(x|\lambda^0)$, Algorithm 1 with CHM1 or CHM2 obtains a set of solutions to cover the whole PF in expected runtime $O(N \cdot n^2 \log n)$.*

Proof. Note that the optimal solution $1^{n/2}0^{n/2}$ will be kept by subproblem $g(x|\lambda^0)$ forever after it has been found for the first time. For ease of observation, we list the two components of solutions in the PS of 4LOTZ as in Table 1. Observe that if CHM1 or CHM2 flips only all bits in the contiguous region from $p' = n/2 - i$ to $p'' = n/2 + j - 1$ in $1^{n/2}0^{n/2}$ (i.e., A_0B_0), the Pareto optimal solution A_iB_j is created (except A_0B_0), where $i, j \in [0..n/2]$. Thus, if all pairs $(i, j) \in [0..\frac{n}{2}] \times [0..\frac{n}{2}] \setminus \{0 \times 0\}$ are selected at least once, all solutions in the PS of 4LOTZ have been created. Note that the total number of pairs is $(\frac{n}{2}+1)^2 - 1 = \frac{n^2}{4} + n$.

Table 1: The two components of solutions in PS of 4LOTZ.

A_0	$1^{n/2}$	$0^{n/2}$	B_0
A_1	$1^{n/2-1}0^1$	$1^1 0^{n/2-1}$	B_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$A_{n/2-i}$	$1^{n/2-i}0^i$	$1^j 0^{n/2-j}$	$B_{n/2-j}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$A_{n/2-1}$	$1^1 0^{n/2-1}$	$1^{n/2-1}0^1$	$B_{n/2-1}$
$A_{n/2}$	$0^{n/2}$	$1^{n/2}$	$B_{n/2}$

For CHM1 and CHM2, by Lemma 1, in an execution only all bits in any contiguous region from p' to p'' in $1^{n/2}0^{n/2}$ are flipped with probability $\Theta(\frac{1}{n^2})$. Thus, in a generation Algorithm 1 with CHM1 or CHM2 creates a new Pareto optimal solution from $1^{n/2}0^{n/2}$ with probability $\binom{n^2/4+n-t}{1} \cdot \Theta(\frac{1}{n^2}) = \Theta(\frac{n^2/4+n-t}{n^2})$, where t denotes the number of pairs that have been selected before. Therefore, after the optimal solution $1^{n/2}0^{n/2}$ has been found for $g(x|\lambda^0)$, the expected runtime of Algorithm 1 with CHM1 or CHM2 finding all solutions in the PS is upper bounded by

$$\sum_{t=1}^{\frac{n^2}{4}+n-1} \frac{N \cdot n^2}{n^2/4 + n - t} \leq \sum_{t=1}^{\frac{n^2}{4}+n} \frac{N \cdot n^2}{t} = O(N \cdot n^2 \log n).$$

Recall that the algorithm consumes N fitness evaluations in any generation. □

By Lemmas 9 and 10, we have the following theorem.

Theorem 5. *For 4LOTZ, if $N = O(1)$, Algorithm 1 with CHM1 or CHM2 finds a set of solutions to cover the whole PF in expected runtime $O(n^2 \log n)$.*

Conclusions

The existing theoretical analyses on CHM operators are only concerned with optimizing single-objective optimization problems. This paper presents runtime analyses of an MOEA based on the well-known MOEA/D framework with two typical CHM operators on optimizing five benchmark MOPs, which include four bi-objective and one many-objective problems. The results on the four bi-objective problems, i.e., COCZ, LPTNO, Dec-obj-MOP and Plateau-MOP, show that using CHM operators can always find the Pareto fronts in expected runtime better than or as good as using the well-studied SBM operator. Moreover, the expected runtimes of using CHM operators on the many-objective problem, i.e., mLOTZ ($m = 4$), are better than

the best known bound by a factor of n. These results indicate that using CHM operator in MOEA/D framework is promising method for solving MOPs. This study provides insight into the optimization behavior of CHM operators in the well-known MOEA/D framework, and might be helpful for designing efficient MOEAs in future research.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (61773410, 61673403, Yuren Zhou).

References

Alizadeh, E.; Meskin, N.; and Khorasani, K. 2017. A negative selection immune system inspired methodology for fault diagnosis of wind turbines. *IEEE Transactions on Cybernetics* 47(11):3799–3813.

Bian, C.; Qian, C.; and Tang, K. 2018. A general approach to running time analysis of multi-objective evolutionary algorithms. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, 1405–1411. Stockholm, Sweden: AAAI Press.

Corus, D.; He, J.; Jansen, T.; Oliveto, P. S.; Sudholt, D.; and Zarges, C. 2017. On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica* 78(2):714–740.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2019. Artificial immune systems can find arbitrarily good approximations for the NP-hard number partitioning problem. *Artificial Intelligence* 274:180–196.

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

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

Dudek, G. 2017. Artificial immune system with local feature selection for short-term load forecasting. *IEEE Transactions on Evolutionary Computation* 21(1):116–130.

Geng, B.; Jiao, L.; Gong, M.; Li, L.; and Wu, Y. 2019. A two-step personalized location recommendation based on multi-objective immune algorithm. *Information Sciences* 475:161–181.

Giel, O., and Lehre, P. K. 2010. On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation* 18(3):335–356.

He, X.; Zhou, Y.; Chen, Z.; and Zhang, Q. 2019. Evolutionary many-objective optimization based on dynamical decomposition. *IEEE Transactions on Evolutionary Computation* 23(3):361–375.

Huang, Z.; Zhou, Y.; Chen, Z.; and He, X. 2019. Running time analysis of MOEA/D with crossover on discrete optimization problem. In *Proceeding of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI'19, 2296–2303. Honolulu, Hawaii, USA: AAAI Press.

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

Jansen, T., and Zarges, C. 2009. A theoretical analysis of immune inspired somatic contiguous hypermutations for function optimization. In *Proceedings of the 8th International Conference on Artificial Immune Systems*, ICARIS'09, 80–94. York, UK: Springer.

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

Jansen, T., and Zarges, C. 2014. Reevaluating immune-inspired hypermutations using the fixed budget perspective. *IEEE Transactions on Evolutionary Computation* 18(5):674–688.

Jansen, T.; Oliveto, P. S.; and Zarges, C. 2011. On the analysis of the immune-inspired B-cell algorithm for the vertex cover problem. In *Proceedings of the 10th International Conference on Artificial Immune Systems*, ICARIS'11, 117–131. Cambridge, UK: Springer.

Jiang, S., and Yang, S. 2015. An improved multiobjective optimization evolutionary algorithm based on decomposition for complex pareto fronts. *IEEE Transactions on Cybernetics* 46(2):421–437.

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

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

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

Lin, Q.; Ma, Y.; Chen, J.; Zhu, Q.; Coello, C. A. C.; Wong, K.-C.; and Chen, F. 2018. An adaptive immune-inspired multi-objective algorithm with multiple differential evolution strategies. *Information Sciences* 430:46–64.

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

Qian, C.; Yu, Y.; Tang, K.; Yao, X.; and Zhou, Z.-H. 2019. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence* 275:279–294.

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

Shang, R.; Jiao, L.; Liu, F.; and Ma, W. 2011. A novel immune clonal algorithm for MO problems. *IEEE Transactions on Evolutionary Computation* 16(1):35–50.

Xia, X., and Zhou, Y. 2018. On the effectiveness of immune inspired mutation operators in some discrete optimization problems. *Information Sciences* 426:87–100.

Xu, N.; Ding, Y.; Ren, L.; and Hao, K. 2018. Degeneration recognizing clonal selection algorithm for multimodal optimization. *IEEE Transactions on Cybernetics* 48(3):848–861.

Yao, G.; Ding, Y.; Ren, L.; Hao, K.; and Chen, L. 2016. An immune system-inspired rescheduling algorithm for workflow in cloud systems. *Knowledge-Based Systems* 99:39–50.

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

Zhou, Y.; He, J.; and Nie, Q. 2009. A comparative runtime analysis of heuristic algorithms for satisfiability problems. *Artificial Intelligence* 173(2):240–257.