An Analysis on Recombination in Multi-Objective Evolutionary Optimization

Chao Qian, Yang Yu*, Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China

Abstract

Evolutionary algorithms (EAs) are increasingly popular approaches to multi-objective optimization. One of their significant advantages is that they can directly optimize the Pareto front by evolving a population of solutions, where the recombination (also called crossover) operators are usually employed to reproduce new and potentially better solutions by mixing up solutions in the population. Recombination in multi-objective evolutionary algorithms is, however, mostly applied heuristically. In this paper, we investigate how from a theoretical viewpoint a recombination operator will affect a multi-objective EA. First, we employ artificial benchmark problems: the `Weighted LPTNO` problem (a generalization of the well-studied `LOTZ` problem), and the well-studied `COCZ` problem, for studying the effect of recombination. Our analysis discloses that recombination may accelerate the filling of the Pareto front by recombining diverse solutions and thus help solve multi-objective optimization. Because of this, for these two problems, we find that a multi-objective EA with recombination enabled achieves a better expected running time than any known EAs with recombination disabled. We further examine the effect of recombination on solving the multi-objective minimum spanning tree problem, which is an NP-Hard problem. Following our finding on the artificial problems, our analysis shows that recombination also helps accelerate filling the Pareto front and thus helps find approximate solutions faster.

Key words: Evolutionary algorithms, multi-objective optimization, recombination, crossover, running time, computational complexity

*Corresponding author
Email addresses: `qianc@lamda.nju.edu.cn` (Chao Qian), `yuy@nju.edu.cn` (Yang Yu), `zhouzh@nju.edu.cn` (Zhi-Hua Zhou)

1. Introduction

Multi-objective optimization [48] requires one to find a set of solutions called the optimal Pareto set, because the objectives usually conflict with each other. Evolutionary algorithms (EAs), which are a kind of stochastic metaheuristic optimization approach [4], are becoming increasingly popular for multi-objective optimization [6]. EAs maintain a population of solutions, and iteratively improve the population by reproducing new solutions from the population and updating the population to contain the best-so-far solutions. Because EAs are population-based approaches, they have an advantage of directly optimizing the Pareto front, instead of, e.g., running an algorithm many times each for a Pareto optimal solution.

A characterizing feature of EAs is the recombination operator for reproducing new solutions. Recombination (also called *crossover*) operators take two or more individual solutions from the maintained population and mix them up to form new solutions. Thus, they are population-based operators and typically appear in multi-objective EAs (MOEAs), e.g. the popular NSGA-II [7]. The MOEAs that use recombination operators have been successful in solving many real-world problems, e.g., electric power dispatch problem [1], multiprocessor system-on-chip design [15], aeronautical and aerospace engineering [3], and multicommodity capacitated network design [25]. However, recombination operators are understood only at the introductory level. Discovering the effect of recombination in MOEAs not only can enhance our understanding of this kind of nature-inspired operators, but also be helpful for designing improved algorithms. This paper studies the effect of recombination from a theoretical viewpoint.

1.1. Related Work

The theoretical foundation of EAs has developed quickly in the past few decades, e.g. [12, 21, 51, 38]. Most of the previous analyses focused on EAs with mutation operators, while only a few include recombination operators. We briefly review theoretical work on recombination operators.

Properties of recombination operators have been addressed in the scenario of single-objective optimization. Early empirical analyses include [31] and [47]. Later, running time analyses provided a theoretical understanding of recombination operators. Several crossover-only evolutionary algorithms were shown to be effective on the H-IFF problem which is difficult for any kind of mutation-based EAs [50, 8]. Jansen and Wegener [22, 23] proved that crossover operators can be crucially important on some artificial problems. Crossover operators subsequently have been shown to be useful in more cases. These include Ising models [16, 49], the TwoPaths instance class of the problem

of computing unique input-output sequences [30], some instance classes of the vertex cover problem [39], and the all-pairs shortest path problem [10, 9, 11]. Meanwhile, on the contrary, Richter et al. [44] produced the Ignoble Trail functions where a crossover operator was shown to be harmful. Recently, Kötzing et al. [27] found that crossover with ideal diversity can lead to drastic speedups on the OneMax and Jump problems, and they also analyzed how crossover probability and population size affect population diversity. The analysis approaches for recombination have been developed such as the *switch analysis* approach [52, 53] that compares the running time of an EA turning recombination on and off. While all these studies are in the scenario of single-objective optimization, the results are difficult to generalize to the scenario of multi-objective optimization. In particular, multi-objective optimization aims at finding a set of optimal and non-dominated solutions rather than a single optimal solution, where the situation becomes more complex and is untouched.

Early analyses of multi-objective EAs concentrate on the conditions under which MOEAs can find the Pareto optimal solutions given unlimited time, e.g., [20, 45, 46]. For running time analyses, studies on two bi-objective pseudo-Boolean functions the LOTZ and COCZ problems extended respectively from the well-studied LeadingOnes and OneMax problems [12] have led to some disclosure of limited time behaviors of MOEAs. Note that none of these previously analyzed MOEAs uses recombination operators. We rename the previous MOEAs in Table 1 for a clearer presentation. Table 2 lists the previous analyses results of these MOEAs for the two problems.

Original name	Unified name	Explanation
SEMO [28]	MOEA^{onebit}	A simple MOEA with one-bit mutation
GSEMO [18]	$\mathsf{MOEA}^{bitwise}$	A simple MOEA with bit-wise mutation
FEMO [28]	$\mathsf{MOEA}^{onebit}_{fair}$	A modification of SEMO so that every solution has equal total reproduction chance
GEMO [29]	$\mathsf{MOEA}^{onebit}_{greedy}$	A modification of SEMO so that only new born solutions dominating some current solutions have reproduction chance

Table 1: Unified names of the previously analyzed MOEAs in this paper.

The recent work by Neumann and Theile [35] is, to the best of our knowledge, the first and only work analyzing crossover operators in MOEAs. They proved that a crossover operator can speed up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. As discovered through their analysis, the crossover operator can be helpful in the interplay with the mutation operator, such that good solutions can be evolved efficiently.

We also note that there are studies that analyze MOEAs for solving single-objective problems, e.g., [36, 17, 34]. However, we concern ourselves with multi-objective problems.

1.2. Our Contribution

Multi-objective optimization aims at finding a set of optimal solutions with different balances of the objectives, which was not involved in the analysis for single-objective optimization. This paper extends our preliminary work [40] to investigate whether recombination operators can have any effect on solving the multi-objective optimization tasks.

We study the effect of recombination by comparing the performance of MOEAs using the diagonal multi-parent crossover [14] to that using only mutation. We derive the expected running time of the MOEAs using recombination together with the one-bit mutation and the bit-wise mutation, denoted respectively as $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$. The one-bit mutation implies a kind of local search, and the bit-wise mutation is regarded as a global search. We further indicate the probability of applying the recombination by using the subscript following "recomb", e.g., $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ uses the recombination with probability 0.5 and $\mathsf{MOEA}^{onebit}_{recomb,0}$ does not use the recombination. Since EAs are a kind of metaheuristic optimization approach, i.e., they are problem-independent, a typical way to study EAs theoretically is to use model problems to disclose the properties of EAs. In this paper, we first extend our preliminary work [40] to employ two artificial multi-objective problems: the `Weighted LPTNO` (Weighted Leading Positive Ones Trailing Negative Ones) problem and the well-studied `COCZ` (Count Ones Count Zeros) problem, for analyzing MOEAs. Note that `Weighted LPTNO` is generalized from the well-studied `LOTZ` (Leading Ones Trailing Zeros) problem. We then study MOEAs for the multi-objective minimum spanning tree (MST) problem, an NP-Hard problem [2, 13], which is more representative of real-world problems.

For the artificial problems, we derive the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$. We respectively compare them to that of $\mathsf{MOEA}^{onebit}_{recomb,0}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0}$ as well as previously analyzed MOEAs without recombination on the `LOTZ` and `COCZ` problems. Their expected running time is listed in Table 2. From the results, we can conclude that, among the MOEAs with one-bit mutation, $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ achieves the best performance, which is faster than the other mutation only MOEAs; among the MOEAs with bit-wise mutation, $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ is also better than the other mutation only MOEAs. We further carry out experiments to compensate for the theoretical bounds in Table 2 that are not tight, e.g., where only the upper bound is known for the MOEA^{onebit} on `COCZ`. The empirical observations suggest that, for the MOEAs whose exact asymptotic running time complexities are unknown, their running time is close to the derived upper bounds. Thus, the empirical observations confirm that $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ are the best among the compared MOEAs.

Through the analysis of $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ for the artificial problems, we discover that the recombination operator works in the studied situation by accelerating the filling of the Pareto

mutation	MOEA	LOTZ	COCZ
one-bit	MOEA^{onebit}	$\Theta(n^3)$ [28]	$O(n^2 \ln n)$ [29]
	$\mathsf{MOEA}^{onebit}_{fair}$	$O(n^2 \ln n)$ [28]	$\Omega(n^2), O(n^2 \ln n)$ [29]
	$\mathsf{MOEA}^{onebit}_{greedy}$	$O(n^2 \ln n)$ [29]	$\Theta(n^2)$ [29]
	$\mathsf{MOEA}^{onebit}_{recomb,0}$	$\Theta(n^3)$	$\Omega(n^2), O(n^2 \ln n)$
	$\mathsf{MOEA}^{onebit}_{recomb,0.5}$	$\Theta(n^2)$	$\Theta(n \ln n)$
bit-wise	$\mathsf{MOEA}^{bitwise}$	$O(n^3)$ [18]	$O(n^2 \ln n)$
	$\mathsf{MOEA}^{bitwise}_{recomb,0}$	$\Omega(n^2), O(n^3)$	$\Omega(n \ln n), O(n^2 \ln n)$
	$\mathsf{MOEA}^{bitwise}_{recomb,0.5}$	$\Theta(n^2)$	$\Theta(n \ln n)$

Table 2: Expected running time bounds of several MOEAs on LOTZ and COCZ, where the first column denotes the used mutation operator by the MOEAs in the corresponding row.

front through recombining the diverse optimal solutions that have been found. It is worth noting that this mechanism is different from that analyzed in [35], where the crossover operator works by its interplay with mutation. Moreover, this finding is unique to multi-objective optimization, as there is no Pareto front in single-objective situations.

On the multi-objective MST problem, we follow the idea that recombination can accelerate the filling of the Pareto front by recombining diverse solutions, and analyze the expected running time of $\mathsf{MOEA}^{bitwise}_{recomb}$. Let m and n denote the number of edges and nodes respectively. Let $w_{\max}$ and $w_{\min}$ denote the maximum and minimum among the maximum values for each kind of weight respectively. Let $|F^*|$ and $|conv(F^*)|$ denote the size of the optimal Pareto front and its convex sub-front (definitions are in Section 5.1) respectively, and $N_{gc} \geq 0$, $C_{\min} \geq 1$ be two parameters that depend on concrete problem instances. First, on a subclass of bi-objective MST problem with a strictly convex optimal Pareto front, we find that the expected running time of $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ has the upper bound of

$$O\Big(m^2 n w_{\min}\big(|F^*| + \frac{\ln n + \ln w_{\max}}{n w_{min}} - N_{gc}(1 - \frac{1}{m})\big)\Big).$$

By comparing with the previous result for $\mathsf{MOEA}^{bitwise}$ [33]

$$O\Big(m^2 n w_{\min}\big(|F^*| + \ln n + \ln w_{\max}\big)\Big),$$

$\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ appears to be more efficient. Moreover, to obtain the 2-approximation solutions for the general bi-objective MST problem, the expected running time of $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ has the upper bound of

$$O\Big(m^2 n w_{\min}\big(|conv(F^*)| + \frac{\ln n + \ln w_{\max}}{n w_{min}} - N_{gc}(C_{\min} - \frac{1}{m})\big)\Big),$$

which improves the previous result for $\mathsf{MOEA}^{bitwise}$ [33]

$$O\Big(m^2 n w_{\min}\big(|conv(F^*)| + \ln n + \ln w_{\max}\big)\Big).$$

The comparisons imply that recombination operators may help MOEAs in solving real-world multi-objective optimizations.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries on multi-objective optimization. Section 3 presents the $\mathsf{MOEA}^{onebit}_{recomb}$ and the $\mathsf{MOEA}^{bitwise}_{recomb}$ algorithms. Section 4 studies these two algorithms on the artificial problems, and Section 5 studies them on the multi-objective minimum spanning tree problem. Section 6 concludes.

2. Multi-Objective Optimization

Multi-objective optimization requires to simultaneously optimize two or more objective functions, as in Definition 1. When there are two objective functions, it is also called as *b*i-objective optimization. We consider maximization here, while minimization can be defined similarly.

Definition 1 (Multi-Objective Optimization)
Given a feasible solution space $\mathcal{X}$ and objective functions $f_1, \ldots, f_m$, the maximum multi-objective optimization aims to find the solution $\boldsymbol{x}^*$ satisfying

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{f}(\boldsymbol{x}) = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big)$$

where $\boldsymbol{f}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})\big)$ is the objective vector of the solution $\boldsymbol{x}$.

Usually, the objectives are conflicted, i.e., optimization of one objective alone will degrade the other objectives, and it is impossible to have one solution that optimizes all the objectives simultaneously. Therefore, multi-objective optimization tries to find a set of solutions according to some criteria. One commonly used criterion is the Pareto optimality, which utilizes the *domination* relation between solutions as in Definition 2. The solution set by Pareto optimality is called Pareto set, as in Definition 3.

Definition 2 (Domination)
Let $\boldsymbol{f} = (f_1, f_2, \ldots, f_m) : \mathcal{X} \to \mathbb{R}^m$ be the objective vector, where $\mathcal{X}$ is the feasible solution space, and $\mathbb{R}^m$ is the objective space. For two solutions $\boldsymbol{x}$ and $\boldsymbol{x}' \in \mathcal{X}$:

1. *$\boldsymbol{x}$* weakly dominates *$\boldsymbol{x}'$ if, for all i that $1 \le i \le m$, $f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{x}')$, denoted as $\succeq_{\boldsymbol{f}}$;*
2. *$\boldsymbol{x}$* dominates *$\boldsymbol{x}'$ if, $\boldsymbol{x}$ weakly dominates $\boldsymbol{x}'$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{x}')$ for some i, denoted as $\succ_{\boldsymbol{f}}$.*

Definition 3 (Pareto Optimality)
Let $\boldsymbol{f} = (f_1, f_2, \ldots, f_m) : \mathcal{X} \to \mathbb{R}^m$ be the objective vector, where $\mathcal{X}$ is the feasible solution space, and $\mathbb{R}^m$ is the objective space. A solution $\boldsymbol{x}$ is Pareto optimal *if there is no other solution in $\mathcal{X}$ that dominates $\boldsymbol{x}$. A set of solutions is called* Pareto set *if it contains only Pareto optimal solutions. The collection of objective values of a set of solutions is called the* front *of the set. If a set is a Pareto set, its front is also called the* Pareto front.

With the goal of finding the largest Pareto set, or called the *optimal Pareto set*, the running time of an MOEA is counted as the number of calls to $\boldsymbol{f}$ until it finds the Pareto front of the optimal Pareto set, or called the *optimal Pareto front*. That is, the MOEA should find at least one corresponding solution for each element in the optimal Pareto front. Note that this definition agrees with that in [28, 18, 29]. Since MOEAs are naturally stochastic algorithms, we measure the performance of MOEAs by the expected running time.

SEMO [28], as described in Algorithm 1, is a simple MOEA, and also the first analyzed MOEA due to its simplicity, which explains the common structure of various MOEAs. In the SEMO, $\mathcal{X}$ is the solution space, and it employs the one-bit mutation operator (i.e., step 5) where the action of "flipping" is implemented problem-specifically. Usually, when $\mathcal{X} = \{0,1\}^n$, the "flipping" action interchanges between 0 and 1 of a solution bit. A newly generated solution is compared with the solutions in the population, and then only non-dominated solutions are kept in the population.

Algorithm 1 (SEMO [28])
Given solution space $\mathcal{X}$ and objective function vector $\boldsymbol{f}$, SEMO consists of the following steps:

1. *Randomly choose $\boldsymbol{x} \in \mathcal{X}$*
2. $P \leftarrow \{\boldsymbol{x}\}$
3. ***loop***
4. *Choose $\boldsymbol{x}$ from P uniformly at random*
5. *Create $\boldsymbol{x}'$ by flipping a randomly chosen bit of $\boldsymbol{x}$*
6. ***if*** *$\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succeq_{\boldsymbol{f}} \boldsymbol{x}'$*
7. $P \leftarrow (P - \{\boldsymbol{z} \in P | \boldsymbol{x}' \succ_{\boldsymbol{f}} \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$
8. ***end if***
9. ***end loop***

Following SEMO, two modifications of SEMO, i.e., FEMO and GEMO, were proposed. FEMO [28] accelerates the exploration of the optimal set by using a fair sampling strategy, which makes that every solution produces the same number of offspring solutions at the end. GEMO [29] extends FEMO to achieve maximum progress toward the Pareto front by using a greedy selection mechanism, which allows only the newly included solution dominating some current solutions to have reproduction chance. All of these MOEAs use one-bit mutation operator. By replacing the one-bit mutation op-

erator which searches locally in SEMO with the bit-wise mutation operator which searches globally, the global SEMO (GSEMO) algorithm [18] was proposed and analyzed. That is, GSEMO is the same as SEMO except that step 5 in Algorithm 1 changes to be "Create $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ with probability $\frac{1}{n}$", where n is the solution length. For presenting these MOEAs clearly, we rename them in this paper, as in Table 1.

3. Recombination Enabled MOEAs

The recombination-incorporated MOEA ($\text{MOEA}^{onebit}_{recomb}$) studied in this paper is depicted in Algorithm 2, which is extended from the algorithm MOEA^{onebit} by incorporating a recombination operator. The components of $\text{MOEA}^{onebit}_{recomb}$ are explained in the following.

It is well known that the diversity of a population is important to the success of recombination operators, since recombination makes no progress from similar solutions. Therefore, we should employ some diversity control in $\text{MOEA}^{onebit}_{recomb}$. We use *objective diversity (od)* measure based on the assumption that the diversity of a set of solutions is consistent with the difference of their objective vectors. By this assumption, a population has a high diversity if it contains solutions with good objective values on different objectives. Thus, we define objective diversity of a set of solutions as the number of objectives, on which at least one solution in this set has a good objective value. Formally, for a set of solutions S, define a variable q_i for the i-th objective $(1 \leq i \leq m)$,

$$q_i = \begin{cases} 1 & \text{if } \max\{f_i(\boldsymbol{x}) \mid \boldsymbol{x} \in S\} \geq \theta_i, \\ 0 & \text{otherwise}, \end{cases}$$

where θ_i is the "goodness" threshold of the i-th objective, then the objective diversity of S

$$od(S) = \sum_{i=1}^{m} q_i.$$

Given m objectives, the largest value of objective diversity is m. Here, we use the objective diversity with $\theta_i =$ the minimal local optimal value of the i-th objective.

To make the initial population diverse enough, we use an initialization process, described in Definition 4. In the initialization process, m independent runs of randomized local search (RLS) are employed to optimize the m objective functions $f_1, f_2, \ldots, f_m$, each RLS corresponds to one objective. In the RLS, the solution is examined whether it is local optimal for an objective every N mutation steps, where N is the size of the neighbor space of a solution. When to check whether a solution is local optimal for an objective, we use the solutions having Hamming distance 1 with the current solution as the neighborhood. Thus, $N = n$, i.e., the length of a solution. This initialization procedure is terminated when local optimal solutions are found for all the objectives.

Definition 4 (Initialization)

Input: m *solutions* $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$ *from* $\mathcal{X}$*; Output:* m *local optimal solutions corresponding to one of the* m *objectives respectively; the Initialization procedure consists of the following steps:*

1. ***repeat***
2. ***repeat*** *the following process* N *times*
3. *Create* $\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_m$ *by flipping a randomly chosen bit of* $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$ *respectively*
4. ***if*** $f_i(\boldsymbol{x}'_i) > f_i(\boldsymbol{x}_i)$ *for an* i ***then***
5. $\boldsymbol{x}_i \leftarrow \boldsymbol{x}'_i$
6. ***end if***
7. ***end repeat***
8. ***until*** $\forall i$*:* $\boldsymbol{x}_i$ *is a local optimum for* f_i

This process makes the initial population contain one good solution for each objective, i.e., $q_i = 1$ $(1 \leq i \leq m)$. Thus, the objective diversity of the initial population is m. Since a solution in the population is eliminated only if it is dominated by a new solution, there always exist good solutions for each objective. As the result, the objective diversity of the population always keeps m, the maximal objective diversity, throughout the evolution process.

In each reproduction step, MOEA$^{onebit}_{recomb}$ picks a set of m solutions with the objective diversity at least $m/2$ from the current population to carry out recombination. In order to do so, the best solutions each for one of the randomly selected $m/2$ objectives are selected at first, denoted as a set P^1_s. The remaining $m - |P^1_s|$ solutions are randomly chosen from the population excluding P^1_s. Thus, the selected solutions for recombination have an objective diversity at least $m/2$.

In the reproduction procedure, we use the parameter p_c to control the use of recombination. That is, in each reproduction step, the offspring solutions are generated by recombination with probability p_c, and otherwise generated by mutation. In the end of each iteration, the offspring solutions P_o are used to update the current population. For an offspring solution, if there is no solution in the current population which can weakly dominate it, it will be included into the population, and then the population is cleaned up by removing all solutions that are no longer non-dominated.

Algorithm 2 (MOEA$^{onebit}_{recomb}$)

Given solution space $\mathcal{X}$ *and objective function vector* $\boldsymbol{f}$ *of length* m*, MOEA*$^{onebit}_{recomb}$ *consists of the following steps:*

1. $P \leftarrow$ *Initialization*(m *solutions randomly from* $\mathcal{X}$)
2. ***loop***
3. $P_s^1 \leftarrow$ *select the best solution in* P *for each of the randomly selected* $m/2$ *objectives*
4. $P_s^2 \leftarrow$ *randomly select* $m - |P_s^1|$ *solutions from* $P - P_s^1$
5. $P_s \leftarrow P_s^1 \cup P_s^2$
6. $r \leftarrow$ *uniformly chosen from* $[0,1]$ *at random*
7. ***if*** $r < p_c$ ***then***
8. $P_o \leftarrow$ *Recombination*(P_s)
9. ***else***
10. $P_o \leftarrow$ *for each solution* $\boldsymbol{x} \in P_s$*, flip a randomly chosen bit*
11. ***end if***
12. ***for*** *each solution* $\boldsymbol{x}' \in P_o$
13. ***if*** $\nexists \boldsymbol{z} \in P$ *such that* $\boldsymbol{z} \succeq_{\boldsymbol{f}} \boldsymbol{x}'$ ***then***
14. $P \leftarrow (P - \{\boldsymbol{z} \in P \mid \boldsymbol{x}' \succ_{\boldsymbol{f}} \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$
15. ***end if***
16. ***end for***
17. ***end loop***

The recombination operator employed in $\mathsf{MOEA}_{recomb}^{onebit}$ is the diagonal multi-parent crossover [14], as in Definition 5. For m solutions, diagonal multi-parent crossover randomly selects $m-1$ crossover points between adjacent bits and creates m offspring solutions by sequentially combining the components partitioned by the crossover points, which is a generalization of one-point crossover over two solutions.

Definition 5 (Recombination [14])
Given m *solutions whose length is* n*, randomly select* $m-1$ *crossover points from* $n-1$ *positions between adjacent bits, and create* m *offspring solutions as follows. Denote the order of the* m *parents as* $1, 2, ..., m$*. The* m *offspring solutions are generated by combining* m *components partitioned by the* $m-1$ *crossover points, where the components of the* i*-th* $(1 \leq i \leq m)$ *offspring solution sequentially come from parents* $i, i+1, ..., m-1, m, 1, ..., i-1$*.*

In the algorithm $\mathsf{MOEA}_{recomb}^{onebit}$, the employed mutation operator is the one-bit mutation operator (i.e., step 10), which searches locally. If there are a set of non-dominated solutions such that they weakly dominate all their Hamming neighbors, and the current population is contained in this set, the population cannot escape from this set through the one-bit mutation operator. To solve such difficulty, we can modify the one-bit mutation operator in $\mathsf{MOEA}_{recomb}^{onebit}$ to a more general mutation operator, the bit-wise mutation operator which searches globally. We call the modified algorithm $\mathsf{MOEA}_{recomb}^{bitwise}$, which will also be studied in our paper. $\mathsf{MOEA}_{recomb}^{bitwise}$ is almost the same algorithm

as $\text{MOEA}^{onebit}_{recomb}$. The only difference is that step 10 of $\text{MOEA}^{bitwise}_{recomb}$ is "$P_o \leftarrow$ for each solution $\boldsymbol{x} \in P_s$, flip each bit with probability $\frac{1}{n}$" while that of $\text{MOEA}^{onebit}_{recomb}$ is "$P_o \leftarrow$ for each solution $\boldsymbol{x} \in P_s$, flip a randomly chosen bit". Since $\text{MOEA}^{onebit}_{recomb}$ is an extension of MOEA^{onebit} by incorporating a recombination operator, $\text{MOEA}^{bitwise}_{recomb}$ can also be viewed as an extension of $\text{MOEA}^{bitwise}$ using the same way.

4. Analysis of Recombination on Artificial Problems

4.1. The Problems

Two bi-objective pseudo-Boolean model problems LOTZ (Leading Ones Trailing Zeros) and COCZ (Count Ones Count Zeros) are usually used to investigate the properties of MOEAs [28, 18, 29], as the listed in Table 2.

For LOTZ, the first objective is to maximize the number of leading one bits (the same as the LeadingOnes problem [12]), and the other objective is to maximize the number of trailing zero bits. These two objectives have n-order bit interactions.

Definition 6 (LOTZ [28])
The pseudo-Boolean function LOTZ: $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:

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

As analyzed in [28], the objective space of LOTZ can be partitioned into $n+1$ subsets F_i, where $i \in \{0, \ldots, n\}$ is the sum of the two objective values, i.e., $\boldsymbol{f}(\boldsymbol{x}) \in F_i$ if $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) = i$. Obviously, $F_n = \{(0,n), (1,n-1), \ldots, (n,0)\}$ is the optimal Pareto front, and the optimal Pareto set has $n+1$ elements, which are $0^n, 10^{n-1}, \ldots, 1^n$.

For COCZ, the two objectives are linear functions. The first objective is to maximize the number of one bits (the same as the OneMax problem [12]), and the other objective is to maximize the number of one bits in the first half of the solution plus the number of zero bits in the second half. The two objectives are corporative in maximizing the number of one bits in the first half of the solution, but conflict in the second half.

Definition 7 (COCZ [29])
The pseudo-Boolean function COCZ: $\{0,1\}^n \to \mathbb{N}^2$ is defined as follows:

$$\textit{COCZ}(\boldsymbol{x}) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} (1 - x_i) \right),$$

where n is even.

As analyzed in [29], the objective space of COCZ can be partitioned into $n/2+1$ subsets F_i, where $i \in \{0,\dots,n/2\}$ is the number of one bits in the first half of the solution. It is obvious that each F_i contains $n/2+1$ different objective vectors $(i+j, i+n/2-j)$, where $j \in \{0,\dots,n/2\}$ is the number of one bits in the second half. The optimal Pareto front is $F_{n/2} = \{(\frac{n}{2}, n), (\frac{n}{2}+1, n-1), \dots, (n, \frac{n}{2})\}$, and the optimal Pareto set is $\{1^{\frac{n}{2}} *^{\frac{n}{2}}; * \in \{0,1\}\}$, the size of which is $2^{\frac{n}{2}}$.

In this paper, we will use Weighted LPTNO (Weighted Leading Positive Ones Trailing Negative Ones) and COCZ to investigate the effect of recombination operators for MOEAs. Weighted LPTNO as in Definition 8 is a bi-objective pseudo-Boolean problem class, which is generalized from LOTZ. The first objective is to maximize the number of leading positive one bits, and the other objective is to maximize the number of trailing negative one bits.

Definition 8 (Weighted LPTNO)
The function Weighted LPTNO: $\{-1,1\}^n \to \mathbb{R}^2$ is defined as follows:

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

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

The objective vector of Weighted LPTNO can be represented as $(\sum_{k=1}^{i} 2^k w_k, \sum_{k=n-j+1}^{n} 2^{n+1-k} v_k)$, where i and j are the number of leading positive one bits and trailing negative one bits, respectively. The optimal Pareto set has $n+1$ elements, which are $1^n, 1^{n-1}(-1), \dots, 1(-1)^{n-1}, (-1)^n$. The optimal Pareto front is $\{(\sum_{k=1}^{i} 2^k w_k, \sum_{k=i+1}^{n} 2^{n+1-k} v_k) \mid 0 \le i \le n\}$.

It is worthwhile to note that Weighted LPTNO, though which can be considered as an extension of LOTZ by shifting the solution space from $\{0,1\}^n$ to $\{-1,1\}^n$ and adding weights w_i and v_i, has very different properties from LOTZ. For LOTZ, the optimal Pareto front is symmetric, i.e., if (a,b) is in the optimal Pareto front, so is (b,a), and is thus convex. However, for Weighted LPTNO, the optimal Pareto front can be nonsymmetric and nonconvex in general, which may result in much more complicated situations.

4.2. Analysis on the Weighted LPTNO Problem

First, we derive the running time of the initialization procedure of $\text{MOEA}^{onebit}_{recomb}$ and $\text{MOEA}^{bitwise}_{recomb}$. The initialization is to optimize the two objectives of a problem separately by two independent RLS. For Weighted LPTNO, the objectives have the same structure as the LeadingOnes problem, which we know that RLS takes $\Theta(n^2)$ time to optimize by Theorem 13 in [24].

During the optimization of RLS in the initialization, a solution is examined whether it is local optimal for an objective every n mutation steps. The running time of examining once is n because the

neighborhood size of a solution is n. Thus, the total running time of examining is the same as that of optimization. We then have the following lemma.

Lemma 1

On `Weighted LPTNO`*, the expected running time of the initialization procedure of* $MOEA_{recomb}^{onebit}$ *and* $MOEA_{recomb}^{bitwise}$ *is* $\Theta(n^2)$.

Then, we focus on the size of the population during the evolution process.

Lemma 2

On `Weighted LPTNO`*, the population size of MOEAs maintaining non-dominated solutions is always not larger than* $n+1$*, and equals to* $n+1$ *iff the population is the optimal Pareto set.*

Proof. Because the solutions in the population have a one-to-one correspondence with the objective vectors in the front of the population, the size of the population equals to the size of its front. Thus, we just need to investigate the size of the front of the population.

For the first objective of `Weighted LPTNO`, there are $n+1$ possible values, $\{\sum_{i=1}^{j} 2^i w_i \mid 0 \leq j \leq n\}$. In the front of the current population, any possible value of the first objective can have at most one corresponding value of the second objective, because two solutions with the same first objective value and different second objective values are comparable, which violates the property that the solutions in the population are non-dominated. Thus, the size of the front is not larger than $n+1$. If the size of the front equals to $n+1$, the values on the first dimension of the $n+1$ elements in the front are $0, 2w_1, \sum_{i=1}^{2} 2^i w_i, \ldots, \sum_{i=1}^{n} 2^i w_i$, respectively. Because the solutions in the population are non-dominated, the corresponding values on the second dimension must decrease, i.e., they are $\sum_{i=1}^{n} 2^{n+1-i} v_i, \sum_{i=2}^{n} 2^{n+1-i} v_i, \ldots, 2v_n, 0$, respectively. Thus, the front is just the optimal Pareto front. □

In the evolution process of the MOEAs analyzed in this paper, it is easy to see that the population contains at most one Pareto optimal solution for each element in the optimal Pareto front, and the Pareto optimal solution will never be eliminated once it has been found. Thus, to find the optimal Pareto front, it is sufficient to increase the number of Pareto optimal solutions enough (e.g., the size of the optimal Pareto front) times. We call the event that the number of Pareto optimal solutions increases in one step a success. Then, by analyzing the expected steps for a success and summing up the expected steps for all the required successes, we can get the expected running time bound of the MOEAs. In the following proof, we will often use this idea.

Note that we investigate the bi-objective problems in this paper, thus when selecting solutions for reproduction in $\mathrm{MOEA}_{recomb}^{onebit}$ and $\mathrm{MOEA}_{recomb}^{bitwise}$, the best solution in the current population for

a randomly selected objective is selected first, and then the other solution is randomly chosen from the remaining solutions. Also since the bi-objective problems are considered in this paper, the recombination operator used is just one-point crossover. In the following analysis, we always denote $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ with crossover probability $p_c = 0.5$ by $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ respectively, and denote them with crossover probability $p_c = 0$ by $\mathsf{MOEA}^{onebit}_{recomb,0}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0}$ respectively.

Theorem 1

On `Weighted LPTNO`*, the expected running time of* $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ *and* $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ *is* $\Theta(n^2)$.

Proof. For `Weighted LPTNO`, the size of the optimal Pareto front is $n+1$. After the initialization, the two Pareto optimal solutions 1^n and $(-1)^n$ have been found. Thus, it is sufficient to increase the number of Pareto optimal solutions $n-1$ times for finding the optimal Pareto front.

Then, we consider the expected steps for a success when starting from a population P containing $i+1$ $(1 \leq i \leq n-1)$ Pareto optimal solutions. In the selection procedure, since the two solutions 1^n and $(-1)^n$, which are optimal for the two objectives of `Weighted LPTNO` respectively, have been found, it actually selects one solution $\boldsymbol{x}$ randomly from $\{1^n,(-1)^n\}$ first, and then selects the other solution randomly from the remaining solutions $P-\{\boldsymbol{x}\}$. We then consider two cases in selection for the probability p of generating one new Pareto optimal solution by one-point crossover in one step: the two solutions 1^n and $(-1)^n$ are selected; one selected solution is either 1^n or $(-1)^n$ and the other selected one is a Pareto optimal solution from $P-\{1^n,(-1)^n\}$. In the former case which happens with probability $\frac{1}{|P|-1}$, $p=\frac{n-i}{n-1}$. In the latter case which happens with probability $\frac{1}{2(|P|-1)}$, suppose that the selected Pareto optimal solution from $P-\{1^n,(-1)^n\}$ has k $(0<k<n)$ leading positive ones, the number of Pareto optimal solutions having more than k leading positive ones in the current population is k' $(1 \leq k' \leq i-1)$ and the other selected solution is 1^n, then $p=\frac{n-k-k'}{n-1}$. Correspondingly, when the other selected solution is $(-1)^n$, then $p=\frac{k-i+k'}{n-1}$. Thus, in the latter case, $p=\frac{1}{2(|P|-1)}\cdot\frac{n-k-k'}{n-1}+\frac{1}{2(|P|-1)}\cdot\frac{k-i+k'}{n-1}=\frac{n-i}{2(|P|-1)(n-1)}$. By combining these two cases, $p=\frac{1}{|P|-1}\cdot\frac{n-i}{n-1}+(i-1)\cdot\frac{n-i}{2(|P|-1)(n-1)}=\frac{(i+1)(n-i)}{2(|P|-1)(n-1)}$, where the factor $(i-1)$ is because there are $i-1$ Pareto optimal solutions in $P-\{1^n,(-1)^n\}$ for selection in the latter case. Because the crossover probability is $\frac{1}{2}$, the number of Pareto optimal solutions increases in one step with probability at least $\frac{(i+1)(n-i)}{4(|P|-1)(n-1)} \geq \frac{(i+1)(n-i)}{4(n-1)^2}$, where the inequality is by $|P| \leq n$ from Lemma 2. Thus, the expected steps for a success when starting from a population containing $i+1$ Pareto optimal solutions is at most $\frac{4(n-1)^2}{(i+1)(n-i)}$.

Because two offspring solutions need to be evaluated in one reproduction step, the running time of one step is counted as 2. Thus, the expected running time to find the optimal Pareto front

after initialization is at most $2 \cdot \sum_{i=1}^{n-1} \frac{4(n-1)^2}{(i+1)(n-i)}$, i.e., $O(n \ln n)$. By combining the expected running time $\Theta(n^2)$ of the initialization in Lemma 1, the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ on `Weighted LPTNO` is $\Theta(n^2)$. □

Therefore, we have proved that the expected running time of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ with crossover probability 0.5 on `Weighted LPTNO` is $\Theta(n^2)$. The running time on the `LOTZ` problem, a special case of `Weighted LPTNO`, has been well studied for the MOEAs with only mutation. Compared with all the previously analyzed MOEAs with one-bit mutation or bit-wise mutation as in the 3rd column of Table 2, $\mathsf{MOEA}^{onebit}_{recomb,0.5}$/$\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ has better running time on `LOTZ`.

Then, we also analyze the running time of $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ turning off recombination on `Weighted LPTNO` to see whether recombination is crucial for the efficiency of $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ on this problem.

Theorem 2

On `Weighted LPTNO`, the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0}$ *is* $\Theta(n^3)$.

Proof. For `Weighted LPTNO`, the offspring Pareto optimal solution generated by one-bit mutation on a Pareto optimal solution must be adjacent to the parent Pareto optimal solution. After the initialization, the two Pareto optimal solutions 1^n and $(-1)^n$ have been found. Thus, it is easy to see that the population in any step of the evolutionary process is always constructed by two continuous subsets L and R of the optimal Pareto set $\{1^n, 1^{n-1}(-1), \ldots, (-1)^n\}$, where $1^n \in L$ and $(-1)^n \in R$. Then, we divide the optimization process into n phases, where the population in the i-th phase $(1 \leq i \leq n)$ consists of $i+1$ Pareto optimal solutions and the n-th phase corresponds to that the optimal Pareto set has been found. In the following, we will consider the probability of generating new Pareto optimal solutions in one step in each phase.

In the first phase, $|L| = 1$ and $|R| = 1$, where $|*|$ denotes the size of a set. For the reproduction, the two solutions 1^n and $(-1)^n$ are selected. For 1^n, the offspring solution generated by one-bit mutation can be accepted only if the rightmost positive one bit is mutated. For $(-1)^n$, only if the leftmost negative one bit is mutated, the offspring solution can be accepted. Thus, in one step, two new Pareto optimal solutions will be generated simultaneously with probability $\frac{1}{n^2}$; only one new Pareto optimal solution will be generated with probability $\frac{2(n-1)}{n^2}$; otherwise, no new Pareto optimal solution will be generated.

Then, we consider the i-th phase $(1 < i \leq n-1)$. In the selection procedure of reproduction, since 1^n and $(-1)^n$, which are optimal for the two objectives of `Weighted LPTNO` respectively, have been found, one solution will be randomly selected from $\{1^n, (-1)^n\}$, and the other solution will be

randomly selected from the remaining solutions. The probabilities for two selected solutions by this procedure are $\frac{1}{2}$ and $\frac{1}{i}$, respectively. Then, there are two cases.

Case 1: $\min\{|L|,|R|\} > 1$. In this case, one-bit mutation on either 1^n or $(-1)^n$ will never generate new Pareto optimal solutions, and only the rightmost solution of L or the leftmost solution of R can generate one new Pareto optimal solution with probability $\frac{1}{n}$ by one-bit mutation. Thus, one new Pareto optimal solution can be generated in one step with probability $\frac{2}{in}$; otherwise, no new Pareto optimal solution will be generated.

Case 2: $\min\{|L|,|R|\} = 1$. Suppose that $|L| = 1$. If the selected solution from $\{1^n,(-1)^n\}$ is $(-1)^n$, one-bit mutation on $(-1)^n$ will never generate new Pareto optimal solutions, and only when the other selected solution is 1^n or the leftmost solution of R, mutation on it can generate one new Pareto optimal solution with probability $\frac{1}{n}$. If the selected solution from $\{1^n,(-1)^n\}$ is 1^n, by mutation on 1^n, one new Pareto optimal solution $1^{n-1}(-1)$ can be generated with probability $\frac{1}{n}$, and only when the other selected solution is the leftmost solution of R, mutation on it can generate one new Pareto optimal solution with probability $\frac{1}{n}$. Thus, when $i < n-1$, in one step, only one new Pareto optimal solution will be generated while $\min\{|L|,|R|\}$ is still 1 with probability $\frac{1}{in} - \frac{1}{2in^2}$; one or two new Pareto optimal solutions will be generated while $\min\{|L|,|R|\} > 1$ with probability $\frac{1}{2n} + \frac{1}{2in}$; otherwise, no new Pareto optimal solution will be generated. When $i = n-1$, the last undiscovered Pareto optimal solution will be found in one step with probability $\frac{n^2+2n-1}{2(n-1)n^2}$.

We know that during the evolution process after initialization, the state of the population will change as follows:

1. start at the 1st phase;
2. transfer to case 2;
3. stay at case 2;
4. transfer to case 1;
5. stay at case 1;
6. end at the n-th phase, i.e, the optimal Pareto front is found.

It may skip steps 2 and 3, because the population can directly transfer to case 1 when leaving phase 1; it may also skip steps 4 and 5, because the population can always stay at case 2 until the optimal Pareto front is found.

From the analysis above, we know that the probability of generating one new Pareto optimal solution in the process of steps 3 and 5 is $\Theta(\frac{1}{in})$. Moreover, the probability of transferring at steps 2 and 4 is $\Omega(\frac{1}{n^2})$. Thus, the expected steps after the initialization procedure to generate the optimal Pareto front is $\Theta(\sum_{i=1}^{n-1} in) = \Theta(n^3)$. By combining the expected running time $\Theta(n^2)$ of the initialization in Lemma 1, the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0}$ on `Weighted LPTNO` is $\Theta(n^3)$. □

Theorem 3

On `Weighted LPTNO`*, the expected running time of* $\mathsf{MOEA}^{bitwise}_{recomb,0}$ *is* $\Omega(n^2)$ *and* $O(n^3)$.

Proof. After the initialization procedure, the two Pareto optimal solutions 1^n and $(-1)^n$ have been found. Thus, it is sufficient to increase the number of Pareto optimal solutions $n-1$ times to find the optimal Pareto front.

Before finding the optimal Pareto front, there always exists at least one Pareto optimal solution in the current population which can generate one new Pareto optimal solution by flipping just the rightmost positive one bit or the leftmost negative one bit. Because the probability of selecting a specific solution is at least $\frac{1}{n-1}$ by Lemma 2, the number of Pareto optimal solutions increases in one step with probability at least $\frac{1}{n-1}\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en(n-1)}$.

Thus, the expected running time to find the optimal Pareto front after the initialization procedure is at most $2en(n-1)^2$. By combining the expected running time $\Theta(n^2)$ of the initialization procedure in Lemma 1, the expected running time of the whole evolution process is $\Omega(n^2)$ and $O(n^3)$. □

We have proved that the expected running time of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ without crossover on `Weighted LPTNO` is $\Theta(n^3)/\Omega(n^2)$, which increases from $\Theta(n^2)$ of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ with crossover probability 0.5. Thus, recombination is crucial for the efficiency of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ on `Weighted LPTNO`.

From the analysis of $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ on `Weighted LPTNO`, we can find that recombination works by accelerating the filling of the Pareto front through recombining diverse optimal solutions found-so-far. For example, when the two diverse Pareto optimal solutions 1^n and $1^{n/2}(-1)^{n/2}$ are selected for reproduction, the probability of generating offspring Pareto optimal solutions which are different from the parents by one-point crossover is $\frac{n/2-1}{n}$, while the probability on any two selected solutions by mutation (one-bit or bit-wise) is at most $\frac{4}{n}$.

4.3. Analysis on the `COCZ` *Problem*

First, the initialization of $\mathsf{MOEA}^{onebit}_{recomb}$ and $\mathsf{MOEA}^{bitwise}_{recomb}$ is to optimize the two objectives of `COCZ`, which have the same structure as the OneMax problem, by two independent RLS. By Theorem 11 in [24], it is known that the running time of RLS on OneMax is $\Theta(n \ln n)$. During the optimization of RLS in the initialization, a solution is examined whether it is local optimal for an objective every n mutation steps. The running time of examining once is n. Thus, the total running time of examining is the same as that of optimization. Then, we can get the following lemma.

Lemma 3

On COCZ*, the expected running time of the initialization procedure of* $\mathit{MOEA}^{onebit}_{recomb}$ *and* $\mathit{MOEA}^{bitwise}_{recomb}$ *is* $\Theta(n \ln n)$.

Then, we bound the size of the population during the evolution.

Lemma 4

On COCZ*, the population size of MOEAs maintaining non-dominated solutions is always not larger than* $n/2+1$.

Proof. By the definition of COCZ, the objective vector can be represented as $(i+j, i+n/2-j)$, where i and j $(0 \leq i, j \leq n/2)$ are the number of one bits in the first and second half of a solution, respectively. For each value of j, there can be only one corresponding value of i, because two objective vectors with the same j value and different i values are comparable, which violates the property that the MOEAs maintain non-dominated solutions. Thus, the front of the population contains at most $n/2+1$ objective vectors.

Because the population has a one-to-one correspondence with its front, the size of the population equals to the size of its front. Thus, the population size is not larger than $n/2+1$. □

Theorem 4

On COCZ*, the expected running time of* $\mathit{MOEA}^{onebit}_{recomb,0.5}$ *and* $\mathit{MOEA}^{bitwise}_{recomb,0.5}$ *is* $\Theta(n \ln n)$.

Proof. For COCZ, the size of the optimal Pareto front is $\frac{n}{2}+1$. After the initialization procedure, the two Pareto optimal solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$ have been found. Thus, it is sufficient to increase the number of Pareto optimal solutions $\frac{n}{2}-1$ times to find the optimal Pareto front.

Then, we consider the expected steps for a success when starting from a population P containing $i+1$ $(1 \leq i \leq \frac{n}{2}-1)$ Pareto optimal solutions. In the selection procedure, since the two solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$, which are optimal for the two objectives of COCZ respectively, have been found, the algorithm actually selects one solution $\boldsymbol{x}$ randomly from $\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$ first, and then selects the other solution randomly from the remaining solutions $P-\{\boldsymbol{x}\}$. Then, we consider two cases in selection for the probability p of generating one or two new Pareto optimal solutions by one-point crossover in one step: the two solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$ are selected; one selected solution is either 1^n or $1^{\frac{n}{2}}0^{\frac{n}{2}}$ and the other selected one is a Pareto optimal solution from $P-\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$. In the former case which happens with probability $\frac{1}{|P|-1}$, $p \geq \frac{n/2-i}{n-1}$. In the latter case which happens with probability $\frac{1}{2(|P|-1)}$, suppose that the selected Pareto optimal solution from $P-\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$ has k $(1 \leq k \leq \frac{n}{2}-1)$ 0 bits, the number of Pareto optimal solutions having less than k 0 bits in the current population is k' $(1 \leq k' \leq i-1)$ and the other selected solution is 1^n, then $p \geq \frac{k-k'}{n-1}$. Correspondingly, when

the other selected solution is $1^{\frac{n}{2}}0^{\frac{n}{2}}$, then $p \geq \frac{n/2-k-i+k'}{n-1}$. Thus, in the latter case, $p \geq \frac{1}{2(|P|-1)} \cdot \frac{k-k'}{n-1} + \frac{1}{2(|P|-1)} \cdot \frac{n/2-k-i+k'}{n-1} = \frac{n/2-i}{2(|P|-1)(n-1)}$. By combining these two cases, $p \geq \frac{1}{|P|-1} \cdot \frac{n/2-i}{n-1} + (i-1) \cdot \frac{n/2-i}{2(|P|-1)(n-1)} = \frac{(i+1)(n/2-i)}{2(|P|-1)(n-1)}$. Because the crossover probability is $\frac{1}{2}$, the number of Pareto optimal solutions increases in one step with probability at least $\frac{(i+1)(n/2-i)}{4(|P|-1)(n-1)} \geq \frac{(i+1)(n/2-i)}{2n(n-1)}$, where the inequality is by $|P| \leq n/2+1$ from Lemma 4. Thus, the expected steps for a success is at most $\frac{2n(n-1)}{(i+1)(n/2-i)}$.

Therefore, the expected running time to find the optimal Pareto front is at most $2 \cdot \sum_{i=1}^{\frac{n}{2}-1} \frac{2n(n-1)}{(i+1)(n/2-i)}$, i.e., $O(n \ln n)$. By combining the expected running time $\Theta(n \ln n)$ of the initialization procedure in Lemma 3, the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ on COCZ is $\Theta(n \ln n)$. □

Therefore, we have proved that the expected running time of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ with crossover probability 0.5 on COCZ is $\Theta(n \ln n)$. Compared with all the previously analyzed MOEAs with one-bit mutation as in the last cell of the 2nd row of Table 2, $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ has better running time on COCZ. For $\mathsf{MOEA}^{bitwise}_{recomb}$, it uses bit-wise mutation. Note that $\mathsf{MOEA}^{bitwise}$ is the only previously analyzed MOEA with bit-wise mutation, and its expected running time on COCZ is unknown. Thus, for the comparison purpose, we derive the running time of $\mathsf{MOEA}^{bitwise}$ on COCZ.

Theorem 5

On COCZ, the expected running time of $\mathsf{MOEA}^{bitwise}$ is $O(n^2 \ln n)$.

Proof. We divide the evolutionary process into two phases. The first phase starts after initialization and finishes until the first Pareto optimal solution is found. The second phase finishes until the optimal Pareto front is found.

For the first phase, let j $(0 \leq j \leq n/2)$ denote the maximal number of 1 bits in the first half of the solutions in the current population. Because a solution with less 1 bits in its first half cannot dominate a solution with more 1 bits in its first half, j cannot decrease. Then, j increases in one step with probability at least $\frac{1}{n/2+1}\frac{n/2-j}{n}(1-\frac{1}{n})^{n-1} \geq \frac{n/2-j}{en(n/2+1)}$, since it is sufficient to select the solution with j number of 1 bits in its first half for mutation, flip one 0 bit in the first half of this solution and keep the other bits unchanged, where the probability of selection is at least $\frac{1}{n/2+1}$ by Lemma 4 and the probability of mutation is $\frac{n/2-j}{n}(1-\frac{1}{n})^{n-1}$. Because any solution with $\frac{n}{2}$ number of 1 bits in its first half is a Pareto optimal solution, $\frac{n}{2}$ such steps increasing the value of j are sufficient to find the first Pareto optimal solution. Thus, the expected running time of the first phase is at most $\sum_{j=0}^{n/2-1} \frac{en(n/2+1)}{n/2-j}$, i.e., $O(n^2 \ln n)$.

For the second phase, before finding the optimal Pareto front, there always exists at least one Pareto optimal solution in the current population which can generate one new Pareto optimal solution by

flipping just one 1 bit or one 0 bit in its second half. We call such Pareto optimal solutions boundary Pareto optimal solutions. Thus, the number of Pareto optimal solutions can increase by 1 in one step with probability at least $\frac{1}{n/2+1}\frac{\min\{i,n/2-i\}}{n}(1-\frac{1}{n})^{n-1} \geq \frac{\min\{i,n/2-i\}}{en(n/2+1)}$, since it is sufficient to select one boundary Pareto optimal solution for mutation, and flip just one 1 bit or one 0 bit in its second half, where the probability of selection is at least $\frac{1}{n/2+1}$ by Lemma 4, the probability of mutation is at least $\frac{\min\{i,n/2-i\}}{n}(1-\frac{1}{n})^{n-1}$, and i is the number of 1 bits in the second half of the selected boundary Pareto optimal solution. Because $\frac{n}{2}$ such steps increasing the number of Pareto optimal solutions are sufficient to find the optimal Pareto front of COCZ, the expected running time of the second phase is at most $\sum_{i=1}^{\lceil n/4\rceil} 2 \cdot \frac{en(n/2+1)}{i}$, i.e., $O(n^2 \ln n)$.

By combining the running time of these two phases, the expected running time of the whole evolution process is $O(n^2 \ln n)$. □

Therefore, compared with MOEA^{bitwise} on COCZ, MOEA$^{bitwise}_{recomb,0.5}$ has better upper bound of running time. Then, we analyze the running time of MOEA$^{onebit}_{recomb}$ and MOEA$^{bitwise}_{recomb}$ turning off recombination on COCZ to see whether recombination is crucial for the efficiency of MOEA$^{onebit}_{recomb}$ and MOEA$^{bitwise}_{recomb}$ on this problem.

Theorem 6

On COCZ, the expected running time of MOEA$^{onebit}_{recomb,0}$ is $\Omega(n^2)$ and $O(n^2 \ln n)$.

Proof. For COCZ, we define that two Pareto optimal solutions are consecutive if the difference of the number of 0 bits for these two solutions is 1. Then, the offspring Pareto optimal solution generated by one-bit mutation on a Pareto optimal solution must be consecutive with the parent Pareto optimal solution. After the initialization, the two Pareto optimal solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$ have been found. Then, it is easy to see that the population in the evolutionary process is always constructed by two continuous set of Pareto optimal solutions L and R, where $1^n \in L$, $1^{\frac{n}{2}}0^{\frac{n}{2}} \in R$, and every two adjacent Pareto optimal solutions in L or R are consecutive. Then, we divide the optimization process into $\frac{n}{2}$ phases, where the population in the i-th phase $(1 \leq i \leq \frac{n}{2})$ consists of $i+1$ Pareto optimal solutions and the $\frac{n}{2}$-th phase corresponds to that the optimal Pareto front has been found. In the following, we will consider the probability of generating new Pareto optimal solutions in one step in each phase.

In the first phase, $|L| = 1$ and $|R| = 1$. For the reproduction, the two solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$ are selected. For 1^n, the offspring solution generated by one-bit mutation will be accepted if the 1 bit in the second half is mutated. For $1^{\frac{n}{2}}0^{\frac{n}{2}}$, if the 0 bit is mutated, the offspring solution will be accepted. Thus, in one step, two new Pareto optimal solutions will be generated simultaneously with proba-

bility $\frac{1}{4}$; only one new Pareto optimal solution will be generated with probability $\frac{1}{2}$; otherwise, no new Pareto optimal solution will be generated.

Then, we consider the i-th phase ($1 < i \leq \frac{n}{2} - 1$). In the selection procedure of reproduction, since 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$, which are optimal for the two objectives of COCZ respectively, have been found, one solution will be randomly selected from $\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$, and the other solution will be randomly selected from the remaining solutions. The probabilities for two selected solutions by this procedure are $\frac{1}{2}$ and $\frac{1}{i}$, respectively. Then, there are two cases.

Case 1: $\min\{|L|, |R|\} > 1$. In this case, one-bit mutation on either 1^n or $1^{\frac{n}{2}}0^{\frac{n}{2}}$ will never generate new Pareto optimal solutions, and only the rightmost solution in L and the leftmost solution in R can generate one new Pareto optimal solution with probability $\frac{n/2-(|L|-1)}{n}$ and $\frac{n/2-(|R|-1)}{n}$ respectively by one-bit mutation. Thus, one new Pareto optimal solution can be generated in one step with probability $\frac{n/2-(|L|-1)}{in} + \frac{n/2-(|R|-1)}{in} = \frac{n-i+1}{in}$; otherwise, no new Pareto optimal solution will be generated.

Case 2: $\min\{|L|, |R|\} = 1$. Suppose that $|L| = 1$. If the selected solution from $\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$ is $1^{\frac{n}{2}}0^{\frac{n}{2}}$, one-bit mutation on it will never generate new Pareto optimal solutions, and only when the other selected solution is 1^n or the leftmost solution of R, mutation on it can generate one new Pareto optimal solution with probability $\frac{1}{2}$ and $\frac{n/2-(i-1)}{n}$, respectively. If the selected solution from $\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$ is 1^n, by mutation on 1^n, one new Pareto optimal solution can be generated with probability $\frac{1}{2}$, and only when the other selected solution is the leftmost solution of R, mutation on it can generate one new Pareto optimal solution with probability $\frac{n/2-(i-1)}{n}$. Thus, when $i < \frac{n}{2} - 1$, in one step, only one new Pareto optimal solution will be generated while $\min\{|L|, |R|\}$ is still 1 with probability $\frac{3(n/2-i+1)}{4in}$; one or two new Pareto optimal solutions will be generated while $\min\{|L|, |R|\} > 1$ with probability $\frac{i+1}{4i}$; otherwise, no new Pareto optimal solution will be generated. When $i = \frac{n}{2} - 1$, the last undiscovered Pareto optimal solution will be found in one step with probability $\frac{n^2+12}{4n^2-8n}$.

The state of the population during the evolution will change as similar as that in the proof of Theorem 2. From the analysis above, we know that the probabilities of generating one new Pareto optimal solution in the process of steps 3 and 5 are $\Theta(\frac{n/2-i+1}{in})$ and $\Theta(\frac{n-i+1}{in})$, respectively. Moreover, the probability of transferring at steps 2 and 4 is $\Theta(1)$. Thus, the expected steps after the initialization procedure to generate the optimal Pareto front is $\Omega(\sum_{i=1}^{n/2-1} \frac{in}{n-i+1})$ and $O(\sum_{i=1}^{n/2-1} \frac{in}{n/2-i+1})$, i.e., $\Omega(n^2)$ and $O(n^2 \ln n)$. By combining the expected running time $\Theta(n \ln n)$ of the initialization in Lemma 3, the expected running time of $\mathsf{MOEA}^{onebit}_{recomb,0}$ on COCZ is $\Omega(n^2)$ and $O(n^2 \ln n)$. □

Theorem 7

On COCZ, the expected running time of $\mathsf{MOEA}^{bitwise}_{recomb,0}$ is $\Omega(n \ln n)$ and $O(n^2 \ln n)$.

Proof. After the initialization procedure, the two Pareto optimal solutions 1^n and $1^{\frac{n}{2}}0^{\frac{n}{2}}$ have been found. Thus, it is sufficient to increase the number of Pareto optimal solutions $\frac{n}{2}-1$ times to find the optimal Pareto front.

Before finding the optimal Pareto front, there always exists at least one Pareto optimal solution in the current population which can generate one new Pareto optimal solution by flipping just one 1 bit or one 0 bit in its second half. Because the probability of selecting a specific solution is at least $\frac{1}{n/2}$ by Lemma 4, the number of Pareto optimal solutions increases in one step with probability at least $\frac{1}{n/2}\frac{\min\{i,n/2-i\}}{n}(1-\frac{1}{n})^{n-1} \geq \frac{\min\{i,n/2-i\}}{en^2/2}$, where i is the number of 1 bits in the second half of the selected Pareto optimal solution.

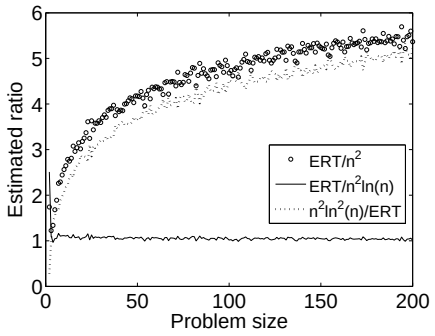
Then, the expected running time to find the optimal Pareto front after the initialization is at most $2\cdot\sum_{i=1}^{\lceil(n-2)/4\rceil} 2\cdot\frac{en^2/2}{i}$, i.e., $O(n^2\ln n)$. By combining the running time $\Theta(n\ln n)$ of the initialization in Lemma 3, the expected running time of the whole evolution process is $\Omega(n\ln n)$ and $O(n^2\ln n)$. □

We have proved that the expected running time of $\text{MOEA}^{onebit}_{recomb}$/$\text{MOEA}^{bitwise}_{recomb}$ without crossover on COCZ is $\Omega(n^2)$/$\Omega(n\ln n)$, which increases from $\Theta(n\ln n)$ of $\text{MOEA}^{onebit}_{recomb}$/$\text{MOEA}^{bitwise}_{recomb}$ with crossover probability 0.5. Thus, recombination is crucial for the efficiency of $\text{MOEA}^{onebit}_{recomb}$/$\text{MOEA}^{bitwise}_{recomb}$ on the COCZ problem. From the analysis, we can also find that recombination works by accelerating the filling of the Pareto front through recombining diverse optimal solutions found-so-far, as that we have found from the analysis on the Weighted LPTNO problem.

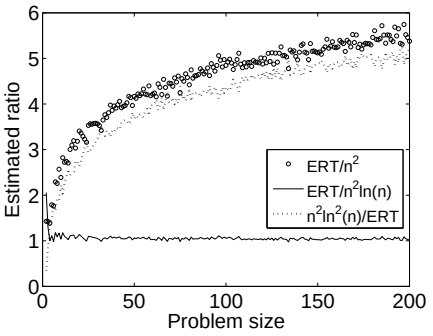
4.4. Empirical Verification

Some running time bounds in Table 2 are not tight, which makes the comparison of performance between MOEAs not strict. For example, when comparing $\text{MOEA}^{onebit}_{recomb,0.5}$ with MOEA^{onebit} on COCZ, we can only say that the expected running time of $\text{MOEA}^{onebit}_{recomb,0.5}$ is better than the upper bound of the expected running time of MOEA^{onebit} proved-so-far. To have a more meaningful comparison, we estimate the running time order by experiments. On each problem size, we repeat independent runs of an MOEA 1000 times, and then the average running time is recorded as an estimation of the expected running time, which will be called as ERT for short. Figures 1 and 2 plot the results.

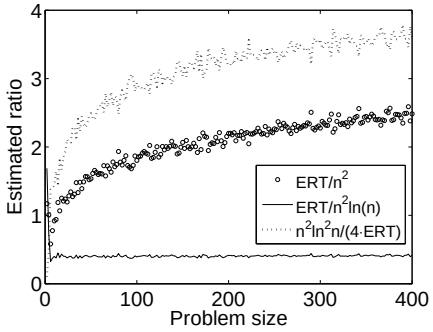
From Figure 1(a), we can observe that for $\text{MOEA}^{onebit}_{fair}$ on LOTZ, the curve of the ERT divided by $n^2\ln n$ tends to a constant, and both the curve of the ERT divided by n^2 and the curve of $n^2\ln^2 n$ divided by the ERT grow in a closely logarithmic trend. Therefore, the observation suggests that the expected running time of $\text{MOEA}^{onebit}_{fair}$ is approximately in the order of $n^2\ln n$. Similarly, Figure 1(b) and (c) suggest that both the ERT of $\text{MOEA}^{onebit}_{greedy}$ on LOTZ and the ERT of MOEA^{onebit} on COCZ is approximately in the order of $n^2\ln n$. Therefore, by combining the proved results, we can conclude that



(a) MOEA$^{onebit}_{fair}$ on LOTZ

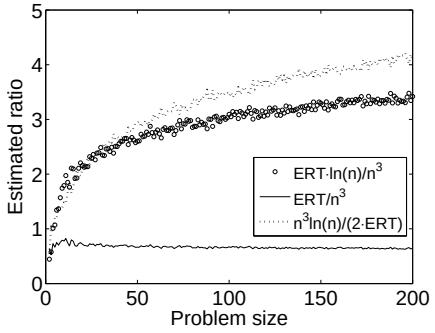


(b) MOEA$^{onebit}_{greedy}$ on LOTZ

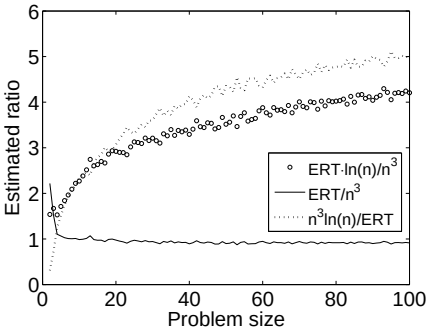


(c) MOEA^{onebit} on COCZ

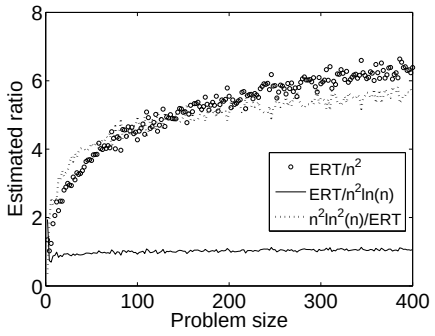
Figure 1: Estimated ERT of several MOEAs with one-bit mutation operator on LOTZ and COCZ.



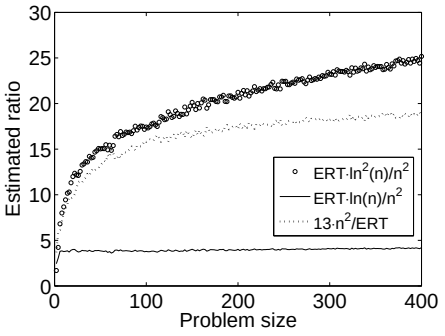
(a) MOEA^{bitwise} on LOTZ



(b) MOEA$^{bitwise}_{recomb,0}$ on LOTZ

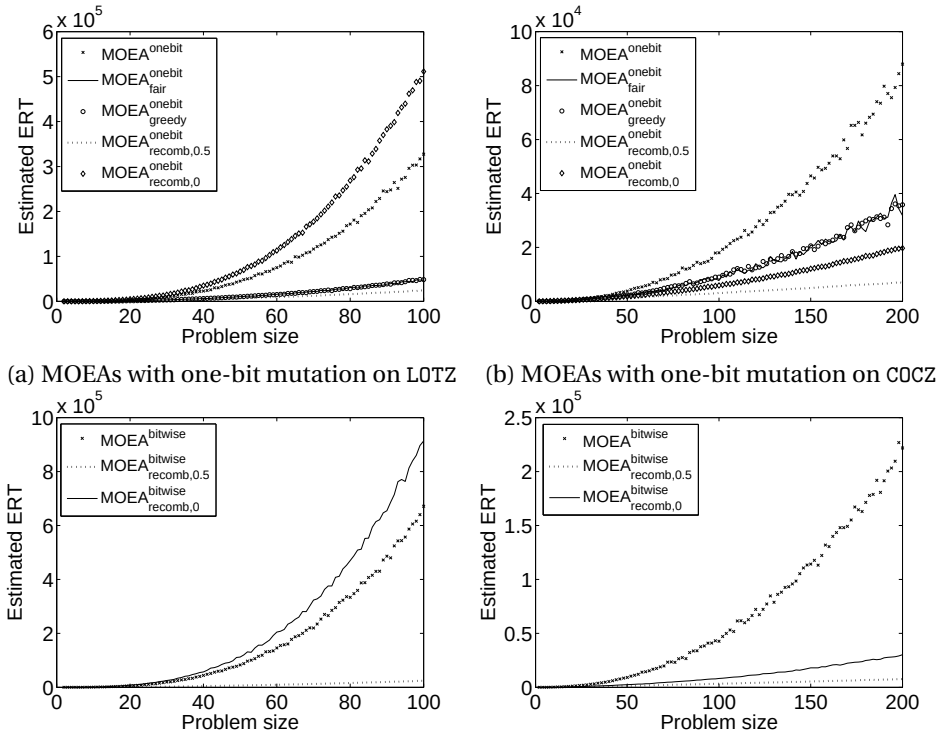


(c) MOEA^{bitwise} on COCZ



(d) MOEA$^{bitwise}_{recomb,0}$ on COCZ

Figure 2: Estimated ERT of several MOEAs with bit-wise mutation operator on LOTZ and COCZ.



(a) MOEAs with one-bit mutation on LOTZ (b) MOEAs with one-bit mutation on COCZ

(c) MOEAs with bit-wise mutation on LOTZ (d) MOEAs with bit-wise mutation on COCZ

Figure 3: Comparison between estimated ERT of several MOEAs on LOTZ and COCZ.

$\mathsf{MOEA}^{onebit}_{recomb,0.5}$ is the most efficient algorithm among the MOEAs with one-bit mutation on LOTZ and COCZ.

Figure 2 suggests that the ERT of $\mathsf{MOEA}^{bitwise}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0}$ on LOTZ is approximately in the order of n^3, and the ERT of $\mathsf{MOEA}^{bitwise}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0}$ on COCZ is approximately in the order of $n^2 \ln n$ and $n^2/\ln n$, respectively. Therefore, we can conclude that $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ is the most efficient algorithm among the MOEAs with bit-wise mutation on the two problems.

Then, in Figure 3, we compare the expected running time of $\mathsf{MOEA}^{onebit}_{recomb}$/$\mathsf{MOEA}^{bitwise}_{recomb}$ with other MOEAs empirically when the problem size is not large. The problem size for LOTZ is set in the range from 2 to 100, and that for the COCZ problem is set even integers in the range from 2 to 200. It can be observed that even in this empirical comparison, the ERT of $\mathsf{MOEA}^{onebit}_{recomb,0.5}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ on the two problems is the smallest among the corresponding MOEAs.

5. Analysis of Recombination on the Multi-Objective Minimum Spanning Tree Problem

5.1. The Problem

The single-objective minimum spanning tree (MST) problem is a classical polynomial solvable combinatorial problem, which is to find a connected subgraph with the minimum weight from an undi-

rected connected graph. However, the multi-objective MST problem with at least two objectives, where each edge of a graph is assigned to a weight vector rather than a single weight, has been proved to be NP-Hard [2, 13]. It has found many real applications in designing networks. For example, in the design for a layout of telecommunication systems, besides the cost for connection between terminals, other factors, e.g., the time for communication and the network reliability, may also need to be considered. For solving this problem, many algorithms have been proposed for approximation, e.g., deterministic algorithms [19, 43] and evolutionary algorithms [55, 5].

The multi-objective MST problem can be described as follows. Given an undirected connected graph $G = (V, E)$ on n vertices and m edges where V and $E = \{e_1, e_2, \ldots, e_m\}$ are the vertex set and edge set respectively, each edge e_i has a weight vector $(w_1^i, w_2^i, \ldots, w_k^i)$, where $w_j^i > 0$ is the value of edge e_i with respect to the j-th weight. The goal is to find connected subgraphs $G' = (V, E' \subseteq E)$ which minimize the objective vector

$$(\sum_{e_i \in E'} w_1^i, \sum_{e_i \in E'} w_2^i, \ldots, \sum_{e_i \in E'} w_k^i).$$

The special case with $k = 1$ is just the single-objective MST problem. Let $w_j^{\max}$ denote the maximal value for the j-th weight (i.e., $w_j^{\max} = \max\{w_j^i | 1 \leq i \leq m\}$), $w_{\max} = \max\{w_j^{\max} | 1 \leq j \leq k\}$ and $w_{\min} = \min\{w_j^{\max} | 1 \leq j \leq k\}$.

For MOEAs solving the multi-objective MST problem, a solution $\boldsymbol{x}$ is usually represented by a Boolean string of length m, i.e., $\boldsymbol{x} \in \{0, 1\}^m$, where $x_i = 1$ means that the edge e_i is selected by $\boldsymbol{x}$ [42]. A commonly used fitness function for minimizing the j-th objective [33] is

$$f_j(\boldsymbol{x}) = (c(\boldsymbol{x}) - 1)w_{ub}^2 + (\sum_{i=1}^m x_i - n + 1)w_{ub} + \sum_{i=1}^m x_i w_j^i, \tag{1}$$

where $c(\boldsymbol{x})$ is the number of connected components of the subgraph described by $\boldsymbol{x}$, and $w_{ub} = n^2 w_{\max}$. Note that in this fitness function, the first term $(c(\boldsymbol{x}) - 1)w_{ub}^2$ makes that a subgraph with fewer connected components is better, the second term $(\sum_{i=1}^m x_i - n + 1)w_{ub}$ makes that a connected subgraph with fewer edges is better, and the last term $\sum_{i=1}^m x_i w_j^i$ makes that a spanning tree with a smaller weight is better. That is, $f_j(\boldsymbol{x})$ is minimized with respect to the lexicographic order of $(c(\boldsymbol{x}), \sum_{i=1}^m x_i, \sum_{i=1}^m x_i w_j^i)$.

By the fitness function of Eq.1, all the objectives are consistent on decreasing the number of connected components and decreasing the number of edges of a connected subgraph. Thus, a Pareto optimal solution must be a spanning tree.

We will consider the bi-objective MST problem in our analysis. In particular, we will analyze solving a subclass of bi-objective MST problem with a strictly convex optimal Pareto front, and approximating the general bi-objective MST problem. The two tasks are described as follows.

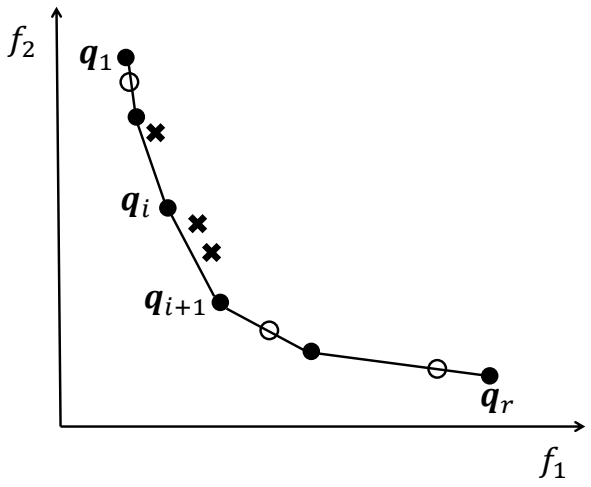


Figure 4: A Pareto front and its (strictly) convex sub-front.

Let F denote a Pareto front of a bi-objective minimization problem. The convex and strictly convex sub-front of F, denoted as $conv(F)$ and $sconv(F)$ respectively, is defined in Definition 9.

Definition 9 ((Strictly) Convex Sub-Front)
Given a Pareto front F of a bi-objective minimization problem, let the convex sub-front $conv(F)$ be the smallest subset of F such that

$$\forall \boldsymbol{q} \in F - conv(F),\ \exists \boldsymbol{q}', \boldsymbol{q}'' \in conv(F),\ \exists c_1, c_2 > 0 :\ \boldsymbol{q} = c_1\boldsymbol{q}' + c_2\boldsymbol{q}'' \wedge c_1 + c_2 > 1,$$

and let the strictly convex sub-front $sconv(F)$ be the smallest subset of F such that

$$\forall \boldsymbol{q} \in F - sconv(F),\ \exists \boldsymbol{q}', \boldsymbol{q}'' \in sconv(F),\ \exists c_1, c_2 > 0 :\ \boldsymbol{q} = c_1\boldsymbol{q}' + c_2\boldsymbol{q}'' \wedge c_1 + c_2 \geq 1.$$

For example, in Figure 4, F consists of all the points represented by $\bullet$, $\circ$ and $\times$, then $conv(F)$ consists of the points represented by both $\bullet$ and $\circ$, and $sconv(F)$ consists of the points represented by $\bullet$.

Let F^* denote the optimal Pareto front. Our first task is to analyze MOEAs solving a subclass of bi-objective MST problem where $F^* = sconv(F^*)$, i.e., the optimal Pareto front equals its strictly convex sub-front.

Our second task is to approximate the optimal Pareto front of the general bi-objective MST problem, particularly, find $(1+\epsilon)$-approximation as defined in Definition 10, which means that for every Pareto optimal solution we have a solution that is $(1+\epsilon)$ close on every objective. Note that, approximation here is considered for minimization. Then, the expected running time of MOEAs for approximation is the number of fitness evaluations until a desired approximation of the optimal Pareto front is found.

Definition 10 ($(1+\epsilon)$-Approximation)
Given a k-objective optimization task, for any $\epsilon > 0$, a set P of solutions is a $(1+\epsilon)$-approximation of the optimal Pareto front if $\forall \boldsymbol{q}^* = (q_1^*, \ldots, q_k^*) \in F^*$, i.e., any point in the optimal Pareto front, there exists a solution $\boldsymbol{x} \in P$ such that $\forall i = 1, \ldots, k : f_i(\boldsymbol{x}) \leq (1+\epsilon) \cdot q_i^*$.

5.2. Analysis

In this section, we analyze the expected running time of $\mathsf{MOEA}_{recomb}^{bitwise}$ on the bi-objective MST problem for the above two tasks, i.e., finding the optimal Pareto front for the bi-objective MST problem with $F^* = sconv(F^*)$, and finding a 2-approximation of the optimal Pareto front for the general bi-objective MST problem.

For the first task, we first analyze the initialization procedure of $\mathsf{MOEA}_{recomb}^{bitwise}$ on the bi-objective MST problem. From Definition 4, we know that the initialization is to optimize two single-objective MST problems with respect to w_1 and w_2 separately by two independent RLS. For RLS, it generates an offspring solution by flipping a randomly chosen bit of the parent solution. For a spanning tree, it cannot be improved by one-bit flip, which will lead to either an unconnected subgraph or a non-spanning tree; it can decrease its weight by flipping two proper bits, which inserts an edge leading to a cycle and deletes an existed edge with a larger weight in the created cycle. Thus, we modify the RLS by its reproduction behavior. That is, with probability 0.5, it generates an offspring by flipping a randomly chosen bit of the parent; otherwise, it generates an offspring solution by flipping two randomly chosen bits. From Theorem 2 in [37], it is known that the expected running time of such RLS for finding a minimum spanning tree with respect to w_1 (or w_2) is upper bounded by $O(m^2(\ln n + \ln w_1^{\max})$ (or $O(m^2(\ln n + \ln w_2^{\max}))$. Here, when examining whether a solution is local optimal for an objective, we use the solutions having Hamming distance not larger than 2 with the current solution as its neighbor space. Thus the parameter N in Definition 4 is $\Theta(m^2)$, i.e., the examining will be done every $\Theta(m^2)$ iteration. Because the running time of examining once is $\Theta(m^2)$ by the neighborhood size $\Theta(m^2)$ of a solution, the total running time of examining in the initialization is as same as that of optimization. Then, we have the following lemma.

Lemma 5
On the bi-objective MST problem, the expected running time of the initialization procedure of $\mathsf{MOEA}_{recomb}^{bitwise}$ is $O(m^2(\ln n + \ln w_{\max}))$.

Lemma 6 gives a property for the points in $sconv(F^*)$, which will be used in the following analysis. We denote these points by $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r$ in the lexicographic order (i.e., the first value of $\boldsymbol{q}_i$ increases with i), as showed in Figure 4. Let $d(T, T') = |E(T) - E(T')|$ denote the distance of two spanning

trees T and T', i.e., the minimum number of two edge exchanges for constructing T' from T, where $E(T)$ denotes the edge set of T.

Lemma 6 ([33])
If $F^* = sconv(F^*)$, for each Pareto optimal spanning tree T with objective vector $\boldsymbol{q}_i$ $(1 \leq i < r)$, there always exists a Pareto optimal spanning tree T' with objective vector $\boldsymbol{q}_{i+1}$ such that $d(T,T') = 1$.

We then define a specific optimization path of MOEAs for finding $sconv(F^*)$ of a bi-objective MST problem instance, as in Definition 11. At any time of the optimization, it uses crossover if crossover can produce new elements in $sconv(F^*)$; otherwise, it uses mutation.

Definition 11 (Crossover-First-Mutation-Second Path)
For a bi-objective MST problem instance, assume that two Pareto optimal solutions T_1^* and T_r^* for $\boldsymbol{q}_1$ and $\boldsymbol{q}_r$ respectively have been found. A crossover-first-mutation-second path *is an optimization path of MOEAs for finding the remaining part of $sconv(F^*)$ (i.e., $\{\boldsymbol{q}_2,\ldots,\boldsymbol{q}_{r-1}\}$), which behaves as follows: at any time of the optimization, if crossover on T_1^* (or T_r^*) and any other Pareto optimal solution with objective vector $\boldsymbol{q}_i$ that has been found can generate a Pareto optimal solution with objective vector $\boldsymbol{q}_j$ that has not been found, it will always apply crossover until crossover generates it; otherwise, assuming that $\{\boldsymbol{q}_1,\ldots,\boldsymbol{q}_i\}$ have been found and $\boldsymbol{q}_{i+1}$ has not been found, it will always apply mutation until mutation generates a Pareto optimal solution with objective vector $\boldsymbol{q}_{i+1}$. The process of generating a new $\boldsymbol{q}_i$ by crossover only and that by mutation only on this path are called* good crossover *and* good mutation*, respectively.*

In the following analysis, we will not distinguish a solution and the subgraph that it represents for convenience, and we will use $w_j(e_i)$ and $w_j(T)$ to denote the value of an edge e_i and a spanning tree T for the j-th weight w_j respectively, i.e., $w_j(e_i) = w_j^i$ and $w_j(T) = \sum_{e\in E(T)} w_j(e)$. Given a bi-objective MST problem instance and two Pareto optimal solutions T_1^* and T_r^* for $\boldsymbol{q}_1$ and $\boldsymbol{q}_r$ respectively, let $N_{gc}(T_1^*,T_r^*)$ denote the maximum number of good crossover for all possible crossover-first-mutation-second paths starting from T_1^* and T_r^*, and let $N_{gc} = \min\{N_{gc}(T_1^*,T_r^*) \mid f(T_1^*) = \boldsymbol{q}_1 \wedge f(T_r^*) = \boldsymbol{q}_r\}$.

Theorem 8
The expected running time of $MOEA^{bitwise}_{recomb,0.5}$ on the bi-objective MST problem with $F^* = sconv(F^*)$ until finding the optimal Pareto front is $O(mnw_{\min}N_{gc} + m^2nw_{\min}(|F^*| - N_{gc}) + m^2(\ln n + \ln w_{\max}))$, where $0 \leq N_{gc} \leq |F^*|$.

Proof. We divide the optimization process into two phases. The first phase starts after initialization and finishes until two Pareto optimal solutions for $\boldsymbol{q}_1$ and $\boldsymbol{q}_r$ respectively are found. The second phase finishes until the optimal Pareto front is found.

In the first phase, two minimum spanning trees with respect to w_1 and w_2 respectively have been found after the initialization procedure. Then, we are to analyze the expected running time until a Pareto optimal solution T_1^* for $\boldsymbol{q}_1$ is found. T_1^* is actually a minimum spanning tree for w_1; meanwhile, it is also Pareto optimal, i.e., it has the minimum weight for w_2 among all minimum spanning trees for w_1. Note that, a minimum spanning tree for w_1 will be always existed in the population, because it has the minimum value on the first objective f_1 with respect to w_1, and a non-minimum spanning tree for w_1 has a larger value on f_1 and thus cannot dominate it. Let T_1 denote the minimum spanning tree for w_1 in the current population. Then, we analyze the expected running time for constructing T_1^* from T_1. As T_1^* is a minimum spanning tree for w_1, from [32], we know that there exists a bijection α from $E(T_1^*) - E(T_1)$ to $E(T_1) - E(T_1^*)$ such that for each edge $e \in E(T_1^*) - E(T_1)$, $\alpha(e) \in Cyc(T_1, e)$ and $w_1(e) \leq w_1(\alpha(e))$, where $Cyc(T_1, e)$ is the cycle led by the insertion of e into T_1. Because T_1 is also a minimum spanning tree for w_1, it must hold that $w_1(e) = w_1(\alpha(e))$. We also have $w_2(e) \leq w_2(\alpha(e))$, because otherwise we can construct a spanning tree T_1' with $w_2(T_1') < w_2(T_1^*)$ and $w_1(T_1') = w_1(T_1^*)$, which contradicts that T_1^* is Pareto optimal. Thus, we have shown that for constructing T_1^* from T_1, there always exist k ($k = |E(T_1^*) - E(T_1)| > 0$) number of two edge exchanges which can keep the offspring spanning tree minimum for w_1 and improve it for w_2. Note that, (1+1)-EA is a simple single-objective EA which maintains one solution, reproduces an offspring solution by bit-wise mutation and updates the parent solution if the offspring is better in every iteration. Thus, if we assume that T_1 is always selected for mutation in every iteration, the process of constructing T_1^* from T_1 follows the optimization procedure of (1+1)-EA for constructing a minimum spanning tree from a non-minimum spanning tree with respect to w_2, the expected running time of which has been proved to be $O(m^2(\ln n + \ln w_2^{\max}))$ from Theorem 2 in [37]. Since T_1 is optimal for the first objective, it will be selected for reproduction with probability at least $\frac{1}{2}$ in every iteration. The probability of employing mutation in reproduction for $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ is $\frac{1}{2}$. Thus, the probability of selecting T_1 for mutation in each iteration is $\Theta(1)$, which implies that the expected running time for finding T_1^* is $O(m^2(\ln n + \ln w_2^{\max}))$. Similarly, we can prove that the expected running time for finding a corresponding Pareto optimal solution T_r^* for $\boldsymbol{q}_r$ is $O(m^2(\ln n + \ln w_1^{\max}))$. Thus, the expected running time of the first phase is $O(m^2(\ln n + \ln w_{\max}))$.

Then, in the second phase, we are to analyze the expected running time for finding the remaining part of the optimal Pareto front (i.e., $\{\boldsymbol{q}_2, \ldots, \boldsymbol{q}_{r-1}\}$). We consider an arbitrary crossover-first-mutation-second path starting from T_1^* and T_r^*. Obviously, its expected running time is an upper bound on the expected running time of this phase, because at any time of the optimization, there are many other possible ways to find new Pareto optimal solutions, except the way used by the path. Let P denote the current population. Because T_1^* and T_r^* that have been found are optimal for the two objectives with respect to w_1 and w_2 respectively, in the selection procedure for reproduction, a solution $\boldsymbol{x}_1$ will be first selected from $\{T_1^*, T_r^*\}$ randomly, and then another solution $\boldsymbol{x}_2$

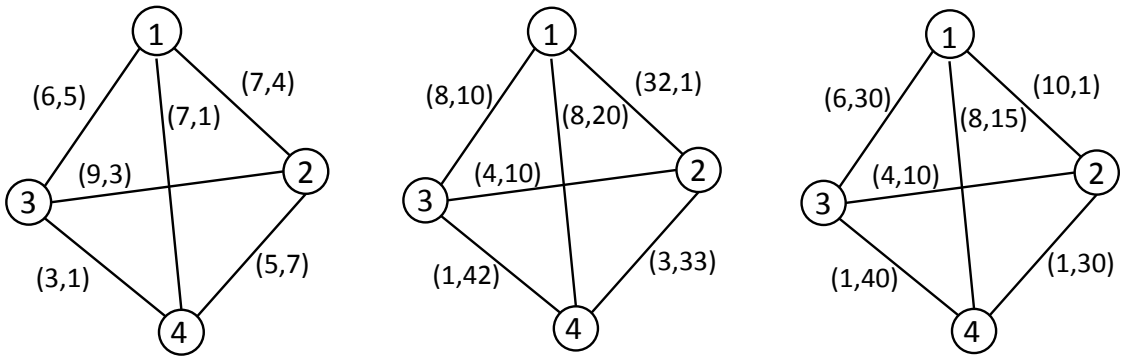
will be selected randomly from the remaining solutions $P-\{\boldsymbol{x}_1\}$. The probabilities for two selected solutions by this procedure are $\frac{1}{2}$ and $\frac{1}{|P|-1}$, respectively. Then, for good crossover, the probability of generating a new $\boldsymbol{q}_i$ in one step is at least $\frac{1}{2}\cdot(\frac{1}{2}\cdot\frac{1}{|P|-1})\cdot\frac{1}{m}$, where $\frac{1}{2}$ is the probability of applying crossover, $(\frac{1}{2}\cdot\frac{1}{|P|-1})$ is the probability of selecting two specific Pareto optimal solutions and $\frac{1}{m}$ is the probability of selecting a proper crossover point; and for good mutation, the probability of generating a new $\boldsymbol{q}_i$ in one step is at least $\frac{1}{2}\cdot\frac{1}{|P|-1}\cdot\frac{1}{m^2}(1-\frac{1}{m})^{m-2}$ by Lemma 6, where $\frac{1}{2}$ is the probability of applying mutation, $\frac{1}{|P|-1}$ is the lower bound for the probability of selecting a specific solution, and $\frac{1}{m^2}(1-\frac{1}{m})^{m-2}$ is the probability of flipping two specific bits. Because for any value of one objective, there exists at most one corresponding solution in the population, we have $|P|\leq nw_{\min}$. Thus, the above two probabilities become at least $\frac{1}{4mnw_{\min}}$ and $\frac{1}{2em^2nw_{\min}}$ respectively, which implies that the expected running time of a good crossover and a good mutation is at most $4mnw_{\min}$ and $2em^2nw_{\min}$ respectively. Then, we can get that the expected running time of the crossover-first-mutation-second path with the maximum number of good crossover is at most $4mnw_{\min}\cdot N_{gc}(T_1^*,T_r^*)+2em^2nw_{\min}\cdot(r-2-N_{gc}(T_1^*,T_r^*))\leq 4mnw_{\min}\cdot N_{gc}+2em^2nw_{\min}\cdot(r-2-N_{gc})$, which is also an upper bound for the expected running time of this phase.

By combining the expected running time of the above two phases and that of the initialization in Lemma 5, we can get that the expected running time of the whole process is upper bounded by $O(mnw_{\min}N_{gc}+m^2nw_{\min}(r-N_{gc})+m^2(\ln n+\ln w_{\max}))$, where $r=|F^*|$. □

Then, we analyze the approximation for the general bi-objective MST problem. By Lemma 7, we know that we can find a 2-approximation of the optimal Pareto front by finding $sconv(F^*)$. For deriving its expected running time, we can follow the above proof except that the expected steps for a good mutation is different, because F^* usually does not equal to $sconv(F^*)$ which makes Lemma 6 not hold. Let $C_i=\{\boldsymbol{r}\in conv(F^*)\mid \boldsymbol{q}_i,\boldsymbol{q}_{i+1}\in sconv(F^*)\wedge r_1>q_{i,1}\wedge r_1\leq q_{i+1,1}\}$, where $r_1,q_{i,1},q_{i+1,1}$ are the first value of $\boldsymbol{r},\boldsymbol{q}_i,\boldsymbol{q}_{i+1}$, respectively. That is, C_i consists of points on the i-th linear segment of $conv(F^*)$. From the proof of Theorem 9 in [33], it is known that for constructing $\boldsymbol{q}_{i+1}$ from $\boldsymbol{q}_i$, there exists an evolution path where the elements are from C_i and in the lexicographic order, and the adjacent two elements can be reached by a single two edge exchange. The expected running time for a specific two edge exchange on a specific solution is $O(m^2nw_{\min})$. Thus, a good mutation needs $O(m^2nw_{\min}|C_i|)$ expected running time. Let $C_{\min}=\min\{|C_i|\mid 1\leq i\leq r-2\}$. Then, we can derive Theorem 9.

Lemma 7 ([33])

For the minimization of two objective functions with positive objective values, a solution set containing a corresponding solution for each point in $sconv(F^*)$ is a 2-approximation of the optimal Pareto front F^*.



(a) Example 1 (b) Example 2 (c) Example 3

Figure 5: Three examples of the bi-objective MST problem.

Theorem 9

The expected running time of MOEA$^{bitwise}_{recomb,0.5}$ on the bi-objective MST problem until finding a 2*-approximation of the optimal Pareto front is* $O(mnw_{\min}N_{gc} + m^2nw_{\min}(|conv(F^*)| - N_{gc}C_{\min}) + m^2(\ln n + \ln w_{\max}))$*, where* $0 \leq N_{gc} \leq |sconv(F^*)|$.

Neumann [33] has derived the expected running time of MOEA^{bitwise} on the bi-objective MST problem for the two tasks. The results are showed in Theorem 10.

Theorem 10 ([33])

For MOEA^{bitwise} on the bi-objective MST problem,

(1) if $F^* = sconv(F^*)$*, the expected running time until finding the optimal Pareto front is* $O(m^2nw_{\min}(|F^*| + \ln n + \ln w_{\max}))$*;*

(2) in general cases, the expected running time until finding a 2*-approximation of the optimal Pareto front is* $O(m^2nw_{\min}(|conv(F^*)| + \ln n + \ln w_{\max}))$.

By comparing Theorem 10 with Theorem 8 and 9, we find that recombination can make MOEAs more efficient on the bi-objective MST problem. For the special case with $F^* = sconv(F^*)$, we can observe that the expected running time for finding $\boldsymbol{q}_1$ and $\boldsymbol{q}_r$ decreases from $m^2nw_{\min}(\ln n + \ln w_{\max})$ to $m^2(\ln n + \ln w_{\max})$, and the expected running time for finding the remaining part of $sconv(F^*)$ decreases by $N_{gc}(m^2nw_{\min} - mnw_{\min})$. For the approximation in general cases, we can observe the similar result.

5.3. Empirical Verification

We give some examples to show that N_{gc} is usually larger than 0. Figure 5 lists three complete graphs with 4 nodes, where each edge has two weights. We use a Boolean string of length 6 to represent a solution, where the bits correspond to the edges $1 \leftrightarrow 2, 1 \leftrightarrow 3, 1 \leftrightarrow 4, 2 \leftrightarrow 3, 2 \leftrightarrow 4, 3 \leftrightarrow 4$, respectively. For a complete graph with 4 nodes, there are 16 possible spanning trees, as in the 1st

column of Table 3. Table 3 also shows the objective value of each spanning tree for each example problem, and the elements of the optimal Pareto front are denoted by †. The optimal Pareto fronts of these three examples are plotted in the left part of Figure 6, where the corresponding convex subfront consists of the points on the piece-wise linear function. We can observe that $F^* = sconv(F^*)$ for example 1 and 2. In Table 3, we also denote the points of $sconv(F^*)$ in the lexicographic order by $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_r$, where $r = |sconv(F^*)|$. We then construct the crossover-first-mutation-second paths of these three examples, and show them in the right part of Figure 6, where 'cor' and 'mut' represent the crossover and mutation respectively, and the number i represents the execution order of the current operator on the path. Let's look at the path of example 1. First, crossover on the two corresponding Pareto optimal solutions for $\boldsymbol{q}_1$ and $\boldsymbol{q}_4$ can generate the Pareto optimal solution for $\boldsymbol{q}_2$; after that, crossover cannot generate new Pareto optimal solutions, then mutation applies to the solution for $\boldsymbol{q}_2$ to generate the Pareto optimal solution for $\boldsymbol{q}_3$. We can observe that $N_{gc} > 0$ on these examples. We also compare the expected running time of $\mathsf{MOEA}^{bitwise}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ on these example problems empirically. For each MOEA, we repeat independent runs for 1000 times, and then the average running time is recorded as an estimation of the expected running time. The results are showed in Table 4. We can observe that $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ is always better than $\mathsf{MOEA}^{bitwise}$, which is consistent with our theoretical result.

The comparison in Table 4 is on three specific bi-objective MST problem instances. We also generally compare the performance of $\mathsf{MOEA}^{bitwise}$ and $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ by experiments. We use the complete graphs with the number of nodes $n = 4, 5, 6$, respectively. For each size of n, the average running time of 1000 independent runs will be used as an estimation of the expected running time. For each independent run, following [55, 26], the graph is constructed by setting the weight vector of each edge be two integers uniformly randomly selected from $[10, 100]$ and $[10, 50]$, respectively. The results are showed in Table 5. We can observe that $\mathsf{MOEA}^{bitwise}_{recomb,0.5}$ always needs less expected running time than $\mathsf{MOEA}^{bitwise}$.

6. Discussions and Conclusions

This paper extends our preliminary work [40]. Multi-objective evolutionary algorithms, which typically use recombination operators, have been successfully applied in many practical situations, and some theoretical results of MOEAs have been derived in recent years. However, previously analyzed MOEAs rarely incorporate recombination operators. This paper theoretically investigates whether recombination operators can be useful in the scenario of multi-objective optimization.

The Pareto front is a property of multi-objective optimization that is not involved in single-objective optimization, thus we investigate whether a recombination operator can have an effect on solv-

spanning tree	example 1	example 2	example 3
111000	(20,10)	(48,31)	(24,46)
110010	(18,16)	(43,44)	(17,61)
110001	(16,10)	(41,53)	(17,71)
101100	(23,8)	(44,31)†, $\boldsymbol{q}_5$	(22,26)†, $\boldsymbol{q}_5$
101001	(17,6)†, $\boldsymbol{q}_3$	(41,63)	(19,56)
100110	(21,14)	(39,44)	(15,41)†, $\boldsymbol{q}_4$
100101	(19,8)	(37,53)	(15,51)
100011	(15,12)	(36,76)	(12,71)
011100	(22,9)	(20,40)†, $\boldsymbol{q}_4$	(18,55)
011010	(18,13)	(19,63)	(15,75)
010110	(20,15)	(15,53)†, $\boldsymbol{q}_3$	(11,70)†, $\boldsymbol{q}_2$
010101	(18,9)	(13,62)†, $\boldsymbol{q}_2$	(11,80)
010011	(14,13)†, $\boldsymbol{q}_1$	(12,85)†, $\boldsymbol{q}_1$	(8,100)†, $\boldsymbol{q}_1$
001110	(21,11)	(15,63)	(13,55)†, $\boldsymbol{q}_3$
001101	(19,5)†, $\boldsymbol{q}_4$	(13,72)	(13,65)
001011	(15,9)†, $\boldsymbol{q}_2$	(12,95)	(10,85)†

Table 3: The objective values for all possible spanning trees.

MOEA	example 1	example 2	example 3
MOEA^{bitwise}	278.81	337.89	361.17
MOEA$^{bitwise}_{recomb,0.5}$	196.26	269.95	239.51

Table 4: Comparison of estimated expected running time on three example instances.

Problem size	OPF	MOEA^{bitwise}	MOEA$^{bitwise}_{recomb,0.5}$
$n = 4$	3.14	203.76	142.54
$n = 5$	6.44	1403	1204.8
$n = 6$	10.02	5892.9	4828.6

Table 5: Comparison of estimated expected running time on random complete graphs, where problem size denotes the number of nodes and OPF denotes the average size of the optimal Pareto front.

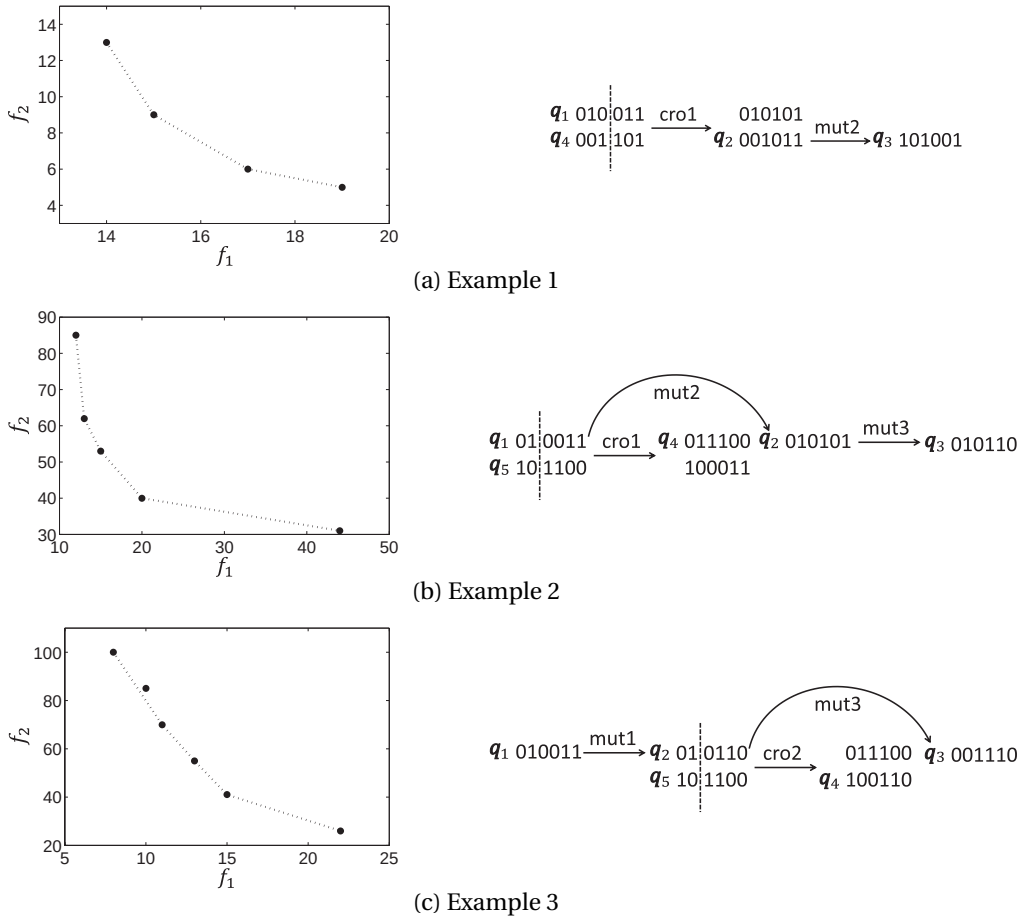


Figure 6: The optimal Pareto front and the crossover-first-mutation-second path.

ing the Pareto front of multi-objective problems. First, we analyze the running time of two multi-objective evolutionary algorithms with a recombination operator: $\text{MOEA}^{onebit}_{recomb}$ using one-bit mutation and $\text{MOEA}^{bitwise}_{recomb}$ using bit-wise mutation, on two artificial model problems `Weighted LPTNO` and `COCZ`. The analytic results of $\text{MOEA}^{onebit}_{recomb}$/$\text{MOEA}^{bitwise}_{recomb}$ turning recombination on and off on these two problems show the helpfulness of crossover. By comparing with the previously analyzed MOEAs on the `LOTZ` (a special case of `Weighted LPTNO`) and `COCZ` problems, we find that $\text{MOEA}^{onebit}_{recomb}$ and $\text{MOEA}^{bitwise}_{recomb}$ are the most efficient. This supports the conclusion that recombination operators can be useful for multi-objective optimization. The analysis discloses that the recombination operator works in the studied situation by accelerating the filling of the Pareto front through recombining diverse optimal solutions found-so-far. Then, we further examine the effect of recombination for solving the multi-objective minimum spanning tree problem, which is an NP-hard problem. We derive the expected running time of $\text{MOEA}^{bitwise}_{recomb}$ with crossover probability 0.5, and by comparing with the previously analyzed $\text{MOEA}^{bitwise}$, we find that recombination can still work.

We observe that the use of the bit-wise mutation does not show advantages over the one-bit mu-

tation in the studied cases, while they have been shown to have different ability [54]. A question that will be investigated in the future is that when the bit-wise mutation leads to a better performance than the one-bit mutation. We shall analyze $\mathsf{MOEA}_{recomb}^{onebit}$ and $\mathsf{MOEA}_{recomb}^{bitwise}$ on more realistic problems with more kinds of objective functions, and we will also try to identify problem classes for which the investigated recombination operator is helpful following [41]. There are several possible directions to extend this work in the future: first, other types as well as other usages of recombination operators in MOEAs could be investigated; second, it is also interesting to study the effect of recombination in different stages of MOEAs as well as the effect of different initialization of MOEAs; third, the interaction between mutation, recombination and selection operators is an important aspect for a full characterization of MOEAs; and moreover, analyzing MOEAs on more than two objectives is an interesting topic.

7. Acknowledgments

We want to thank the associate editor and anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Science Foundation of China (61375061, 61333014), the Jiangsu Science Foundation (BK2012303), and the National Fundamental Research Program of China (2014CB340501).

References

[1] M. Abido. Multiobjective evolutionary algorithms for electric power dispatch problem. *IEEE Transactions on Evolutionary Computation*, 10(3):315–329, 2006.

[2] K. A. Andersen, K. Jörnsten, and M. Lind. On bicriterion minimal spanning trees: An approximation. *Computers & Operations Research*, 23(12):1171–1182, 1996.

[3] A. Arias-Montano, C. Coello Coello, and E. Mezura-Montes. Multiobjective evolutionary algorithms in aeronautical and aerospace engineering. *IEEE Transactions on Evolutionary Computation*, 16(5):662–694, 2012.

[4] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.

[5] G. Chen, S. Chen, W. Guo, and H. Chen. The multi-criteria minimum spanning tree problem based genetic algorithm. *Information Sciences*, 177(22):5050–5063, 2007.

[6] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, 2001.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[8] M. Dietzfelbinger, B. Naudts, C. Van Hoyweghen, and I. Wegener. The analysis of a recombinative hill-climber on H-IFF. *IEEE Transactions on Evolutionary Computation*, 7(5):417–423, 2003.

[9] B. Doerr and M. Theile. Improved analysis methods for crossover-based algorithms. In *Proceedings of the 11th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*, pages 247–254, Montreal, Canada, 2009.

[10] B. Doerr, E. Happ, and C. Klein. Crossover can provably be useful in evolutionary computation. In *Proceedings of the 10th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'08)*, pages 539–546, Atlanta, GA, 2008.

[11] B. Doerr, D. Johannsen, T. Kötzing, F. Neumann, and M. Theile. More effective crossover operators for the all-pairs shortest path problem. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN'10)*, pages 184–193, Krakow, Poland, 2010.

[12] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[13] M. Ehrgott. *Multicriteria Optimization*. Springer-Verlag, Berlin, Germany, 2005.

[14] A.-E. Eiben, P.-E. Raué, and Z. Ruttkay. Genetic algorithms with multi-parent recombination. In *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature (PPSN'94)*, pages 78–87, Jerusalem, Israel, 1994.

[15] C. Erbas, S. Cerav-Erbas, and A. D. Pimentel. Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. *IEEE Transactions on Evolutionary Computation*, 10(3):358–374, 2006.

[16] S. Fischer and I. Wegener. The one-dimensional Ising model: Mutation versus recombination. *Theoretical Computer Science*, 344(2-3):208–225, 2005.

[17] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt. Approximating covering problems by randomized search heuristics using multi-objective models. In *Proceedings of the 9th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, pages 797–804, London, UK, 2007.

[18] O. Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, pages 1918–1925, Canberra, Australia, 2003.

[19] H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52(4):209–230, 1994.

[20] T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, 117(3):553–564, 1999.

[21] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

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

[23] T. Jansen and I. Wegener. Real royal road functions-where crossover provably is essential. *Discrete Applied Mathematics*, 149(1-3):111–125, 2005.

[24] D. Johannsen, P. Kurur, and J. Lengler. Can quantum search accelerate evolutionary algorithms? In *Proceedings of the 12th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'10)*, pages 1433–1440, Portland, OR, 2010.

[25] M. P. Kleeman, B. A. Seibert, G. B. Lamont, K. M. Hopkinson, and S. R. Graham. Solving multicommodity capacitated network design problems using multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 16(4):449–471, 2012.

[26] J. D. Knowles and D. W. Corne. A comparison of encodings and algorithms for multiobjective minimum spanning tree problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'01)*, pages 544–551, Korea, 2001.

[27] T. Kötzing, D. Sudholt, and M. Theile. How crossover helps in pseudo-Boolean optimization. In *Proceedings of the 13th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'11)*, pages 989–996, Dublin, Ireland, 2011.

[28] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running time analysis of multiobjective evolutionary algorithms on a simple discrete optimization problem. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature (PPSN'02)*, pages 44–53, Birmingham, UK, 2002.

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

[30] P. K. Lehre and X. Yao. Crossover can be constructive when computing unique input output sequences. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning (SEAL'08)*, pages 595–604, Melbourne, Australia, 2008.

[31] G. Lin and X. Yao. Analysing crossover operators by search step size. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'97)*, pages 107–110, Indianapolis, IN, 1997.

[32] E. W. Mayr and C. G. Plaxton. On the spanning trees of weighted graphs. *Combinatorica*, 12(4): 433–447, 1992.

[33] F. Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181(3):1620–1629, 2007.

[34] F. Neumann and J. Reichel. Approximating minimum multicuts by evolutionary multi-objective algorithms. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN'08)*, pages 72–81, Dortmund, Germany, 2008.

[35] F. Neumann and M. Theile. How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN'10)*, pages 667–676, Krakow, Poland, 2010.

[36] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[37] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378(1):32–40, 2007.

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

[39] P. Oliveto, J. He, and X. Yao. Analysis of population-based evolutionary algorithms for the vertex cover problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'08)*, pages 1563–1570, Hong Kong, China, 2008.

[40] C. Qian, Y. Yu, and Z.-H. Zhou. An analysis on recombination in multi-objective evolutionary optimization. In *Proceedings of the 13th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'11)*, pages 2051–2058, Dublin, Ireland, 2011.

[41] C. Qian, Y. Yu, and Z.-H. Zhou. On algorithm-dependent boundary case identification for problem classes. In *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature (PPSN'12)*, pages 62–71, Taormina, Italy, 2012.

[42] G. R. Raidl and B. A. Julstrom. Edge sets: an effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, 2003.

[43] R. Ramos, S. Alonso, J. Sicilia, and C. González. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111(3):617–628, 1998.

[44] J. Richter, A. Wright, and J. Paxton. Ignoble trails-where crossover is provably harmful. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN'08)*, pages 92–101, Dortmund, Germany, 2008.

[45] G. Rudolph. On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'98)*, pages 511–516, Piscataway, NJ, 1998.

[46] G. Rudolph and A. Agapie. Convergence properties of some multi-objective evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'00)*, pages 1010–1016, Piscataway, NJ, 2000.

[47] W. Spears. *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer-Verlag, Berlin, Germany, 2000.

[48] R. Steuer. *Multiple Criteria Optimization: Theory, Computations, and Application*. John Wiley & Sons, Inc., New York, NY, 1986.

[49] D. Sudholt. Crossover is provably essential for the ising model on trees. In *Proceedings of the 7th ACM Annual Conference on Genetic and Evolutionary Computation (GECCO'05)*, pages 1161–1167, Washington, DC, 2005.

[50] R. A. Watson. Analysis of recombinative algorithms on a non-separable buildingblock problem. In *Proceedings of the 6th International Workshop on Foundations of Genetic Algorithms (FOGA'00)*, pages 69–89, San Mateo, CA, 2000.

[51] Y. Yu and Z.-H. Zhou. A new approach to estimating the expected first hitting time of evolutionary algorithms. *Artificial Intelligence*, 172(15):1809–1832, 2008.

[52] Y. Yu, C. Qian, and Z.-H. Zhou. Towards analyzing recombination operators in evolutionary search. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN'10)*, pages 144–153, Krakow, Poland, 2010.

[53] Y. Yu, C. Qian, and Z.-H. Zhou. Towards analyzing crossover operators in evolutionary search via general Markov chain switching theorem. CORR abs/1111.0907, 2011.

[54] Y. Yu, X. Yao, and Z.-H. Zhou. On the approximation ability of evolutionary optimization with application to minimum set cover. *Artificial Intelligence*, 180-181:20–33, 2012.

[55] G. Zhou and M. Gen. Genetic algorithm approach on multi-criteria minimum spanning tree problem. *European Journal of Operational Research*, 114(1):141–152, 1999.