Feedback-Based Adaptive Crossover-Rate in Evolutionary Computation

Xiaoyuan Guan[1,2], **Tianyi Yang**[1], **Chunliang Zhao**[3] and **Yuren Zhou**[4*]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]Key Laboratory of Machine Intelligence and Advanced Computing, MOE, Guangzhou, China
[3]School of Data and Science, Qingdao Universityof Science and Technology,Oingdao, China
[4]School of Software Engineering, Sun Yat-sen University, Zhuhai, China
{guanxy36, yangty9}@mail2.sysu.edu.cn,
zhaochunliang@qust.edu.cn, zhouyuren@mail.sysu.edu.cn

Abstract

We propose a novel approach to improve multi-objective evolutionary algorithms by modifying crossover operations. Our approach uses a modifiable cross distribution and virtual point to rebalance the probability distribution of all crossover options. This design reduces runtime for typical pseudo-Boolean functions. Experiments and analysis show our approach effectively optimizes bi-objective problems COCZ and LOTZ in $\Theta(n)$ time during crossover, outperforming conventional crossover multi-objective evolutionary algorithms (C-MOEA) which require $O(n \log n)$ steps. For the tri-objective problem Hierarchical-COCZ, our approach guarantees an expected runtime of $\Theta(n^2 \log n)$, while C-MOEA needs at least $\Omega(n^2 \log n)$ and at most $O(n^2 \log^2 n)$ steps.

1 Introduction

Multi-objective optimization (MOO) refers to the process of optimizing multiple conflicting objectives simultaneously. In real-world problems, it is often the case that multiple objectives need to be considered, and these objectives may compete with each other, making it challenging to find a single optimal solution. Multi-objective evolutionary optimization (MOEO) is a powerful approach that leverages the principles of evolutionary algorithms to address these complex optimization problems. Inspired by natural evolution, MOEO algorithms employ a population of candidate solutions that evolve over generations. These algorithms iteratively apply evolutionary operators such as mutation and crossover to generate a diverse set of potential solutions, known as the Pareto front or Pareto set. The Pareto front represents the trade-off relationship between the objectives, where no solution can be improved in one objective without sacrificing performance in another objective.

Crossover-enabled Evolutionary Algorithms (C-EAs) are a class of EAs that mainly use the crossover operator [Spears, 1993] to combine beneficial traits from different parent solutions. This produces novel offspring with improved fitness. The crossover operator enables C-EAs to efficiently explore complex, non-linear, non-convex solution spaces and find near-optimal or optimal solutions [Doerr *et al.*, 2012; Neumann and Theile, 2010]. By balancing exploration and exploitation, C-EAs can handle challenging optimization problems. Thus, they have been widely adopted across various fields.

Prior theoretical analysis has validated the superiority of C-EAs over other optimization methods. For single-objective problems, C-EAs outperform on numerous discrete optimization problems [Doerr *et al.*, 2012; Malalanirainy and Moraglio, 2022]. In multi-objective optimization, Neumann [Neumann and Theile, 2010] highlighted crossover's effectiveness in accelerating EAs for the all pairs shortest path problem. Furthermore, Qian et al. [Qian *et al.*, 2013] analyzed the runtime of C-MOEAs on pseudo-Boolean functions like COCZ, LOTZ, and Weighted LPTNO, finding that C-MOEAs are more efficient than mutation-only MOEAs. These studies demonstrate that crossover improves EAs for both single- and multi-objective optimization across problems, affirming the benefits of incorporating crossover. Furthermore, Huang et al. [Huang *et al.*, 2019] analyzed a simple C-MOEA called MOEA/D-C on four multi-objective problems. Their runtime analysis showed MOEA/D-C finds Pareto optimal solutions significantly faster with crossover. This provides additional evidence that crossover improves C-EAs, especially for multi-objective optimization.

However, C-EAs still face challenges. A major issue is the lack of guidance for selecting appropriate crossover points, making it difficult to generate optimized offspring. Unlike deep learning algorithms that use gradient information to identify optimal directions, C-EAs follow random directions. This reliance on randomness has drawn criticism, especially compared to gradient-based optimization.

In this paper, we propose a novel approach to reduce C-EAs' computational time. Inspired by Q-learning (a typical Reinforcement Learning) algorithm that uses a Q-value table to guide the action chosen, we propose to use a modifiable crossover distribution to guide evolution direction. Specifically, we adjust the probabilities of crossover points (to be chosen as split points) according to fitness evaluation feedback. Our method prevents repeatedly reproducing non-superior offspring, leading to reduction in expected runtime. Our contributions are summarized as following:

*Corresponding Author

1. We propose a feedback-based schedule to adjust the crossover operation, which is inspired by Q-Learning algorithm. Our method reduces the preference scores of split points that are verified to reproduce non-superior child solutions, thereby can decrease repetitive failure reproduction. Along with the design of virtual point, we retain the stability of crossover distribution and prevent probability backflow from restrained nodes to others.

2. We evaluated this approach on three Pseudo-Boolean problems: COCZ, LOTZ, and Hierarchical-COCZ.

3. We provide a theoretical analysis of expected runtime on the three considered Boolean functions.

2 Related Work

Balancing exploration and exploitation is a key challenge in evolutionary algorithms (EAs). Overemphasizing either can lead to sub-optimal solutions. Evolutionary operators drive exploration by introducing novel solutions, but their effectiveness depends on the mutation rate.

In recent years, several approaches have been proposed to dynamically adjust the mutation rate and improve evolutionary optimization. For example, Böttcher et al. [Böttcher *et al.*, 2010] analyzed the optimal mutation rate for the LEADINGONES function with size n, finding $p \approx 1.59/n$ reduced time by 16% for (1 + 1) EA. Fitness-dependent mutation rates have gained attention recently [Badkobeh *et al.*, 2014; Doerr *et al.*, 2013; Doerr *et al.*, 2021]. These adjust the schedule based on the fitness of the current population. Success-based schedules [Doerr *et al.*, 2019; Lässig and Sudholt, 2011] and learning-based schedules [Doerr *et al.*, 2016] also adapt the rate based on past performance. These studies show that self-adaptive mutation can effectively facilitate evolutionary optimization.

Pursuing more efficient self-adaptive crossover rates has long been a goal in EAs. Researchers have investigated dynamic methods to adjust the crossover probability, such as tying it to the ratio of Euclidean distances between solutions [Dong and Wu, 2009]. This promotes diversity by involving all genes. However, distinct solutions with similar distances can still produce superior but rarely mated offspring. Recent work incorporated elitism by assigning lower crossover rates to fitter individuals [Stanovov *et al.*, 2023]. While reducing alteration of better genes, this schedule may weaken exploration. Additional dynamic adaptation methods can be found in [Gong *et al.*, 2014; Pan *et al.*, 2015; Zhou *et al.*, 2017]. However, balancing exploration and exploitation remains an open challenge.

This paper proposes a new method to select crossover points that balances diversity and exploration. Unlike previous methods, our approach utilizes feedback information during optimization, successful or failed reproduction impacts the crossover distribution, avoiding repeated failures.

3 Preliminary Knowledge

3.1 Multi-Objective Evolutionary Algorithms

Multi-objective optimization problems (MOPs) are defined as

$$\max f(x) = [f_1(x), f_2(x), f_3(x), \cdots, f_m(x)] \quad (1)$$

$$s.t. \quad x \in \mathcal{X} \quad (2)$$

where f is a function consists of m objective functions, $\mathcal{X}$ is the solution space, x denotes a solution. The objective functions in multi-objective optimization are often conflicting - optimizing one erodes the others. Thus, there is no single optimal solution that simultaneously optimizes all objectives. Rather, there are trade-offs between objectives, meaning multi-objective problems lack a definitive best solution.

Instead, the commonly used criterion in multi-objective optimization is called Pareto optimality. For $x_1, x_2 \in \mathcal{X}$, we say x_1 dominates x_2 if for all $i \in \{1, 2, \cdots, m\}$, $f_i(x_1) \geq f_i(x_2)$, and the inequality can be taken for at least one index, we denote the dominating relationship as $x_1 \succ x_2$. A solution is Pareto optimal if there is no other solutions can dominate it, the set of all Pareto optimal solutions is named as Pareto optimal Set (PS), and the set of objective vectors of the Pareto optimal solutions is called Pareto Front [Van Veldhuizen *et al.*, 1998].

Multi-objective evolutionary algorithms (MOEAs) maintain a population of candidate solutions to solve multi-objective problems [Zhou *et al.*, 2011]. The algorithm iteratively mutates and crosses over selected solutions and then eliminates any dominated ones. This evolutionary process repeats until a stopping criterion is met, like a maximum number of generations. Throughout, the MOEA maintains a diverse set of non-dominated solutions representing the Pareto Front.

3.2 Reinforcement Learning

Reinforcement learning (RL) trains agents to maximize rewards through environmental interactions to achieve tasks, rather than execute predefined commands [Kaelbling *et al.*, 1996; Sutton and Barto, 2018]. A common RL algorithm, Q-learning [Clifton and Laber, 2020], suits problems with small discrete state and action spaces. The agent learns via trial-and-error to choose the best actions by updating its Q-value function:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma(\max(Q(s',a'))) - Q(s,a)), \quad (3)$$

where $Q(s,a)$ is the value of taking action a in state s. The agent gets a reward r, learns at rate α, and discounts future rewards by γ. By iteratively improving its decisions, the agent maximizes long-term reward.

In this paper, we draw inspiration from the Q-learning algorithm and develop a preference list similar to the Q-value table. Based on the feedback from evolution and evaluation, we iteratively update this list to adjust the optimization direction of the evolutionary operator.

4 Approach

In evolutionary optimization, crossover point selection is crucial, as it affects the quality and diversity of offspring. Random selection can reduce performance, as it cannot ensure

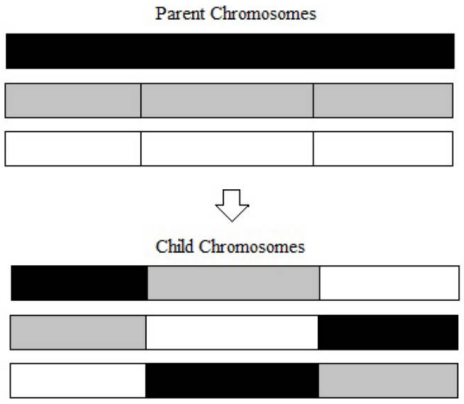

Figure 1: Illustration of multi-parent diagonal-crossover. With 3 parents, 2 nodes will be chosen as crossover points for splitting the chromosomes and reproducing 3 child solutions.

improved crossover results for each iteration. In particular, repeated random points may generate identical offspring lacking diversity. Later in optimization, random selection may also slow convergence or lead to sub-optimal solutions as the feasible space (where superior or non-dominated solutions are located) collapses. Overall, the lack of informative guidance for choosing crossover points is a key cause of poor performance in crossover-enabled algorithms.

4.1 Modifiable Cross Distribution

To improve multi-objective evolutionary optimization, we introduce C-MOEA-MCD, which utilizes a modifiable crossover distribution. Our approach borrows inspirations from Q-leaning, by introducing a preference score list $\mathbf{L}$ indicating the degree of preference for possible crossover option, each option contains not limited to single (for single-point crossover) but can expand to multiple points (for multi-point crossover). To determine the selection probability within the distribution, we use the Softmax function:

$$p_i = \frac{\exp(l_i)}{\sum_{j=1}^{n} \exp(l_j)}, \quad l_i, l_j \in \mathbf{L}, \tag{4}$$

where $l_i \in \mathbf{L}$ represents the preference score of the i^{th} cross-point.

Our approach adaptively adjusts the selection probability using evolutionary feedback, as reinforcement learning alter Q-values according to immediate rewarding. After each reproduction, we evaluate offspring quality. If the crossover option produces inferior offspring or acceptable offspring (but immediately turns out to be unacceptable choice), we decrease its preference score for future iterations by

$$l_i \leftarrow l_i - \alpha \tag{5}$$

where α denotes the update step size. Note that for the purpose to simplify theoretical analysis, we do not introduce flexible update schedule, for m objectives with problem size n studied in this paper, we consider m-parents diagonal crossover [Eiben and Smith, 2015], described as

Algorithm 1 Initialization (Phase 1)

Input: m solutions $\{x_k\}_{k=1}^m$ randomly from solution space
Output: m local optimal solutions each is corresponding to one of m objectives $\{f_k\}_{k=1}^m$

```
while not find the optimal solution do
    for each solution x_k, randomly flip chosen bit (mutation operation) to produce a child solution x'_k
    if f_k(x'_k) > f_k(x_k) then
        x_k ← x'_k
    end if
end while
```

Definition 1 (m**-Parents Diagonal-Crossover**). *Given m solutions whose problem size 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.*

We let the decay parameter $\alpha = \log \binom{n-1}{m-1}$ in Eq.(5). In real scenarios, this parameter can be customized accordingly. This adaptive feedback loop allows the algorithm to dynamically adjust the selection probabilities based on the success or failure of previous choices. This effectively balances exploration and exploitation.

4.2 Virtual Point

However, a key challenge is the potential for decreasing one solution's score to impact others. Reducing one preference score reduces the sum of exp in the denominator of Softmax function, causing undesired oscillations and instability.

To address this issue, we add a virtual point to maintain the Softmax denominator during dynamic updates, preventing negative impacts on other candidates. Specifically, we first normalize the preference values to obtain relative preference scores:

$$\mathbf{L}^0 = \{l_1^0, l_2^0, \cdots, l_{\binom{n-1}{m-1}}^0\} \tag{6}$$
$$\text{where} \quad l_i^0 = l_j^0, \quad \forall 1 \le i, j \le \binom{n-1}{m-1}.$$

The superscript 0 indicates the first generation after initialization. **Note** that if $m = 2$, the length of $\mathbf{L}$ is $n-1$, then the crossover is one-point diagonal crossover.

Denoting $G_0 = \sum_{i=1}^{n} \exp(l_i^0)$ as initialized value of the denominator in Softmax function, in later stages to keep the denominator invariant, we add an extra virtual point with score:

$$l_{\text{vir}} = \log \left[G_0 - \sum_{i=1}^{n} \exp(l_i^t) \right].$$

This prevents the phenomenon that alter one preference score will affect other probabilities. When sampling, the virtual point requires no evaluation and is skipped, avoiding runtime cost.

Algorithm 2 C-MOEA-MCD

Require: objective function $f = (f_1, f_2, \cdots, f_m)$; crossover threshold p_r.
Initialize: preference score string $\mathbf{L} = \{l_1, l_2, \cdots, l_{\binom{n-1}{m-1}}\}$
Initialize: step size α, generation index $t = 0$

Initialization population P (Phase 1)
compute the crossover distribution $p = \text{Softmax}(\mathbf{L})$
while not find all Pareto solutions **do**
  use the m local optimal solutions (derived from Algorithm 1) as parent solution
  sampling $r \in$ Uniform[0,1]
  if $r < p_r$ **then**
    sampling $m-1$ cross points according to p
    if not contains the virtual point **then**
      reproduce m child solutions by applying $m-1$ parent diagonal crossover
    else
      skip to next generation
    end if
    record the crossover points τ in current cycle
  else
    reproduce m child solutions by flipping selected bit on each parent solution
  end if
  for each child solution s **do**
    if $\nexists z \in P$ s.t. $z \succeq s$ **then**
      $P = (P \backslash \{z \in P | s \succ z\}) \cup \{s\}$
    end if
  end for
  Update $l_{\tau_t} = l_{\tau_t} - \alpha$, for $\tau_t \in \tau$
  if $t \neq 0$: **then**
    $l_{\text{vir}} = \log\left[G_0 - \sum \exp(l_i^t)\right]$
  else
    $l_{\text{vir}} = -\infty$
  end if
  $p = \text{Softmax}(\mathbf{L})$, $t = t + 1$
end while
return P.

To make the initial population diverse enough, we use the same initialization setting as in [Qian *et al.*, 2013], the initialization process is described in Algorithm 1. The whole Pseudo-code is stated in Algorithm 2.

5 Runtime Analysis and Empirical Study

Runtime analysis is critical for understanding the efficiency of EAs. Here, we compare the expected runtime between C-MOEA-MCD and C-MOEA on the COCZ, LOTZ and Hierarchical-COCZ problems.

5.1 Analyzed Problem

Definition 2. *The COCZ problem of size n is to find n-bits binary strings which maximize the function*

$$f(x) = (\sum_{i=1}^{n} x_i, \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} (1 - x_i)),$$

where n is even and x_i denotes the i-th bit of $x \in \{0,1\}^n$.

The COCZ problem maximizes two linear functions: the first maximizes overall 1-bits (the same as the **OneMax** problem [Droste *et al.*, 2002]), while the second maximizes 1-bits in the first half and 0-bits in the second half. The Pareto optimal solutions are:

$$\{1^{\frac{n}{2}} 1^i 0^{\frac{n}{2}-i} | 0 \leq i \leq \frac{n}{2}\},$$

the size of the set is $2^{\frac{n}{2}}$, the Pareto front is

$$\{(n, \frac{n}{2}), (n-1, \frac{n}{2}+1), \cdots, (\frac{n}{2}, n)\}.$$

Definition 3. *The LOTZ problem of size n is to find n-bits binary strings which maximize*

$$f(x) = (\sum_{i=1}^{n} \prod_{j=1}^{i} x_j, \sum_{i=1}^{n} \prod_{j=i}^{n} (1 - x_j)),$$

where x_i denotes the i-th bit of $x \in \{0,1\}^n$.

For LOTZ, the first objective maximizes leading 1-bits (same as the **LeadingOnes** problem), and the second maximizes trailing 0-bits. The Pareto front can be represented as:

$$\{(0, n), (1, n-1), \cdots, (n, 0)\},$$

and the set of Pareto optimal solutions are

$$\{0^n, 0^{n-1}1, \cdots, 1^{n-1}0, 1^n\}.$$

A more sophisticated tri-objective problem is considered in this study, which is defined as

Definition 4 (Hierarchical-COCZ)**.**

$$H_1\text{-}COCZ(x) = (f_1(x), f_2(x)), \tag{7}$$
$$H_2\text{-}COCZ(x) = (H_1\text{-}COCZ(x), f_3(x)), \tag{8}$$

where

$$f_1(x) = \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{2n/3} x_i + \sum_{i=2n/3+1}^{n} (1 - x_i),$$

$$f_2(x) = \sum_{i=1}^{n/2} x_i + \sum_{\substack{i \in (n/2, 2n/3] \\ \cup (5n/6, n]}} (1 - x_i) + \sum_{i=2n/3+1}^{5n/6} x_i,$$

$$f_3(x) = \sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{5n/6} (1 - x_i) + \sum_{i=5n/6+1}^{n} x_i.$$

The objective of H_1-COCZ is to optimize f_1 and f_2 simultaneously, the second is to optimize all the three f functions.

5.2 Runtime Analysis on COCZ

Runtime Analysis on Phase 1

The initialization of C-MOEA and C-MOEA-MCD is to optimize the two objectives of COCZ, which have the same structure as the **OneMax** problem, by Theorem 11 in [Johannsen *et al.*, 2010], it is known that the running time of C-MOEA and C-MOEA-MCD on Phase 1 are both $\Theta(n \log n)$.

Runtime Analysis on Phase 2

According to Theorem 6 in [Qian *et al.*, 2013], the expected running time of C-MOEA on Phase 2 of COCZ is $\Omega(n \log n)$.

Our design balances the probabilities of selecting acceptable (which reproduce better offspring) and unacceptable (reproduce offsping not better) crossover points. Denoting the probabilities at generation t as p_{a}^t and p_{un}^t, respectively, the ratio is $r^t = p_{\text{un}}^t / p_{\text{a}}^t$. To reach the next generation by reproducing better offspring, the expected fitness evaluations for a given r_t is $\lceil r_t \rceil + 1$ due to Proposition 1. Omitting subscripts, the key is the rate between p_{un} and p_{a} at the same step.

Proposition 1. *Given the ratio $r = \frac{p_{un}}{p_a}$, the expected runtime from generation to the next is $\lceil r \rceil + 1$.*

Proof 1 (Proof for Proposition 1)**.** *Given the ratio r, and the probability for acceptable reproduction is p_a, the expected runtime for acceptable reproduction is*

$$\frac{1}{p_a} = \frac{p_a + p_{un} + p_{vir}}{p_a} \tag{9}$$

$$= 1 + r + \frac{p_{vir}}{p_a}. \tag{10}$$

Since the virtual point is always skipped directly, thereby no evaluation calls are made, thus the expected runtime for acceptable crossover operation is $\lceil r \rceil + 1$.

Theorem 1. *For COCZ with problem size n and $m = 2$, if let $\alpha = \log(\binom{n-1}{m-1}) = \log(n-1)$, the expected runtime of C-MOEA-MCD at the crossover procedure is $\Theta(n)$ with probability almost 1.*

Due to the fact that, virtual point is always be skipped thereby contributing no fitness evaluation, considering the ratio r in expected runtime estimation is enough. We will show that, r is bounded by 2 with probability $1 - (\frac{1}{2})^{n/4-2}$ where n denotes the problem size.

Proof 2 (Proof for Theorem 1)**.** *After Phase 1, the initialization procedure (see Algorithm 1), the two local optimal solutions are $\{1^n, 1^{\frac{n}{2}}0^{\frac{n}{2}}\}$. For COCZ problem, the crossover used is one-point crossover due to the fact that COCZ problem is a bi-objective problem. There are $n-1$ optional crossover points. We have $\alpha = \log(n-1)$, an probabilistic updating formula after one selection is as follows:*

$$p_i' = \frac{\exp(l_i - \log(n-1))}{G_0} = p_i/(n-1). \tag{11}$$

At the beginning of crossover procedure, the initialized preference score list is $\mathbf{L}^0 = [l_1^0, l_2^0, \cdots, l_{n-1}^0]$, with identical elements. Each point has the same probability $\frac{1}{n-1}$ to be selected. The ratio $r = \frac{n/2}{n/2-1} \approx 1$.

Before optimization ends, consider an extreme case where the algorithm only selects acceptable crossover points, continuously increasing ratio r. Denoting this event as A, the probability is:

$$P(A) = \binom{\frac{n}{2}-1}{Q} (\frac{1}{n-1})^{\frac{n}{2}-1-Q} \leq (\frac{1}{2})^{\frac{n}{2}-1-Q}, \tag{12}$$

where $Q \geq 1$ denotes the number of acceptable crossover points that have never been selected. In this case, the size of r can be expressed as a formula that involves Q by

$$r = [\frac{n}{2} \times \frac{1}{n-1} + (\frac{n}{2} - 1 - Q) \times (\frac{1}{n-1})^2] / \frac{Q}{n-1} \tag{13}$$

$$= [\frac{n}{2} + (\frac{n}{2} - 1 - Q) \times (\frac{1}{n-1})]/Q \tag{14}$$

This implies that the probability when event A occurs, $P(r \geq \frac{n}{2Q}, A)$, is less than $(\frac{1}{2})^{\frac{n}{2}-1-Q}$. If let $Q = \frac{n}{4}$, we have

$$P(r \geq 2, A) \leq (\frac{1}{2})^{\frac{n}{4}-1}. \tag{15}$$

With larger problem size n, the probability of $r \geq 2$ almost equals to zero.

Based on the previous discussion of the extreme case A, we will now consider a more general event B, till the situation that there exists Q acceptable points have not yet selected, the algorithm selects each time m_1 unacceptable points. The probability of event B is

$$\binom{\frac{n}{2}-1}{Q} (\frac{1}{n-1})^{\frac{n}{2}-1-Q} \times$$

$$\sum_{x=0}^{K} \binom{\frac{n}{2}}{m_1 - x} (\frac{1}{n-1})^{m_1 - x} \times \binom{\frac{n}{2}-1-Q}{x} (\frac{1}{(n-1)^2})^x$$

$$\leq \binom{\frac{n}{2}-1}{Q} (\frac{1}{n-1})^{\frac{n}{2}-1-Q} \times \frac{1}{2^{m_1}}, \tag{16}$$

where $K = \min(m_1, \frac{n}{2} - 1 - Q)$. Note that the ratio r expressed in Equation 13 is the upper bound of each event. Thereby we have

$$P(r \geq 2, B) \leq (\frac{1}{2})^{\frac{n}{4}-1} \times \frac{1}{2^{m_1}}. \tag{17}$$

The probability of general events (which contains more unacceptable trails) decreases by factor $\frac{1}{\Psi}$ ($\Psi \geq 2$), we can give an upper bound of the joint probability as

$$P(r \geq 2) = \sum_{event} P(r \geq 2, event) \tag{18}$$

$$\leq 2 \times (\frac{1}{2})^{\frac{n}{4}-1} = (\frac{1}{2})^{\frac{n}{4}-2}. \tag{19}$$

Therefore, the ratio r is bounded by 2 across the entire optimization. By ensuring this, we can control the expected number of fitness evaluations required from one generation to the next to be under $\lceil 2 \rceil + 1 = 3$. Consequently, the expected runtime of C-MOEA-MCD during the crossover procedure is $\Theta(n)$ with probability 1. This concludes our proof.

We recorded the time series of the changes in r during the experiments. The green area represents the range of fluctuations in r during the crossover phase of 500 experiments on COCZ, while the yellow area represents the corresponding values for LOTZ. The solid lines represent the mean curves for each experiment.

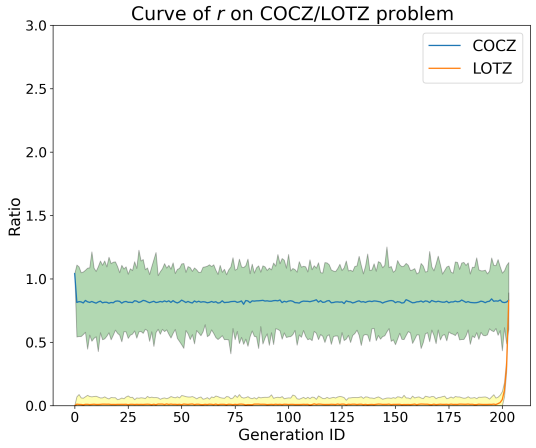

Figure 2: The curve ratio r changes when optimizing COCZ (green shadow area represents the empirical range of r) / LOTZ (yellow shadow area records the empirical range of r) with problem size 200, $\alpha = \log(199)$. We repeat each experiment for 500 runs, the line charts show the average trend. Empirically, the ratio is always below 2, which supports our analysis.

5.3 Runtime Analysis on LOTZ

Runtime Analysis on Phase 1

The initialization of C-MOEA and C-MOEA-MCD is to optimize the two objectives of LOTZ, which have the same structure as the **LeadingOnes** problem, by Theorem 13 in [Johannsen *et al.*, 2010], it is known that the running time of C-MOEA and C-MOEA-MCD on Phase 1 are both $\Theta(n^2)$.

Runtime Analysis on Phase 2

According to [Qian *et al.*, 2013], the expected running time of C-MOEA on Phase 2 of LOTZ is at most $O(n \log n)$.

Theorem 2. *On LOTZ, the expected runtime of C-MOEA-MCD at the Phase 2 is $\Theta(n)$ with a probability of 1.*

Proof idea is same as on the COCZ problem.

Proof 3 (Proof for Theorem 2)**.** *For LOTZ problem with problem size n, there are $n-1$ candidate crossover points, and the crossover operator is one-point crossover too. As discussed in the proof of COCZ problem, given the ratio r , from current generation to next generation, the expected number of fitness evaluations is $\lceil r \rceil + 1$.*

After Phase 1, each crossover point has the same probability of $\frac{1}{n-1}$ to be selected, and the two initialized Pareto optimal solutions are $\{1^n, 0^n\}$. By now, it is easy to obtain that $r = 0$.

The probability of unacceptable points in this problem is growing by updating after performing an acceptable reproduction. Specifically, each time an acceptable reproduction operation is performed, it will convert an acceptable crossover point into an unacceptable one. Meanwhile, the probability of unacceptable points increases by at most $\frac{1}{(n-1)^2}$, and the probability of acceptable points decreases by $\frac{1}{n-1}$. Consider a general case, there exists Q acceptable crossover points that have never been selected, then the probability of unacceptable crossover points p_{un} is at most $\frac{1}{(n-1)^2} \times (n-1-Q)$, and $p_a = \frac{Q}{n-1}$, the ratio r can be estimated by

$$r = \frac{p_{un}}{p_a} \leq \frac{n-1-Q}{Q} \times \frac{1}{n-1} \leq 1. \tag{20}$$

Therefore, each generation can contribute at most $\lceil 1 \rceil + 1 = 2$ fitness evaluation calls, the expected runtime of the crossover procedure on LOTZ is $\Theta(n)$ with a probability of 1.

5.4 Runtime Analysis on Hierarchical-COCZ

We compare our method with C-MOEA which adopt with static crossover rate. On Phase 2, our approach reduces the expected runtime from $O(n \log n)$ to $\Theta(n)$. However, when taking the runtime of Phase 1 into account, the complexity of the whole optimization remained at the scale of $\Theta(n \log n)$. Our method does not improve the efficiency significantly on COCZ and LOTZ.

To highlight the effectiveness of our design, we considered the more complex problem, Hierarchical-COCZ. We separate the whole optimization procedure into 3 steps:

1. The first step is the initialization process, i.e., employing mutation only to get the local optimal solution for each function.
2. Secondly, applying C-MOEA-MCD to fulfill the Pareto front of H_1-COCZ(x). We denote the corresponding Pareto set as P_1.
3. Finally, select a non-dominated solution randomly from P_1, forming the parent group with the local optimal solution of f_3, then use one-point crossover with our design to fulfill the Pareto front for this dimension reduced problem, loops until the whole Parent front is fulfilled.

First of all, the initialized optimal solutions for the three single objective functions are $1^{2n/3}0^{n/3}$, $1^{n/2}0^{n/6}1^{n/6}0^{n/6}$, and $1^{n/2}0^{n/3}1^{n/6}$ respectively.

The Expected Runtime of C-MOEA

The overall runtime consists of the following three parts:

1. First, the initialization process contributes $\Theta(n \log(n))$ calls for each function.
2. Second, denote the initialized optimal solution for f_3 as x_u, for H_1-COCZ, the expected runtime of a general C-MOEA is at least $\Omega(n)$ and at most $O(n \log n)$.
3. Finally, the size of P_1 is $n/3$, the probability that a solution $y \in P_1$ will be chosen is $3/n$. To fulfill the Pareto front between y and x_u, a simple C-MOEA needs on average at least $\Omega(n)$ and at most $O(n \log n)$ calls for fitness evaluation [Qian *et al.*, 2013]. To fulfill the whole Pareto front, with the probability $3/n$ to select a solution in P_1, the expected number of trials to collect all the solutions in P_1 is

$$\sum 1 + \frac{n/3}{n/3-1} + \cdots + n/3 = \Theta(n \log n).$$

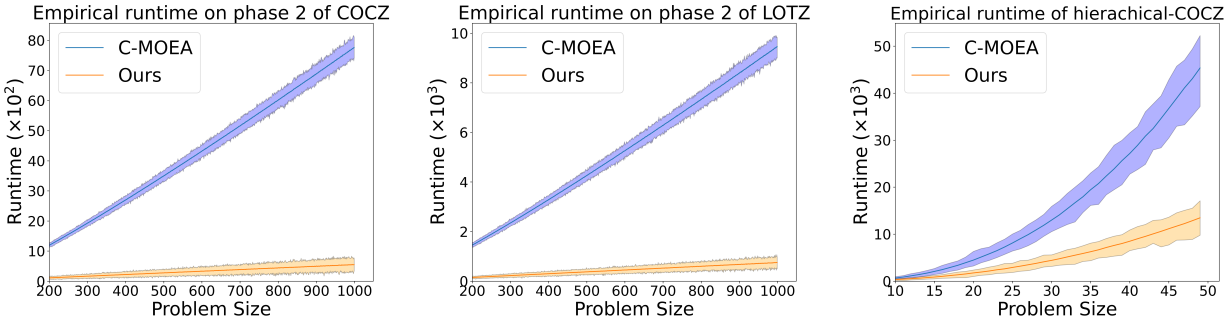

Figure 3: Empirical experiments on the three studied problems: COCZ, LOTZ, and Hierarchical-COCZ. On each problem size, we repeat run the algorithms 500 times, shade area with color indicate the empirical runtime range recorded.

Thus, the overall expected runtime of a simple C-MOEA on Hierarchical-COCZ is at least

$$\Theta(n \log n) + \Omega(n) + \Omega(n) \cdot \Theta(n \log n) = \Omega(n^2 \log n).$$

and at most

$$\begin{aligned}&\Theta(n \log n) + O(n \log n) + O(n \log n) \cdot \Theta(n \log n)\\ =&O(n^2 \log^2 n).\end{aligned}$$

The Expected Runtime of C-MOEA-MCD

The overall runtime also consists of three parts:

1. First, as in simple C-MOEA, the initialization process contributes $\Theta(n \log(n))$ fitness evaluations for each function.
2. Second, according to our analysis on the original COCZ problem, to fulfill the Pareto front of H_1-COCZ, it needs on average $\Theta(n)$ calls for evaluation.
3. Finally, to fulfill the Pareto front between each solution $y \in P_1$ and x_u, our approach needs on average $\Theta(n)$ calls for fitness evaluation. The expected number of trials to collect all the solutions in P_1 is $\Theta(n \log n)$.

Thus, the overall runtime on Hierarchical-COCZ is

$$\Theta(n \log n) + \Theta(n) + \Theta(n) \times \Theta(n \log n) = \Theta(n^2 \log n).$$

5.5 Empirical Study

We conduct the corresponding experiments for the three problems studied to verify the theoretical analysis. For all the problems, denote by n the problem size, we let $\alpha = \log(n - 1)$, $p_r = 0.5$, and we initialize the preference score list for each (dimension reduced) bi-objective string as 1^{n-1}. On each problem size, we repeat 500 times and record the average runtime as the estimation of the expected runtime, the empirical results are shown in Figure 3. We can find that both the curves of C-MOEA-MCD on the three problems are apparently lower, and notably, on the crossover procedure of COCZ and LOTZ problem, the runtime increases with the problem size linearly, which coincides with our theoretical analysis.

6 Conclusion

We propose a simple yet effective method combining crossover distribution modification and virtual points to balance reproduction probability. It utilizes interactive information to regulate selection while ensuring diversity, and prevents undesired probability backflow. This design significantly reduces computational time per generation.

Our approach provides insights into using immediate feedback to adjust evolutionary direction and overcome obstacles. On complex problems it risks probabilities of temporarily unacceptable actions approaching zero, getting trapped in local optima. Resetting the distribution modification can address this.

Overall, we hope our method and analysis inspire efficient evolutionary algorithms leveraging interactive information to steer optimization. Further exploration of generalized algorithms is needed to fully address local optima challenges.

References

[Badkobeh *et al.*, 2014] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13*, 2014.

[Böttcher *et al.*, 2010] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the leadingones problem. In *International Conference on Parallel Problem Solving from Nature*, 2010.

[Clifton and Laber, 2020] Jesse Clifton and Eric Laber. Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 2020.

[Doerr *et al.*, 2012] Benjamin Doerr, Edda Happ, and Christian Klein. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, 425:17–33, 2012.

[Doerr *et al.*, 2013] Benjamin Doerr, Carola Doerr, and Franziska Ebel. Lessons from the black-box: Fast crossover-based genetic algorithms. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, 2013.

[Doerr *et al.*, 2016] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016.

[Doerr *et al.*, 2019] Benjamin Doerr, Carola Doerr, and Johannes Lengler. Self-adjusting mutation rates with provably optimal success rules. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[Doerr *et al.*, 2021] Benjamin Doerr, Carsten Witt, and Jing Yang. Runtime analysis for self-adaptive mutation rates. *Algorithmica*, 2021.

[Dong and Wu, 2009] Min Dong and Yan Wu. Dynamic crossover and mutation genetic algorithm based on expansion sampling. In *International Conference on Artificial Intelligence and Computational Intelligence*. Springer, 2009.

[Droste *et al.*, 2002] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+ 1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[Eiben and Smith, 2015] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[Gong *et al.*, 2014] Wenyin Gong, Zhihua Cai, and Yang Wang. Repairing the crossover rate in adaptive differential evolution. *Applied Soft Computing*, 2014.

[Huang *et al.*, 2019] Zhengxin Huang, Yuren Zhou, Zefeng Chen, and Xiaoyu He. Running time analysis of moea/d with crossover on discrete optimization problem. In *AAAI*, 2019.

[Johannsen *et al.*, 2010] Daniel Johannsen, Piyush P Kurur, and Johannes Lengler. Can quantum search accelerate evolutionary algorithms? In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1433–1440, 2010.

[Kaelbling *et al.*, 1996] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 1996.

[Lässig and Sudholt, 2011] Jörg Lässig and Dirk Sudholt. Adaptive population models for offspring populations and parallel evolutionary algorithms. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*, 2011.

[Malalanirainy and Moraglio, 2022] Tina Malalanirainy and Alberto Moraglio. Runtime analysis of convex evolutionary search algorithm with standard crossover. *Swarm and Evolutionary Computation*, 2022.

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

[Pan *et al.*, 2015] Zhibin Pan, Junjie Wu, Zhaoqi Gao, and Jinghuai Gao. Adaptive differential evolution by adjusting subcomponent crossover rate for high-dimensional waveform inversion. *IEEE Geoscience and Remote Sensing Letters*, 2015.

[Qian *et al.*, 2013] Chao Qian, Yang Yu, and Zhi-Hua Zhou. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 2013.

[Spears, 1993] William M Spears. Crossover or mutation? In *Foundations of genetic algorithms*, volume 2, pages 221–237. Elsevier, 1993.

[Stanovov *et al.*, 2023] Vladimir Stanovov, Lev Kazakovtsev, and Eugene Semenkin. Crossover rate sorting in adaptive differential evolution. *Algorithms*, 2023.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Van Veldhuizen *et al.*, 1998] David A Van Veldhuizen, Gary B Lamont, et al. Evolutionary computation and convergence to a pareto front. In *Late breaking papers at the genetic programming 1998 conference*, 1998.

[Zhou *et al.*, 2011] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation*, 2011.

[Zhou *et al.*, 2017] Yin-Zhi Zhou, Wen-Chao Yi, Liang Gao, and Xin-Yu Li. Adaptive differential evolution with sorting crossover rate for continuous optimization problems. *IEEE transactions on cybernetics*, 2017.