Maintaining Diversity Provably Helps in Evolutionary Multimodal Optimization

Shengjie Ren[1] , **Zhijia Qiu**[1] , **Chao Bian**[1] , **Miqing Li**[2] and **Chao Qian**[1]

[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
[2]School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.
shengjieren36@gmail.com, {qiuzj, bianc, qianc}@lamda.nju.edu.cn, m.li.8@bham.ac.uk

Abstract

In the real world, there exist a class of optimization problems that multiple (local) optimal solutions in the solution space correspond to a single point in the objective space. In this paper, we theoretically show that for such multimodal problems, a simple method that considers the diversity of solutions in the solution space can benefit the search in evolutionary algorithms (EAs). Specifically, we prove that the proposed method, working with crossover, can help enhance the exploration, leading to polynomial or even exponential acceleration on the expected running time. This result is derived by rigorous running time analysis in both single-objective and multi-objective scenarios, including $(\mu + 1)$-GA solving the widely studied single-objective problem, Jump, and NSGA-II and SMS-EMOA (two well-established multi-objective EAs) solving the widely studied bi-objective problem, OneJumpZeroJump. Experiments are also conducted to validate the theoretical results. We hope that our results may encourage the exploration of diversity maintenance in the solution space for multi-objective optimization, where existing EAs usually only consider the diversity in the objective space and can easily be trapped in local optima.

1 Introduction

In the real world, there exist a class of optimization problems where multiple (local) optimal solutions in the solution space correspond to a single point in the objective space, such as in truss structure optimization [Reintjes, 2022], space mission design [Schutze *et al.*, 2011], rocket engine design [Kudo *et al.*, 2011], and functional brain imaging [Sebag *et al.*, 2005]. Such problems belong to multi-modal optimization problems (MMOPs). Note that there are usually two types of MMOPs [Deb and Saha, 2010; Preuss *et al.*, 2021]: one with multiple local optima (typically with different objective function values) and the other with multiple (local) optimal solutions having identical objective function value. In this work, MMOPs we refer to is the latter.

Evolutionary algorithms (EAs) are a kind of randomized heuristic optimization algorithms, inspired by natural evolution. They maintain a set of solutions (called a population), and iteratively improve the population by generating new offspring solutions and replacing inferior ones. EAs, due to their population-based nature and the ability of performing the search globally, have become a popular tool to solve MMOPs [Das *et al.*, 2011; Cheng *et al.*, 2018; Tian *et al.*, 2021; Pan *et al.*, 2023]. Since 1990's, to deal with MMOPs, many ideas have been proposed to improve EAs' ability, including using niching technique [Pétrowski, 1996; Shir, 2012; Preuss, 2015], designing novel reproduction operators and population update methods [Kennedy, 2010; Storn and Price, 1997; Liang *et al.*, 2024], and transforming an MMOP into a multiobjective optimization problem [Wessing *et al.*, 2013; Cheng *et al.*, 2018; Liu *et al.*, 2022].

In contrast to algorithm design of EAs, theoretical studies (e.g., running time complexity analyses) in the area are relatively underdeveloped. This is mainly because sophisticated behaviors of EAs can make theoretical analysis rather difficult. Nevertheless, since the 2000s, there is increasingly interest in rigorously analyzing EAs. Some studies worked on running time analysis tools, such as drift analysis [He and Yao, 2001; Oliveto and Witt, 2011; Doerr *et al.*, 2012], fitness level method [Wegener, 2002; Sudholt, 2013; Dang and Lehre, 2015], and switch analysis [Yu *et al.*, 2015; Yu and Qian, 2015; Qian *et al.*, 2016; Bian *et al.*, 2018]. Some other studies devoted themselves to the analysis of various EAs for solving synthetic or combinatorial optimization problems [Neumann and Witt, 2010; Auger and Doerr, 2011; Zhou *et al.*, 2019; Doerr and Neumann, 2020]. These theoretical results can help understand the mechanisms of EAs and inspire the design of more efficient EAs in practice.

In this paper, we consider running time analysis of EAs for MMOPs. We analytically show that when solving MMOPs, considering the diversity of the solutions in the solution space can be beneficial for the search of EAs. Specifically, we propose a simple method to deal with multiple (local) optimal solutions via comparing their crowdedness in the solution space. We consider both single-objective and multi-objective optimization cases, and incorporate the proposed method into a classical single-objective EA, $(\mu+1)$-GA, and two well-established multi-objective EAs (MOEAs), NSGA-II and SMS-EMOA.

For the single-objective case, we prove that the expected running time of $(\mu + 1)$-GA using the proposed method

for solving (i.e., finding the global optimum) Jump [Droste *et al.*, 2002], a widely studied single-objective problem, is $O(\mu^2 4^k + \mu n \log n + n\sqrt{k}(\mu \log \mu + \log n))$, where n is the problem size, and $k \leq n/4$, a parameter of Jump. Since the expected running time of the original $(\mu+1)$-GA for solving Jump is $O((40e\mu(\mu+1))^k(\frac{1}{\mu-1})^{k-1})\frac{10e}{9}\frac{n^k}{(k-1)!} + n\sqrt{k}(\mu \log \mu + \log n) = O(\mu\sqrt{k}(40e^2\mu n/k)^k)$ [Doerr and Qu, 2023c], our method can bring a polynomial acceleration when k is a constant, e.g., $k = 4$, and bring an exponential acceleration when k is large, e.g., $k = n^{1/4}$.

For the multi-objective case, we prove that the expected running time of NSGA-II and SMS-EMOA (with crossover) using the proposed method for solving (i.e., finding the whole Pareto front) OneJumpZeroJump [Doerr and Zheng, 2021], a widely studied bi-objective problem, is both $O(\mu^2 4^k + \mu n \log n)$, where μ is the population size, n is the problem size, and $k \leq n/4$, a parameter of OneJumpZeroJump. Note that the expected running time of the original NSGA-II [Doerr and Qu, 2023c] and SMS-EMOA (analyzed in this paper) for solving OneJumpZeroJump is $O(\mu^2\sqrt{k}(Cn/k)^k)$, where C is a constant and $k = o(\sqrt{n})$. Thus, our method can also bring a polynomial acceleration when k is a constant, and bring an exponential acceleration when k is large, e.g., $k = n^{1/4}$. The main reason for the acceleration resulting from our method is that it can explicitly preserve solutions with high diversity, and thus can help enhance the exploration by working with crossover. Experiments are also conducted to validate the theoretical results. In addition, it is worth mentioning that since the expected running time of SMS-EMOA (without crossover) for solving OneJumpZeroJump is $O(\mu n^k)$ [Bian *et al.*, 2023], our results also contribute to the theoretical understanding of the effectiveness of using crossover in SMS-EMOA.

2 Preliminaries

In this section, we first introduce multimodal optimization and the considered benchmark problems, Jump and OneJumpZeroJump, followed by the description of the studied algorithms, $(\mu+1)$-GA, NSGA-II and SMS-EMOA.

2.1 Multimodal Optimization

Multimodal optimization refers to the optimization scenario that multiple (local) optimal solutions in the solution space have the same objective value, which arises in many real-world applications, e.g., flow shop scheduling [Basseur *et al.*, 2002], rocket engine optimization [Kudo *et al.*, 2011], and architecture design [Tian *et al.*, 2021]. In this paper, we study two pseudo-Boolean (i.e., the solution space $\mathcal{X} = \{0,1\}^n$) multi-modal optimization problems (MMOPs), Jump and OneJumpZeroJump, which have been widely used in EAs' theoretical analyses [Droste *et al.*, 2002; Doerr and Zheng, 2021; Bian *et al.*, 2023; Doerr and Qu, 2023a; Doerr and Qu, 2023b; Doerr and Qu, 2023c; Lu *et al.*, 2024].

The Jump problem as presented in Definition 1, is to maximize the number of 1-bits of a solution, except for a valley around 1^n (the solution with all 1-bits) where the number of 1-bits should be minimized. Its optimal solution is 1^n with function value $n+k$. We can see that 1^n is global optimal, and any solution with $(n-k)$ 1-bits is local optimal with objective value n; thus, the Jump problem is multi-modal, since multiple local optimal solutions correspond to an objective value.

Definition 1 ([Droste *et al.*, 2002])**.** *The Jump problem is to find an n bits binary string which maximizes*

$$f(\boldsymbol{x}) = \begin{cases} k + |\boldsymbol{x}|_1, & \text{if } |\boldsymbol{x}|_1 \leq n-k \text{ or } \boldsymbol{x} = 1^n, \\ n - |\boldsymbol{x}|_1, & \text{else}, \end{cases}$$

where $k \in \mathbb{Z} \wedge 2 \leq k < n$, and $|\boldsymbol{x}|_1$ denotes the number of 1-bits in $\boldsymbol{x}$.

The OneJumpZeroJump problem is a bi-objective counterpart of the Jump problem. For multi-objective optimization, several objective functions (which are usually conflicting) need to be optimized simultaneously, and thus there does not exist a canonical complete order in the solution space $\mathcal{X}$. And usually the Pareto domination relation is used (Definition 2) to compare solutions. A solution is *Pareto optimal* if it is not dominated by any other solution in $\mathcal{X}$, and the set of objective vectors of all the Pareto optimal solutions is called the *Pareto front*. The goal of multi-objective optimization is to find the Pareto front or its good approximation.

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

- *$\boldsymbol{x}$* weakly dominates *$\boldsymbol{y}$ (denoted as $\boldsymbol{x} \succeq \boldsymbol{y}$) if for any $1 \leq i \leq m, f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{y})$;*
- *$\boldsymbol{x}$* dominates *$\boldsymbol{y}$ (denoted as $\boldsymbol{x} \succ \boldsymbol{y}$) if $\boldsymbol{x} \succeq \boldsymbol{y}$ and $f_i(\boldsymbol{x}) > f_i(\boldsymbol{y})$ for some i;*
- *$\boldsymbol{x}$ and $\boldsymbol{y}$ are* incomparable *if neither $\boldsymbol{x} \succeq \boldsymbol{y}$ nor $\boldsymbol{y} \succeq \boldsymbol{x}$.*

As presented in Definition 3 below, the first objective of OneJumpZeroJump is the same as the Jump problem, while the second objective is isomorphic to the first one, with the roles of 1-bits and 0-bits exchanged. The Pareto front of the OneJumpZeroJump problem is $\{(a, n+2k-a) \mid a \in [2k..n] \cup \{k, n+k\}\}$, whose size is $n-2k+3$, and the Pareto optimal solution corresponding to $(a, n+2k-a)$, $a \in [2k..n] \cup \{k, n+k\}$, is any solution with $(a-k)$ 1-bits. Note that we use $[l..r]$ (where $l, r \in \mathbb{Z}, l \leq r$) to denote the set $\{l, l+1, \ldots, r\}$ of integers throughout the paper. Since any solution with $(a-k)$ 1-bits (where $a \in [2k..n]$) is Pareto optimal with objective vector $(a, n+2k-a)$, the OneJumpZeroJump problem is multimodal.

Definition 3 ([Doerr and Zheng, 2021])**.** *The OneJumpZeroJump problem is to find n bits binary strings which maximize*

$$f_1(\boldsymbol{x}) = \begin{cases} k + |\boldsymbol{x}|_1, & \text{if } |\boldsymbol{x}|_1 \leq n-k \text{ or } \boldsymbol{x} = 1^n, \\ n - |\boldsymbol{x}|_1, & \text{else}, \end{cases}$$

$$f_2(\boldsymbol{x}) = \begin{cases} k + |\boldsymbol{x}|_0, & \text{if } |\boldsymbol{x}|_0 \leq n-k \text{ or } \boldsymbol{x} = 0^n, \\ n - |\boldsymbol{x}|_0, & \text{else}, \end{cases}$$

where $k \in \mathbb{Z} \wedge 2 \leq k < n/2$, and $|\boldsymbol{x}|_1$ and $|\boldsymbol{x}|_0$ denote the number of 1-bits and 0-bits in $\boldsymbol{x}$, respectively.

2.2 Evolutionary Algorithms

The $(\mu + 1)$-GA algorithm is a classical genetic algorithm for solving single-objective optimization problems. The algorithm starts from an initial population of μ random solutions. In each iteration, it selects a solution $\boldsymbol{x}$ from the population P randomly as the parent solution. Then, with probability p_c, it selects another solution $\boldsymbol{y}$ and applies uniform crossover on $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate an offspring solution $\boldsymbol{x}'$; otherwise, $\boldsymbol{x}'$ is set as the copy of $\boldsymbol{x}$. The uniform crossover operator exchanges each bit of two solutions independently with probability $1/2$. Note that uniform crossover actually produces two solutions, but we only pick the first one. Afterwards, bit-wise mutation is applied, which flips each bit of a solution independently with probability $1/n$, on $\boldsymbol{x}'$ to generate one offspring solution. Then, one solution with the worst objective value is removed. The complete procedure is provided in the supplementary material due to space limitation.

Now we introduce two well-established MOEAs, NSGA-II [Deb *et al.*, 2002] and SMS-EMOA [Beume *et al.*, 2007]. The NSGA-II algorithm in Algorithm 1 adopts a $(\mu + \mu)$ steady state mode. It starts from an initial population of μ random solutions (line 1). In each iteration, it selects μ solutions from the current population to form the parent population Q (line 4). Then, for each pair of the solutions in Q, uniform crossover and bit-wise mutation operators are applied sequentially to generate two offspring solutions $\boldsymbol{x}''$ and $\boldsymbol{y}''$ (lines 6–12), where the uniform crossover operator is applied with probability p_c. After μ offspring solutions have been generated in P', the solutions in $P \cup P'$ are partitioned into non-dominated sets $R_1, \ldots, R_v$ (line 14), where R_1 contains all the non-dominated solutions in $P \cup P'$, and R_i $(i \geq 2)$ contains all the non-dominated solutions in $(P \cup P') \setminus \cup_{j=1}^{i-1} R_j$. Then, the solutions in $R_1, R_2, \ldots, R_v$ are added into the next population, until the population size exceeds μ (lines 15–18). For the critical set R_i whose inclusion makes the population size larger than μ, the crowding distance is computed for each of the contained solutions (line 19). Crowding distance reflects the level of crowdedness of solutions in the population. For each objective f_j, $1 \leq j \leq m$, the solutions in R_i are sorted according to their objective values in ascending order, and assume the sorted list is $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k$. Then, the crowding distance of the solution $\boldsymbol{x}^l$ with respect to f_j is set to ∞ if $l \in \{1, k\}$ and $(f_j(\boldsymbol{x}^{l+1}) - f_j(\boldsymbol{x}^{l-1}))/(f_j(\boldsymbol{x}^k) - f_j(\boldsymbol{x}^1))$ otherwise. The final crowding distance of a solution is the sum of the crowding distance with respect to each objective. Finally, the solutions in R_i are selected to fill the remaining population slots where the solutions with larger crowding distance are preferred (line 20).

SMS-EMOA shares a similar framework with $(\mu + 1)$-GA, and the main difference is that SMS-EMOA uses non-dominated sorting and hypervolume indicator to evaluate the quality of a solution and update the population. Specifically, after the offspring solution $\boldsymbol{x}''$ is generated, the solutions in $P \cup \{\boldsymbol{x}''\}$ are first partitioned into non-dominated sets $R_1, \ldots, R_v$, and then one solution $\boldsymbol{z} \in R_v$ that minimizes $\Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$ is removed, where $\Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v) = HV_{\boldsymbol{r}}(R_v) - HV_{\boldsymbol{r}}(R_v \setminus \{\boldsymbol{x}\})$. Note that $HV_{\boldsymbol{r}}(X) = \Lambda\big(\cup_{\boldsymbol{x} \in X} \{\boldsymbol{f}' \in \mathbb{R}^m \mid \forall 1 \leq i \leq m : r_i \leq f'_i \leq f_i(\boldsymbol{x})\}\big)$ denotes the hypervolume of a solution set X with respect to a reference point $\boldsymbol{r} \in \mathbb{R}^m$ (satisfying $\forall 1 \leq i \leq m, r_i \leq \min_{\boldsymbol{x} \in \mathcal{X}} f_i(\boldsymbol{x})$), i.e., the volume of the objective space between the reference point and the objective vectors of the solution set, where Λ denotes the Lebesgue measure. A larger hypervolume implies a better approximation in terms of convergence and diversity.

Algorithm 1 NSGA-II

Input: objective functions $f_1, f_2 \ldots, f_m$, population size μ, probability p_c of using crossover
Output: μ solutions from $\{0,1\}^n$

1: $P \leftarrow \mu$ solutions uniformly and randomly selected from $\{0,1\}^n$ with replacement;
2: **while** criterion is not met **do**
3: let $P' = \emptyset$;
4: generate a parent population Q of size μ;
5: **for** each pair of the parent solutions $\boldsymbol{x}$ and $\boldsymbol{y}$ in Q **do**
6: sample u from uniform distribution over $[0, 1]$;
7: **if** $u < p_c$ **then**
8: apply uniform crossover on $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate two solutions $\boldsymbol{x}'$ and $\boldsymbol{y}'$
9: **else**
10: set $\boldsymbol{x}'$ and $\boldsymbol{y}'$ as copies of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively
11: **end if**
12: apply bit-wise mutation on $\boldsymbol{x}'$ and $\boldsymbol{y}'$ to generate $\boldsymbol{x}''$ and $\boldsymbol{y}''$, respectively, and add $\boldsymbol{x}''$ and $\boldsymbol{y}''$ into P'
13: **end for**
14: partition $P \cup P'$ into non-dominated sets $R_1, \ldots, R_v$;
15: let $P = \emptyset$, $i = 1$;
16: **while** $|P \cup R_i| < \mu$ **do**
17: $P = P \cup R_i$, $i = i + 1$
18: **end while**
19: assign each solution in R_i with a crowding distance;
20: sort the solutions in R_i in ascending order by crowding distance, and add the last $\mu - |P|$ solutions into P
21: **end while**
22: **return** P

Note that in this paper, we consider different methods to select the parent solutions in line 4 of Algorithm 1, i.e., fair selection which selects each solution in P once (the order of solutions is random), uniform selection which selects parent solutions from P independently and uniformly at random for μ times, and binary tournament selection which first picks two solutions randomly from the population P with replacement and then selects a better one for μ times. Furthermore, for all the three algorithms studied in this paper, the probability p_c of using crossover belongs to the interval $[\Omega(1), 1 - \Omega(1)]$.

3 Proposed Diversity Maintenance Method

In this section, we introduce the proposed method that is used to maintain the diversity of solutions in the population update procedure of $(\mu + 1)$-GA, NSGA-II and SMS-EMOA. The general idea of our method is very simple – we take into account the (Hamming) distance of the solutions into the population update procedure of the algorithm when the solutions have the same objective value, such that the distant solutions in the solution space can be preserved.

First, let us consider $(\mu+1)$-GA. In the original population update procedure of $(\mu+1)$-GA, the solution with the minimal objective value is removed, with the ties broken uniformly. That is, the solutions with the minimal objective value are treated equally, regardless of their structure in the solution space. In order to take advantage of the genotype information of the solutions, our method will further consider the diversity of the solutions with the minimal objective value, and preserve the solutions with larger diversity. Specifically, the procedure of selecting solution $\boldsymbol{z}$ to remove is changed as follows (note that $H(\cdot,\cdot)$ denotes the Hamming distance of two solutions):

let $S = \{\boldsymbol{z} \mid f(\boldsymbol{z}) = \min_{\boldsymbol{x}\in P\cup\{\boldsymbol{x}''\}} f(\boldsymbol{x})\}$;
if $|S| \leq 2$ **then**
  let $\boldsymbol{z}$ be a solution randomly selected from S
else
  let $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) = \arg\max_{\boldsymbol{x},\boldsymbol{y}\in S,\boldsymbol{x}\neq\boldsymbol{y}} H(\boldsymbol{x},\boldsymbol{y})$;
  let $\boldsymbol{z}$ be a solution randomly selected from $S\backslash\{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}\}$
end if

Now we consider NSGA-II. When computing the crowding distance for the solutions in the critical non-dominated set, i.e., R_i in line 19 of Algorithm 1, the solutions with the same objective vector must be sorted together. Then, the solutions placed in the first or the last position among these solutions will be assigned a crowding distance larger than 0, while the other solutions will only be with a crowding distance of 0. In this case, the solutions (corresponding to the same objective vector) placed in the boundary positions will be preferred by the measure of crowding distance. In the original NSGA-II, the solutions (corresponding to the same objective vector) are deemed as identical and their relative positions in the calculation of crowding distance are actually assigned randomly, which may harm the efficiency of the algorithm, as we will show later.

As can be seen, here the main idea of our method is to modify the solutions' position in the crowding distance calculation, so that the distant solutions in the solution space are preferred. Specifically, after the solutions in the critical non-dominated set R_i are sorted according to some objective f_j in ascending order (here assuming that the sorted list is $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k$), we reorder the list as follows:

let $G = \{f_j(\boldsymbol{x}^l) \mid l \in [1..k]\}$;
for $g \in G$ **do**
  let $S_g = \{l \in [1..k] \mid f_j(\boldsymbol{x}^l) = g\}$;
  let $(\tilde{a}, \tilde{b}) = \arg\max_{a,b\in S_g, a\neq b} H(\boldsymbol{x}^a, \boldsymbol{x}^b)$;
  let $\hat{l} = \min\{l \in S_g\}$, $\hat{r} = \max\{r \in S_g\}$;
  swap the positions of $\boldsymbol{x}^{\tilde{a}}$ and $\boldsymbol{x}^{\hat{l}}$;
  swap the positions of $\boldsymbol{x}^{\tilde{b}}$ and $\boldsymbol{x}^{\hat{r}}$
end for

That is, for the solutions with the same objective value, the pair of solutions with the largest Hamming distance are put in the first and last positions among these solutions. The crowding distance is then calculated based on the reordered list.

Finally, let us consider SMS-EMOA. In the original population update procedure of SMS-EMOA, the solution in the last non-dominated set R_v with the least Δ value is removed. However, when several solutions in R_v have the same objective vector, they will all have a zero Δ value, implying that one of them will be removed randomly. To make use of their crowdedness in the solution space, we first select the most crowded objective vector (i.e., the objective vector corresponding to the most solutions), and then randomly remove one of those corresponding solutions excluding the two solutions having the largest Hamming distance. Specifically, the procedure of selecting solution $\boldsymbol{z}$ that minimizes $\Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$ will be changed to:

let $G = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in R_v\}$;
let $\boldsymbol{g}^* = \arg\max_{\boldsymbol{g}\in G} |\{\boldsymbol{x} \in R_v \mid \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{g}\}|$;
let $S = \{\boldsymbol{x} \in R_v \mid \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{g}^*\}$;
if $|S| > 2$ **then**
  let $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}) = \arg\max_{\boldsymbol{x},\boldsymbol{y}\in S,\boldsymbol{x}\neq\boldsymbol{y}} H(\boldsymbol{x},\boldsymbol{y})$;
  let $\boldsymbol{z}$ be a solution randomly selected from $S\backslash\{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}\}$;
else
  let $\boldsymbol{z} = \arg\min_{\boldsymbol{x}\in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$
end if

4 $(\mu+1)$-GA on Jump

In this section, we show the expected running time of $(\mu+1)$-GA for solving the Jump problem. Note that the running time of EAs is often measured by the number of fitness evaluations, the most time-consuming step in the evolutionary process. We prove in Theorem 1 that the expected number of fitness evaluations of $(\mu+1)$-GA using the proposed diversity maintenance method for solving Jump is $O(\mu^2 4^k + \mu n \log n + n\sqrt{k}(\mu \log \mu + \log n))$. The proof idea is to divide the optimization procedure into three phases, where the first phase aims at driving the population towards the local optimum, i.e., all the solutions in the population have $(n-k)$ 1-bits, the second phase aims at finding two local optimal solutions with the largest Hamming distance $2k$, and the third phase applies uniform crossover to these two distant local optimal solutions to find the global optimum.

Theorem 1. *For $(\mu+1)$-GA solving Jump with $k \leq n/4$, if using the diversity maintenance method, and a population size μ such that $\mu \geq 2$, then the expected number of fitness evaluations for finding the global optimal solution 1^n is $O(\mu^2 4^k + \mu n \log n + n\sqrt{k}(\mu \log \mu + \log n))$.*

Proof. We divide the optimization procedure into three phases. The first phase starts after initialization and finishes until all the solutions in the population have $(n-k)$ 1-bits; the second phase starts after the first phase, and finishes until the largest Hamming distance between solutions in the population reaches $2k$; the third phase starts after the second phase, and finishes when the global optimal solution 1^n is found. By Lemma 1 in [Dang *et al.*, 2018], we can directly derive that the expected number of fitness evaluations of the first phase is $O(n\sqrt{k}(\mu \log \mu + \log n))$. Note

that after the first phase finishes, all the solutions in the population will always have $(n-k)$ 1-bits (except that 1^n is found), because once a solution with the number of 1-bits in $[0..n-k-1] \cup [n-k+1..n-1]$ is generated, it will be removed in the population update procedure.

Next, we consider the second phase. Let $J_{\max} = \max_{\boldsymbol{x},\boldsymbol{y}\in P} H(\boldsymbol{x},\boldsymbol{y})$ denote the maximal Hamming distance between two solutions in the population P, where $H(\cdot,\cdot)$ denotes the Hamming distance of two solutions. We show that $J_{\max}$ will not decrease, unless 1^n is found. Assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are a pair of solutions which have the maximal Hamming distance. In each iteration, $\boldsymbol{x}$ or $\boldsymbol{y}$ can be removed only if the newly generated solution $\boldsymbol{z}$ has exactly $(n-k)$ 1-bits and survives in the population update procedure (note that we can pessimistically assume that 1^n has not been found), implying that all the solutions in $P \cup \{\boldsymbol{z}\}$ have $(n-k)$ 1-bits. According to the diversity maintenance method introduced in Section 3, there must exist a pair of solutions in $P \cup \{\boldsymbol{z}\}$ with Hamming distance at least $J_{\max}$, which will not be removed, implying that $J_{\max}$ cannot decrease.

We then show that $J_{\max}$ can increase to $2k$ in $O(\mu n \log n)$ expected number of fitness evaluations. Assume that currently $J_{\max} = 2j < 2k$, and $(\boldsymbol{x},\boldsymbol{y})$ is a pair of corresponding solutions, i.e., $H(\boldsymbol{x},\boldsymbol{y}) = 2j$. Let $D(\boldsymbol{x},\boldsymbol{y}) = \{i \in [1..n] \mid x_i = y_i\}$ denote the set of positions which have the identical bit for $\boldsymbol{x}$ and $\boldsymbol{y}$. Suppose that the number of positions in $D(\boldsymbol{x},\boldsymbol{y})$ that have 1-bits is l, then we have $|\boldsymbol{x}| + |\boldsymbol{y}| = 2l + 2j = 2n - 2k$, implying that $l = n-k-j$. Then, we can derive that the number of positions in $D(\boldsymbol{x},\boldsymbol{y})$ that have 0-bits is $n - 2j - l = k - j$. In each iteration, the probability of selecting $\boldsymbol{x}$ as a parent solution is $1/\mu$. In the reproduction procedure, a solution $\boldsymbol{z}$ with $(n-k)$ 1-bits and $H(\boldsymbol{y},\boldsymbol{z}) = 2j+2$ can be generated from $\boldsymbol{x}$ if crossover is not performed (whose probability is $1-p_c$), and a 1-bit together with a 0-bit from the positions in $D(\boldsymbol{x},\boldsymbol{y})$ are flipped by bit-wise mutation (whose probability is $((n-k-j)(k-j)/n^2) \cdot (1-1/n)^{n-2}$). Thus, the probability of generating $\boldsymbol{z}$ is at least

$$\begin{aligned} &\frac{1}{\mu} \cdot (1-p_c) \cdot \frac{(n-k-j)(k-j)}{n^2} \cdot \Big(1-\frac{1}{n}\Big)^{n-2} \\ &\geq (1-p_c)(k-j)(n-k-j)/(e\mu n^2). \end{aligned} \tag{1}$$

This implies that the expected number of fitness evaluations for increasing $J_{\max}$ to $2k$ is at most

$$\frac{e\mu n^2}{1-p_c} \cdot \sum_{j=0}^{k-1} \frac{1}{(k-j)(n-k-j)} \leq \frac{2e\mu n H_k}{1-p_c} = O(\mu n \log n), \tag{2}$$

where the inequality holds because $n-k-j \geq n-2k+1 \geq n/2$ for $j \leq k-1$ and $k \leq n/4$, and the equality holds because $p_c \in [\Omega(1), 1-\Omega(1)]$, and the k-th Harmonic number $H_k = \sum_{j=1}^{k} 1/j = O(\log n)$.

Finally, we consider the third phase. When $J_{\max}$ increases to $2k$, there must exist two solutions $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ such that $|\boldsymbol{x}^*|_1 = |\boldsymbol{y}^*|_1 = n-k$ and $H(\boldsymbol{x}^*,\boldsymbol{y}^*) = 2k$. By selecting $\boldsymbol{x}^*$ and $\boldsymbol{y}^*$ as a pair of parent solutions, exchanging all the $2k$ different bits by uniform crossover, and flipping none of bits in bit-wise mutation, the solution 1^n can be generated, whose probability is at least

$$\frac{1}{\mu^2} \cdot p_c \cdot \frac{1}{2^{2k}} \cdot \Big(1-\frac{1}{n}\Big)^n = \Omega\Big(\frac{1}{\mu^2 2^{2k}}\Big). \tag{3}$$

Thus, the expected number of fitness evaluations of the third phase, i.e., finding 1^n, is $O(\mu^2 2^{2k})$.

Combining the three phases, the total expected number of fitness evaluations is $O(n\sqrt{k}(\mu \log \mu + \log n)) + O(\mu n \log n) + O(\mu^2 2^{2k})$, which leads to the theorem. □

The expected number of fitness evaluations of the original $(\mu+1)$-GA for solving Jump has been shown to be $O(\mu\sqrt{k}(40e^2\mu n/k)^k)$ when $k = o(\sqrt{n})$ [Doerr and Qu, 2023c], and $O(\mu n \log \mu + n^k/\mu + n^{k-1}\log \mu)$ when k is a constant [Dang *et al.*, 2018]. Thus, our result in Theorem 1 shows that when k is large, e.g., $k = n^{1/4}$, the expected running time can be reduced by a factor of $\Omega((10e^2\mu n/k)^k/(\mu n \log n))$, which is exponential; when k is a constant, the expected running time can be reduced by a factor of $\Omega(\max(1, n^{k-1}/(\mu^2 \log n)))$, which is polynomial. The main reason for the acceleration is that the proposed method prefers solutions with large Hamming distance, making the population quickly find two solutions with sufficiently large Hamming distance and then allowing crossover to generate the optimal solution.

Note that Dang *et al.* [2016] also proposed several types of diversity maintenance mechanisms for $(\mu+1)$-GA solving the Jump problem, including mechanisms similar to ours, e.g., total Hamming distance mechanism which maximizes the sum of the Hamming distances between all solutions. Based on the running time bounds, their mechanisms and our proposed mechanism have own advantages. For example, when k and μ are fairly large (e.g., $k = \sqrt{n}$ and $\mu = \sqrt{n}/3$), the expected number of fitness evaluations of $(\mu+1)$-GA using the total Hamming distance mechanism and our proposed mechanism is $O(4^{\sqrt{n}})$ and $O(n4^{\sqrt{n}})$, respectively, implying that the total Hamming distance mechanism is faster by a factor of $\Theta(n)$. For small k and large μ (e.g., $k = \Theta(1)$ and $\mu = \Theta(n)$), however, our proposed mechanism achieves a better running time $O(n^2 \log n)$, compared to $O(n^3 \log n)$ for the total Hamming distance mechanism. Similar observation also holds for the other mechanisms.

5 NSGA-II on OneJumpZeroJump

In the previous section, we have shown that the proposed method can make $(\mu+1)$-GA faster on Jump, while in this section, we show that similar acceleration can be achieved in multi-objective scenario. In particular, we prove in Theorem 2 that the expected number of fitness evaluations of NSGA-II using the proposed diversity maintenance method for solving OneJumpZeroJump is $O(\mu^2 4^k + \mu n \log n)$. The proof idea is to divide the optimization procedure into three phases: 1) to find the inner part $F_I^* = \{(a, n+2k-a) \mid a \in [2k..n]\}$ of the Pareto front; 2) to find two Pareto optimal solutions with $(n-k)$ (or k) 1-bits that have the same objective vector $(n, 2k)$ (or $(2k, n)$) but have Hamming distance $2k$; 3) to find the remaining two extreme vectors in the

Pareto front, i.e., $\{(k, n+k), (n+k, k)\}$, corresponding to the two Pareto optimal solutions 0^n and 1^n, which can be accomplished by applying uniform crossover to those two distant solutions found in the last phase.

Theorem 2. *For NSGA-II solving OneJumpZeroJump with $k \leq n/4$, if using the diversity maintenance method, and a population size μ such that $\mu \geq 4(n-2k+3)$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu^2 4^k + \mu n \log n)$.*

Proof. We divide the optimization procedure into three phases: the first phase starts after initialization and finishes until the inner part F_I^* of the Pareto front is entirely covered by the population P; the second phase starts after the first phase and finishes until the maximal Hamming distance of the solutions which have the objective vector $(n, 2k)$ (i.e., have $(n-k)$ 1-bits) reaches $2k$; the third phase starts after the second phase and finishes when the extreme Pareto optimal solution 1^n is found. Note that the analysis of the last two phases for finding the other extreme Pareto optimal solution 0^n is similar. By Lemma 4 in [Doerr and Qu, 2023a], we can directly derive that the expected number of fitness evaluations of the first phase is $O(\mu n \log n)$.

Then, we consider the second phase. Let $J_{\max} = \max_{\boldsymbol{x},\boldsymbol{y}\in P:\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{y})=(n,2k)} H(\boldsymbol{x},\boldsymbol{y})$ denote the maximal Hamming distance of the solutions which have the objective vector (n, k). We will show that $J_{\max}$ will not decrease after using the proposed diversity maintenance method. Because (n, k) is a point in the Pareto front, any corresponding solution (which has $(n-k)$ 1-bits) must have rank 1, i.e., belong to R_1 in the non-dominated sorting procedure. If $|R_1| \leq \mu$, all the solutions in R_1 will be maintained in the next population, implying the claim holds. If $|R_1| > \mu$, the crowding distance of the solutions in R_1 needs to be computed. When the solutions in R_1 are sorted according to some objective, the solutions corresponding to the objective vector (n, k) will be resorted using our proposed method, and a pair of solutions (denoted as $\boldsymbol{x}$ and $\boldsymbol{y}$) with the largest Hamming distance will be put in the first or the last position among these solutions, thus having a crowding distance larger than 0. Since (n, k) has been obtained, the solutions in R_1 must be Pareto optimal. Note that the size of the Pareto front is $n-2k+3$, thus the number of different objective vectors of the solutions in R_1 is at most $n-2k+3$. For any objective vector, the corresponding solutions will have crowding distance larger than 0 only if they are put in the first or the last position among these solutions, when they are sorted according to some objective. As OneJumpZeroJump has two objectives, at most four solutions corresponding to one objective vector can have a crowding distance larger than 0, implying that at most $4(n-2k+3)$ solutions in R_1 can have a positive crowding distance. Thus, $\boldsymbol{x}$ and $\boldsymbol{y}$ are among the best $4(n-2k+3)$ solutions in R_1. As the population size $\mu \geq 4(n-2k+3)$, they must be included in the next population, implying that $J_{\max}$ will not decrease.

Next, we show that $J_{\max}$ can increase to $2k$ in $O(\mu n \log n)$ expected number of fitness evaluations. The proof is similar to the argument in the proof of Theorem 1, and the main difference is that NSGA-II uses three different parent selection strategies, and produces μ solutions instead of one in each generation. For NSGA-II using uniform selection, Eq. (1) also holds here. But since in each generation, $\mu/2$ pairs of parent solutions will be selected for reproduction, the probability of increasing $J_{\max}$ is at least

$$1-\left(1-\frac{(1-p_c)(k-j)(n-k-j)}{e\mu n^2}\right)^{\mu/2} \geq 1-e^{-\frac{(1-p_c)(k-j)(n-k-j)}{2en^2}} = \Omega\left(\frac{(k-j)(n-k-j)}{n^2}\right),$$

which hold by $1+a \leq e^a$ for any $a \in \mathbb{R}$, and $p_c \in [\Omega(1), 1-\Omega(1)]$. For fair selection, each solution in the current population will be selected once; thus by Eq. (1), the probability of increasing $J_{\max}$ in each generation is at least $(1-p_c)(k-j)(n-k-j)/(en^2) = \Omega((k-j)(n-k-j)/n^2)$. For binary tournament selection, the above equation also holds, but the detailed analysis is provided in the supplementary material due to space limitation. Then, similar to Eq. (2), we can derive that the total expected number of generations for increasing $J_{\max}$ to $2k$ is at most $O(n \log n)$.

Finally, we consider the third phase. For uniform selection, Eq. (3) directly applies here. Thus, the probability of finding 1^n in each generation is at least

$$1-\Big(1-\Omega\Big(\frac{1}{\mu^2 2^{2k}}\Big)\Big)^{\mu/2} = \Omega\Big(\frac{1}{\mu 4^k}\Big).$$

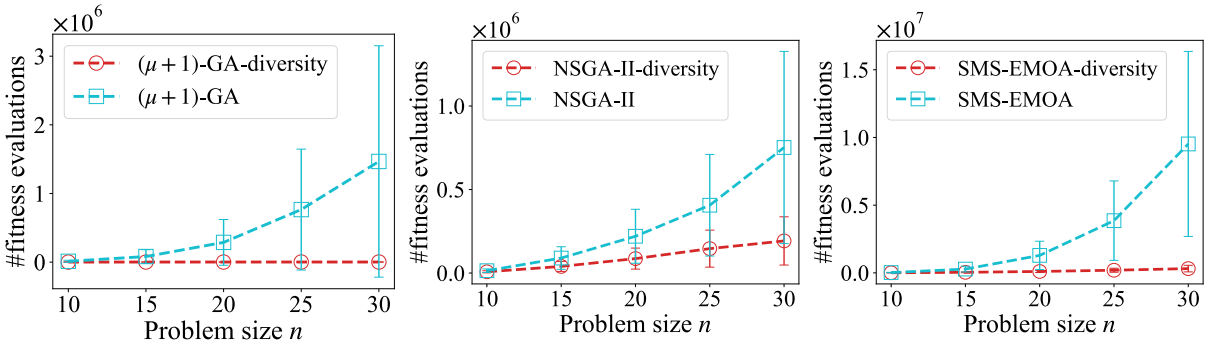
For fair selection, the probability that two solutions are paired together in each generation is at least $1/\mu$, because we can assume that the position of one solution is fixed and it is sufficient to put the other solution beside it. Thus by Eq. (3), the probability of finding 1^n in each generation is at least $\Omega(1/(\mu 4^k))$. For binary tournament selection, the above equation also holds, and the detailed analysis is provided in the supplementary material. Thus, the expected number of generations for finding 1^n is $O(\mu 4^k)$.

Combining the three phases, the total expected number of generation is $O(\mu 4^k + n \log n)$, implying that the expected number of fitness evaluations is $O(\mu^2 4^k + \mu n \log n)$, because each generation of NSGA-II requires to evaluate μ offspring solutions. Thus, the theorem holds. □

The expected number of fitness evaluations of the original NSGA-II for solving OneJumpZeroJump has been shown to be $O(\mu^2\sqrt{k}(Cn/k)^k)$ if $k = o(\sqrt{n})$ [Doerr and Qu, 2023c], where C is a constant. Thus, our result in Theorem 2 shows that the expected running time can be reduced by a factor of $\Omega(\sqrt{k}(Cn/(4k))^k/\log n)$, which is polynomial for a constant k and exponential for large k, e.g., $k = n^{1/4}$. The main reason for the acceleration is similar to that of $(\mu+1)$-GA, i.e., the proposed diversity maintenance method prefers distant solutions in the solution space, and thus enhances the exploration ability of the algorithm by working with the crossover operator.

6 SMS-EMOA on OneJumpZeroJump

In this section, we consider SMS-EMOA on the OneJumpZeroJump problem. We prove in Theorem 3 that the expected number of fitness evaluations of the original SMS-EMOA is $O(\mu^2\sqrt{k}(Cn/k)^k)$, where C is a constant, and



(a) $(\mu+1)$-GA (b) NSGA-II (c) SMS-EMOA

Figure 1: Average number of fitness evaluations of $(\mu+1)$-GA for solving Jump, and NSGA-II/SMS-EMOA for solving OneJumpZeroJump, when the diversity maintenance method is used or not.

then prove in Theorem 4 that the running time reduces to $O(\mu^2 4^k + \mu n \log n)$ if using the proposed diversity maintenance method. Their proofs are similar to that of Theorem 2. That is, we divide the optimization procedure into three phases, where the first phase aims at finding the inner part F_I^* of the Pareto front, the second phase aims at finding two Pareto optimal solutions with $(n-k)$ (or k) 1-bits that have Hamming distance $2k$, and the last phase aims at finding the remaining two extreme Pareto optimal solutions 1^n and 0^n by applying uniform crossover to those two distant solutions found in the last phase. The detailed proofs are provided in the supplementary material due to space limitation.

Theorem 3. *For SMS-EMOA solving OneJumpZeroJump with $k = o(\sqrt{n})$, if using a population size μ such that $\mu \geq 2(n-2k+3)$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu^2\sqrt{k}(Cn/k)^k)$, where C is a constant.*

Theorem 4. *For SMS-EMOA solving OneJumpZeroJump with $k \leq n/4$, if using the diversity maintenance method, and a population size μ such that $\mu \geq 2(n-2k+3)$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu^2 4^k + \mu n \log n)$.*

By comparing Theorems 3 and 4, we can find that by using the diversity maintenance method, the expected number of fitness evaluations of SMS-EMOA for solving OneJumpZeroJump can be reduced by a factor of $\Omega(\sqrt{k}(Cn/(4k))^k/\log n)$, which is polynomial for a constant k and exponential for large k, e.g., $k = n^{1/4}$. The main reason for the acceleration is just similar to that of NSGA-II. That is, for Pareto optimal solutions with $(n-k)$ (or k) 1-bits, the proposed method always maintains the two ones with the largest Hamming distance in the population update procedure, which enables the algorithm to quickly increase the largest Hamming distance until reaching the maximum $2k$, i.e., finishing the second phase.

7 Experiments

In the previous sections, we have proved that using the proposed diversity maintenance method in $(\mu+1)$-GA, NSGA-II and SMS-EMOA can bring a significant acceleration for solving Jump and OneJumpZeroJump. However, as only upper bounds on the running time of the original algorithms have been derived, we conduct experiments to examine their actual performance to complement the theoretical results.

Specifically, we set the problem size n of Jump and OneJumpZeroJump from 10 to 30, with a step of 5, and set the parameter k of the two problems to 4. For $(\mu+1)$-GA solving Jump, the population size μ is set to 2, as suggested in Theorem 1; while for NSGA-II and SMS-EMOA solving OneJumpZeroJump, the population size μ is set to $4(n-2k+3)$ and $2(n-2k+3)$, respectively, as suggested in Theorems 2 and 4. For each n, we run an algorithm 1000 times independently, and record the average number of fitness evaluations until the optimal solution (for Jump) or the Pareto front (for OneJumpZeroJump) is found. We can observe from Figure 1 that using the proposed method can bring a clear acceleration.

8 Conclusion

This paper gives a theoretically study for EAs solving MMOPs, a class of optimization problems with wide applications. Considering the characteristic of MMOPs that multiple (local) optimal solutions in the solution space correspond to a single point in the objective space, we propose to prefer those solutions diverse in the solution space when they are equally good in the objective space. We show that such a method, working with crossover, can benefit the evolutionary search via rigorous running time analysis. Specifically, we prove that for $(\mu+1)$-GA solving the widely studied single-objective problem, Jump, as well as NSGA-II and SMS-EMOA solving the widely studied bi-objective problem, OneJumpZeroJump, the proposed method can lead to polynomial or even exponential acceleration on the expected running time, which is also verified by the experiments. Note that diversity has been theoretically shown to be helpful for evolutionary optimization in various scenarios [Friedrich *et al.*, 2009; Qian *et al.*, 2013; Doerr *et al.*, 2016; Osuna *et al.*, 2020; Sudholt, 2020; Doerr *et al.*, 2023; Qian *et al.*, 2024]. This work theoretically shows the benefit of maintaining the diversity in the solution space for EAs solving MMOPs, thus contributing to this line of research. We hope our results can be beneficial for the design of better EAs for solving MMOPs, and may also encourage the diversity exploration in the solution space for multi-objective EAs, which is often ignored.

Acknowledgments

This work was supported by the National Science and Technology Major Project (2022ZD0116600) and National Science Foundation of China (62276124). Chao Qian is the corresponding author. The supplementary is available at arXiv.

References

[Auger and Doerr, 2011] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics - Foundations and Recent Developments*. World Scientific, Singapore, 2011.

[Basseur *et al.*, 2002] M. Basseur, F. Seynhaeve, and E. Talbi. Design of multi-objective evolutionary algorithms: application to the flow-shop scheduling problem. In *Proceedings of the 2002 CEC*, pages 1151–1156, Honolulu, HI, 2002.

[Beume *et al.*, 2007] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181:1653–1669, 2007.

[Bian *et al.*, 2018] C. Bian, C. Qian, and K. Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In *Proceedings of the 27th IJCAI*, pages 1405–1411, Stockholm, Sweden, 2018.

[Bian *et al.*, 2023] C. Bian, Y. Zhou, M. Li, and C. Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *Proceedings of the 32nd IJCAI*, pages 5513–5521, Macao, SAR, China, 2023.

[Cheng *et al.*, 2018] R. Cheng, M. Li, K. Li, and X. Yao. Evolutionary multiobjective optimization-based multimodal optimization: Fitness landscape approximation and peak detection. *IEEE Transactions on Evolutionary Computation*, 22(5):692–706, 2018.

[Dang and Lehre, 2015] D.-C. Dang and P. K. Lehre. Runtime analysis of non-elitist populations: From classical optimisation to partial information. *Algorithmica*, 75(3):428–461, 2015.

[Dang *et al.*, 2016] D.-C. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, and A. M. Sutton. Escaping local optima with diversity mechanisms and crossover. In *Proceedings of the 18th GECCO*, pages 645–652, Denver, CO, 2016.

[Dang *et al.*, 2018] D.-C. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, and A. M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22(3):484–497, 2018.

[Das *et al.*, 2011] S. Das, S. Maity, B. Qu, and P. N. Suganthan. Real-parameter evolutionary multimodal optimization — a survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88, 2011.

[Deb and Saha, 2010] K. Deb and A. Saha. Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach. In *Proceedings of the 12th GECCO*, page 447–454, Portland, OR, 2010.

[Deb *et al.*, 2002] 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.

[Doerr and Neumann, 2020] B. Doerr and F. Neumann. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, Cham, Switzerland, 2020.

[Doerr and Qu, 2023a] B. Doerr and Z. Qu. A first runtime analysis of the NSGA-II on a multimodal problem. *IEEE Transactions on Evolutionary Computation*, 27(5):1288–1297, 2023.

[Doerr and Qu, 2023b] B. Doerr and Z. Qu. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Proceedings of the 37th AAAI*, pages 12408–12416, Washington, DC, 2023.

[Doerr and Qu, 2023c] B. Doerr and Z. Qu. Runtime analysis for the NSGA-II: Provable speed-ups from crossover. In *Proceedings of the 37th AAAI*, pages 12399–12407, Washington, DC, 2023.

[Doerr and Zheng, 2021] B. Doerr and W. Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Proceedings of the 35th AAAI*, pages 12293–12301, Virtual, 2021.

[Doerr *et al.*, 2012] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[Doerr *et al.*, 2016] B. Doerr, W. Gao, and F. Neumann. Runtime analysis of evolutionary diversity maximization for oneminmax. In *Proceedings of the 18th GECCO*, pages 557–564, Denver, CO, 2016.

[Doerr *et al.*, 2023] B. Doerr, A. Echarghaoui, M. Jamal, and M. S. Krejca. Lasting diversity and superior runtime guarantees for the $(\mu + 1)$ genetic algorithm. *CORR abs/2302.12570*, 2023.

[Droste *et al.*, 2002] 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.

[Friedrich *et al.*, 2009] T. Friedrich, N. Hebbinghaus, and F. Neumann. Comparison of simple diversity mechanisms on plateau functions. *Theoretical Computer Science*, 410(26):2455–2462, 2009.

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

[Kennedy, 2010] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*. Springer US, Boston, MA, 2010.

[Kudo *et al.*, 2011] F. Kudo, T. Yoshikawa, and T. Furuhashi. A study on analysis of design variables in Pareto solutions for conceptual design optimization problem of hybrid rocket engine. In *Proceedings of the 2011 CEC*, pages 2558–2562, New Orleans, LA, 2011.

[Liang *et al.*, 2024] J. Liang, Y. Zhang, K. Chen, B. Qu, K. Yu, C. Yue, and P. N. Suganthan. An evolutionary multiobjective method based on dominance and decomposition for feature selection in classification. *Science China Information Sciences*, 67(2):120101, 2024.

[Liu *et al.*, 2022] Y.-R. Liu, Y.-Q. Hu, H. Qian, C. Qian, and Y. Yu. Zoopt: a toolbox for derivative-free optimization. *Science China Information Sciences*, 65(10):207101, 2022.

[Lu *et al.*, 2024] T. Lu, C. Bian, and C. Qian. Towards running time analysis of interactive multi-objective evolutionary algorithms. In *Proceedings of the 38th AAAI*, pages 20777–20785, Vancouver, Canada, 2024.

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

[Oliveto and Witt, 2011] P. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386, 2011.

[Osuna *et al.*, 2020] E. C. Osuna, W. Gao, F. Neumann, and D. Sudholt. Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multiobjective optimisation. *Theoretical Computer Science*, 832:123–142, 2020.

[Pan *et al.*, 2023] S. Pan, Y. Ma, Y. Wang, Z. Zhou, J. Ji, M. Yin, and S. Hu. An improved master-apprentice evolutionary algorithm for minimum independent dominating set problem. *Frontiers of Computer Science*, 17(4):174326, 2023.

[Pétrowski, 1996] A. Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of the 1996 CEC*, pages 798–803, Nagoya, Japan, 1996.

[Preuss *et al.*, 2021] M. Preuss, M. Epitropakis, X. Li, and J. E. Fieldsend. Multimodal optimization: Formulation, heuristics, and a decade of advances. *Metaheuristics for Finding Multiple Solutions*, pages 1–26, 2021.

[Preuss, 2015] M. Preuss. *Niching Methods and Multimodal Optimization Performance*, pages 115–137. Springer, Cham, Switzerland, 2015.

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

[Qian *et al.*, 2016] C. Qian, Y. Yu, and Z.-H. Zhou. A lower bound analysis of population-based evolutionary algorithms for pseudo-Boolean functions. In *Procedings of the 17th IDEAL*, pages 457–467, Yangzhou, China, 2016.

[Qian *et al.*, 2024] C. Qian, K. Xue, and R.-J. Wang. Quality-diversity algorithms can provably be helpful for optimization. In *Proceedings of the 33rd IJCAI*, page to appear, Jeju Island, South Korea, 2024.

[Reintjes, 2022] C. Reintjes. Optimization of truss structures. In *Algorithm-Driven Truss Topology Optimization for Additive Manufacturing*, pages 45–70. Springer Fachmedien Wiesbaden, Wiesbaden, Germany, 2022.

[Schutze *et al.*, 2011] O. Schutze, M. Vasile, and C. A. C. Coello. Computing the set of epsilon-efficient solutions in multiobjective space mission design. *Journal of Aerospace Computing, Information, and Communication*, 8(3):53–70, 2011.

[Sebag *et al.*, 2005] M. Sebag, N. Tarrisson, O. Teytaud, J. Lefevre, and S. Baillet. A multi-objective multi-modal optimization approach for mining stable spatio-temporal patterns. In *Proceedings of the 19th IJCAI*, pages 859–864, Edinburgh, UK, 2005.

[Shir, 2012] O. M. Shir. *Niching in Evolutionary Algorithms*, pages 1035–1069. Springer Berlin Heidelberg, Heidelberg, Germany, 2012.

[Storn and Price, 1997] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[Sudholt, 2013] D. Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17(3):418–435, 2013.

[Sudholt, 2020] D. Sudholt. The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 359–404, 2020.

[Tian *et al.*, 2021] Y. Tian, R. Liu, X. Zhang, H. Ma, K.-C. Tan, and Y. Jin. A multipopulation evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 25(3):405–418, 2021.

[Wegener, 2002] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In *Evolutionary Optimization*, pages 349–369. Springer US, Boston, MA, 2002.

[Wessing *et al.*, 2013] S. Wessing, M. Preuss, and G. Rudolph. Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In *Proceedings of the 2013 CEC*, pages 103–110, Cancún, Mexico, 2013.

[Yu and Qian, 2015] Y. Yu and C. Qian. Running time analysis: Convergence-based analysis reduces to switch analysis. In *Proceedings of the 2015 CEC*, pages 2603–2610, Sendai, Japan, 2015.

[Yu *et al.*, 2015] Y. Yu, C. Qian, and Z.-H. Zhou. Switch analysis for running time analysis of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 19(6):777–792, 2015.

[Zhou *et al.*, 2019] Z.-H. Zhou, Y. Yu, and C. Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, Singapore, 2019.