Stochastic Population Update Can Provably Be Helpful in Multi-Objective Evolutionary Algorithms

Chao Bian[1] , **Yawen Zhou**[1] , **Miqing Li**[2] and **Chao Qian**[1*]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.
{bianc, zhouyw, qianc}@lamda.nju.edu.cn, m.li.8@bham.ac.uk

Abstract

Evolutionary algorithms (EAs) have been widely and successfully applied to solve multi-objective optimization problems, due to their nature of population-based search. Population update is a key component in multi-objective EAs (MOEAs), and it is performed in a greedy, deterministic manner. That is, the next-generation population is formed by selecting the first population-size ranked solutions (based on some selection criteria, e.g., non-dominated sorting, crowdedness and indicators) from the collections of the current population and newly-generated solutions. In this paper, we question this practice. We analytically present that introducing randomness into the population update procedure in MOEAs can be beneficial for the search. More specifically, we prove that the expected running time of a well-established MOEA (SMS-EMOA) for solving a commonly studied bi-objective problem, OneJumpZeroJump, can be exponentially decreased if replacing its deterministic population update mechanism by a stochastic one. Empirical studies also verify the effectiveness of the proposed stochastic population update method. This work is an attempt to challenge a common practice for the population update in MOEAs. Its positive results, which might hold more generally, should encourage the exploration of developing new MOEAs in the area.

1 Introduction

Multi-objective optimization refers to an optimization scenario that there are more than one objective to be considered at the same time, which often appears in real-world applications. Since the objectives of a multi-objective optimization problem (MOP) are usually conflicting, there does not exist a single optimal solution, but instead a set of solutions which represent different trade-offs between these objectives, called Pareto optimal solutions. The images of all Pareto optimal solutions of an MOP in the objective space are called the Pareto front. In multi-objective optimization, the goal of an optimizer is to find a good approximation of the Pareto front.

Evolutionary algorithms (EAs) [Bäck, 1996; Eiben and Smith, 2015] are a large class of randomized heuristic optimization algorithms, inspired by natural evolution. They maintain a set of solutions, i.e., a population, and iteratively improve it by generating new solutions and selecting better ones. Due to the population-based nature, EAs are well-suited to solving MOPs, and have been widely used in various real-world scenarios [Coello Coello and Lamont, 2004; Deb, 2011; Zhou *et al.*, 2019].

In multi-objective EAs (MOEAs), two key components are solution generation and population update. The former is concerned with parent selection and reproduction (e.g., crossover and mutation), while the latter (also called environmental selection or population maintenance) is concerned with maintaining a solution set which represents the best solutions found so far. In the area of evolutionary multi-objective optimization, the research focus is mainly on population update. That is, when designing an MOEA, attention is placed on how to update the population by newly-generated solutions so that a set of well distributed non-dominated solutions is preserved. With this aim, many selection criteria emerge, such as non-dominated sorting [Goldberg, 1989], crowding distance [Deb *et al.*, 2002], scalarizing functions [Zhang and Li, 2007] and quality indicators [Zitzler and Künzli, 2004].

A prominent feature in population update of MOEAs is its deterministic manner (apart from random tie breaking). That is, the next-generation population is formed by always selecting the first population-size ranked solutions out of the collections of the current population and newly-generated solutions. It is believed that a population formed by the best solutions has a higher chance to generate even better solutions.

In this paper, we challenge this belief. We analytically show that introducing randomness into the population update procedure may be beneficial for the search. Specifically, we consider a well-established MOEA, $\mathcal{S}$ metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA) [Beume *et al.*, 2007], and propose a simple stochastic population update method. Instead of removing the worst solution[1] from the collections of the current population and

*This work was supported by the National Science Foundation of China (62022039, 62276124). Chao Qian is the corresponding author.

[1]SMS-EMOA adopts a $(\mu + 1)$ steady state mode, where selecting the μ best solutions means removing the worst solution.

newly-generated solution, the proposed stochastic method first randomly selects a subset from the collections and then removes the worst solution from the subset. We theoretically show that this simple modification makes SMS-EMOA significantly better on OneJumpZeroJump, a bi-objective optimization problem commonly used in theoretical analyses of MOEAs [Doerr and Zheng, 2021; Doerr and Qu, 2022a; Doerr and Qu, 2022b; Doerr and Qu, 2022c]. More specifically, we first prove that the expected running time of the original SMS-EMOA is $O(\mu n^k)$ and $\Omega(n^k)$, where μ is the population size, n is the problem size and k $(2 \leq k < n/2)$ is the parameter of the problem. Then, we prove that using stochastic population update can decrease the expected running time to $O(\mu n^k \cdot \min\{1, \mu/2^{k/4}\})$. In other words, stochastic population update can bring an exponential acceleration when $\mu = poly(n)$ and $k = \Omega(n)$, where $poly(n)$ denotes any polynomial of n. The intuitive reason for this occurrence is that by introducing randomness into the population update procedure, the evolutionary search can go across inferior regions between different Pareto optimal solutions more easily, thus facilitating to find the whole Pareto front. Experiments are also conducted to verify our theoretical results.

There is an increasing interest to study MOEAs theoretically. Primitive theoretical work mainly focuses on analyzing the expected running time of GSEMO/SEMO, a simple MOEA which employs the bit-wise/one-bit mutation operator to generate an offspring solution in each iteration and keeps the non-dominated solutions generated-so-far in the population, in solving multi-objective synthetic and combinatorial optimization problems [Giel, 2003; Laumanns *et al.*, 2004a; Laumanns *et al.*, 2004b; Neumann, 2007; Horoba, 2009; Giel and Lehre, 2010; Neumann and Theile, 2010; Doerr *et al.*, 2013; Qian *et al.*, 2013; Bian *et al.*, 2018]. Based on (G)SEMO, the effectiveness of some parent selection and reproduction methods, e.g., greedy parent selection [Laumanns *et al.*, 2004b], diversity-based parent selection [Friedrich *et al.*, 2010; Covantes Osuna *et al.*, 2020], fairness-based parent selection [Laumanns *et al.*, 2004b; Friedrich *et al.*, 2011], fast mutation and stagnation detection [Doerr and Zheng, 2021], crossover [Qian *et al.*, 2013], and selection hyper-heuristics [Qian *et al.*, 2016], has also been studied.

Recently, researchers have started attempts to analyze practical MOEAs. The expected running time of $(\mu + 1)$ SIBEA, i.e., a simple MOEA using the hypervolume indicator to update the population, was analyzed on several synthetic problems [Brockhoff *et al.*, 2008; Nguyen *et al.*, 2015; Doerr *et al.*, 2016], which contributes to the theoretical understanding of indicator-based MOEAs. Later, people have started to consider well-established algorithms in the evolutionary multi-objective optimisation area. For example, Huang et al. [2021] considered MOEA/D, and examined the effectiveness of different decomposition methods by comparing the running time for solving many-objective synthetic problems. Very recently, Zheng *et al.* [2022] analyzed the expected running time of NSGA-II (which uses non-dominated sorting and crowding distance mechanisms to update the population) for solving OneMinMax and LOTZ. Later on, Zheng and Doerr [2022] considered a modified crowding distance method, which updates the crowding distance of solutions once the solution with the smallest crowding distance is removed, and proved that the modified method can approximate the Pareto front better than the original crowding distance method in NSGA-II. Bian and Qian [2022] proposed a new parent selection method, stochastic tournament selection (i.e., k tournament selection where k is uniformly sampled at random), to replace the binary tournament selection of NSGA-II, and proved that the method can decrease the expected running time asymptotically. There is also some work analyzing the effectiveness of the mutation [Doerr and Qu, 2022a] and crossover [Doerr and Qu, 2022c; Dang *et al.*, 2023] operators in NSGA-II. The lower bounds of NSGA-II solving OneMinMax and OneJumpZeroJump have also been analyzed [Doerr and Qu, 2022b].

Our running time analysis about SMS-EMOA contributes to the theoretical understanding of another major type of MOEAs, i.e., combining non-dominated sorting and quality indicators to update the population, for the first time. More importantly, our work presents a potential drawback of an important component, i.e., population update, in existing MOEAs. That is, instead of updating the population in a deterministic way, introducing randomness may be helpful for the search, even reducing the expected running time of MOEAs exponentially. Though being proved in a special case, this finding may hold more generally, and may inspire the design of more efficient MOEAs in practice.

2 Preliminaries

In this section, we first introduce multi-objective optimization and SMS-EMOA, and then present a new population update method, i.e., stochastic population update. Finally, we introduce the OneJumpZeroJump problem studied in this paper.

2.1 Multi-objective Optimization

Multi-objective optimization aims to optimize two or more objective functions simultaneously, as shown in Definition 1. Note that in this paper, we consider maximization (minimization can be defined similarly), and pseudo-Boolean functions, i.e., the solution space $\mathcal{X} = \{0,1\}^n$. The objectives are usually conflicting, thus there is no canonical complete order in the solution space $\mathcal{X}$, and we use the *domination* relationship in Definition 2 to compare solutions. A solution is *Pareto optimal* if there is no other solution in $\mathcal{X}$ that dominates it, and the set of objective vectors of all the Pareto optimal solutions constitutes the *Pareto front*. The goal of multi-objective optimization is to find the Pareto front or its good approximation.

Definition 1 (Multi-objective Optimization). *Given a feasible solution space $\mathcal{X}$ and objective functions $f_1, f_2, \ldots, f_m$, multi-objective optimization can be formulated as*

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

Definition 2 (Domination). *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 $\boldsymbol{y} \succeq \boldsymbol{x}$.*

Algorithm 1 SMS-EMOA

```
Input: objective functions $f_1, f_2 \ldots, f_m$, population size $\mu$
Output: $\mu$ 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:     select a solution $\boldsymbol{x}$ from $P$ uniformly at random;
4:     generate $\boldsymbol{x}'$ by flipping each bit of $\boldsymbol{x}$ independently with probability $1/n$;
5:     $P \leftarrow$ POPULATION UPDATE $(P \cup \{\boldsymbol{x}'\})$
6: end while
7: return $P$
```

Algorithm 2 POPULATION UPDATE (Q)

```
Input: a set $Q$ of solutions, and a reference point $\boldsymbol{r} \in \mathbb{R}^m$
Output: $|Q|-1$ solutions from $Q$
1: partition $Q$ into non-dominated sets $R_1, R_2, \ldots, R_v$;
2: let $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$;
3: return $Q \setminus \{\boldsymbol{z}\}$
```

2.2 SMS-EMOA

The SMS-EMOA algorithm [Beume *et al.*, 2007] as presented in Algorithm 1 is a popular MOEA, which employs non-dominated sorting and hypervolume indicator to evaluate the quality of a solution and update the population. SMS-EMOA starts from an initial population of μ solutions (line 1). In each generation, it randomly selects a solution from the current population (line 3), and then applies bit-wise mutation to generate an offspring solution (line 4). Then, the worst solution in the union of the current population P and the newly generated solution is removed (line 5), by using the POPULATION UPDATE subroutine as presented in Algorithm 2.

As can be seen in Algorithm 2, the POPULATION UPDATE subroutine first partitions a set Q of solutions (where $Q = P \cup \{\boldsymbol{x}'\}$ in Algorithm 1) into non-dominated sets $R_1, R_2, \ldots, R_v$ (line 1), where R_1 contains all the non-dominated solutions in Q, and R_i ($i \geq 2$) contains all the non-dominated solutions in $Q \setminus \cup_{j=1}^{i-1} R_j$. Then, one solution $\boldsymbol{z} \in R_v$ that minimizes $\Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v) := HV_{\boldsymbol{r}}(R_v) - HV_{\boldsymbol{r}}(R_v \setminus \{\boldsymbol{x}\})$ is removed (lines 2–3), where $HV_{\boldsymbol{r}}(R_v) = \Lambda\big(\cup_{\boldsymbol{x} \in R_v} \{\boldsymbol{f}' \in \mathbb{R}^m \mid \forall 1 \leq i \leq m : r_i \leq f'_i \leq f_i(\boldsymbol{x})\}\big)$ denotes the hypervolume of R_v with respect to a reference point $\boldsymbol{r} \in \mathbb{R}^m$ (satisfying $\forall 1 \leq i \leq m, r_i \leq \min_{\boldsymbol{x} \in \mathcal{X}} f_i(\boldsymbol{x})$), and Λ denotes the Lebesgue measure. The hypervolume of a solution set measures the volume of the objective space between the reference point and the objective vectors of the solution set, and a larger hypervolume value implies a better approximation ability with regards to both convergence and diversity. This implies that the solution with the least value of Δ in the last non-dominated set R_v can be regarded as the worst solution in Q, which is thus removed.

2.3 Stochastic Population Update

Previous MOEAs only considered deterministic and greedy population update methods, i.e., a dominating solution or a solution with better indicator value is always preferred. However, these methods may be too greedy, and thus limit the performance of MOEAs. In this paper, we introduce STOCHASTIC POPULATION UPDATE as presented in Algorithm 3 into SMS-MOEA, to replace the original POPULATION UPDATE procedure in line 5 of Algorithm 1. These two procedures are similar, except that the removed solution is selected from a subset Q' of Q in Algorithm 3, instead of from the entire set Q, where Q' consists of N solutions chosen randomly from Q. Note that we set $N = \lfloor |Q|/2 \rfloor$ here for examination, while other values of N can also be used in practical applications.

Algorithm 3 STOCHASTIC POPULATION UPDATE (Q)

```
Input: a set $Q$ of solutions, and a reference point $\boldsymbol{r} \in \mathbb{R}^m$
Output: $|Q|-1$ solutions from $Q$
1: $Q' \leftarrow \lfloor |Q|/2 \rfloor$ solutions uniformly and randomly selected from $Q$ without replacement;
2: partition $Q'$ into non-dominated sets $R_1, R_2, \ldots, R_v$;
3: let $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$;
4: return $Q \setminus \{\boldsymbol{z}\}$
```

2.4 OneJumpZeroJump Problem

The OneJumpZeroJump problem is a multi-objective counterpart of the Jump problem, a classical single-objective pseudo-Boolean benchmark problem in EAs' theoretical analyses [Doerr and Neumann, 2020]. The goal of the Jump problem 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. Formally, it aims to find an n-bits binary string which maximizes

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

where $k \in [2..n-1]$, and $|\boldsymbol{x}|_1$ denotes the number of 1-bits in $\boldsymbol{x}$. Note that we use $[a..b]$ (where $a, b \in \mathbb{N}, a \leq b$) to denote the set $\{a, a+1, \ldots, b\}$ of integers throughout the paper.

The OneJumpZeroJump problem as presented in Definition 3 is constructed based on the Jump problem, and has been widely used in MOEAs' theoretical analyses [Doerr and Zheng, 2021; Doerr and Qu, 2022a; Doerr and Qu, 2022b; Doerr and Qu, 2022c]. Its first objective 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. Figure 1 illustrates the values of f_1 and f_2 with respect to the number of 1-bits of a solution.

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

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

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

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.

According to Theorem 5 of [Doerr and Zheng, 2021], the Pareto set of the OneJumpZeroJump problem is

$$S^* = \{\boldsymbol{x} \in \{0,1\}^n \mid |\boldsymbol{x}|_1 \in [k..n-k] \cup \{0, n\}\},$$

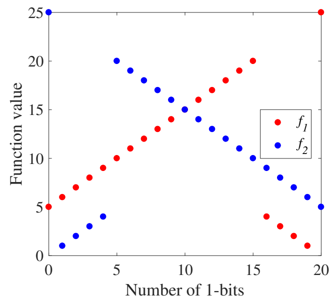

Figure 1: The function value of the OneJumpZeroJump problem vs. the number of 1-bits of a solution when $n = 20$ and $k = 5$.

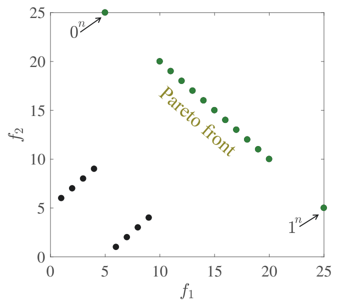

Figure 2: The objective vectors of the OneJumpZeroJump problem when $n = 20$ and $k = 5$, where the set of all the green points denotes the Pareto front.

and the Pareto front is

$$F^* = \boldsymbol{f}(S^*) = \{(a, n+2k-a) \mid a \in [2k..n] \cup \{k, n+k\}\},$$

whose size is $n - 2k + 3$. Figure 2 illustrates the objective vectors and the Pareto front. We use

$$S_I^* = \{\boldsymbol{x} \in S^* \mid |\boldsymbol{x}|_1 \in [k..n-k]\} \quad (1)$$

and

$$F_I^* = \boldsymbol{f}(S_I^*) = \{(a, 2k+n-a) \mid a \in [2k..n]\} \quad (2)$$

to denote the inner part of the Pareto set and Pareto front, respectively. Note that each objective value of the OneJumpZeroJump problem must be larger than zero, thus we set the reference point $\boldsymbol{r} = (0, 0)$ in SMS-EMOA.

3 Running Time Analysis of SMS-EMOA

In this section, we analyze the expected running time of SMS-EMOA in Algorithm 1 for solving the OneJumpZeroJump 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. As SMS-EMOA generates only one offspring solution in each generation, its running time is just equal to the number of generations. We prove in Theorems 1 and 2 that the upper and lower bounds on the expected number of generations of SMS-EMOA solving OneJumpZeroJump are $O(\mu n^k)$ and $\Omega(n^k)$, respectively. Note that we use $poly(n)$ to denote any polynomial of n.

Theorem 1 (Upper bound)**.** *For SMS-EMOA solving OneJumpZeroJump, if using a population size μ such that $n - 2k + 3 \leq \mu$, then the expected number of generations for finding the Pareto front is $O(\mu n^k)$.*

Theorem 2 (Lower bound)**.** *For SMS-EMOA solving OneJumpZeroJump with $n - 2k = \Theta(n)$, if using a population size μ such that $n - 2k + 3 \leq \mu = poly(n)$, then the expected number of generations for finding the Pareto front is $\Omega(n^k)$.*

The proof of Theorem 1 relies on Lemma 1, which shows that once an objective vector $\boldsymbol{f}^*$ in the Pareto front is found, it will always be maintained, i.e., there will always exist a solution whose objective vector is $\boldsymbol{f}^*$ in the population.

Lemma 1. *For SMS-EMOA solving OneJumpZeroJump, if using a population size μ such that $n - 2k + 3 \leq \mu$, then an objective vector $\boldsymbol{f}^*$ in the Pareto front will always be maintained once it has been found.*

Proof. Suppose the objective vector $(a, n+2k-a), a \in [2k..n] \cup \{k, n+k\}$, in the Pareto front is obtained by SMS-EMOA, i.e., there exists a solution $\boldsymbol{x}$ in Q (i.e., $P \cup \{\boldsymbol{x}'\}$ in line 5 of Algorithm 1) such that $\boldsymbol{f}(\boldsymbol{x}) = (a, n+2k-a)$. If at least two solutions correspond to the objective vector, then at least one of them will still be maintained in the next generation because only one solution is removed by Algorithm 2.

If only one solution (denoted as $\boldsymbol{y}$) corresponds to the objective vector, then we have $\boldsymbol{y} \in R_1$ in the POPULATION UPDATE procedure because $\boldsymbol{y}$ cannot be dominated by any other solution; and $\Delta_{\boldsymbol{r}}(\boldsymbol{y}, R_1) = HV_{\boldsymbol{r}}(R_1) - HV_{\boldsymbol{r}}(R_1 \setminus \{\boldsymbol{y}\}) > 0$ because the region

$$\{\boldsymbol{f}' \in \mathbb{R}^2 \mid a-1 < f_1' \leq a, n+2k-a-1 < f_2' \leq n+2k-a\}$$

cannot be covered by any objective vector in $\boldsymbol{f}(\{0,1\}^n) \setminus \{(a, n+2k-a)\}$. Then, we consider two cases.

(1) There exists one solution $\boldsymbol{x}$ in R_1 such that $k \leq |\boldsymbol{x}|_1 \leq n-k$. Then, R_1 cannot contain solutions whose number of 1-bits are in $[1..k-1] \cup [n-k+1..n-1]$, because these solutions must be dominated by $\boldsymbol{x}$. If at least two solutions in Q have the same objective vector, then they must have a zero Δ value, because removing one of them will not decrease the hypervolume covered. Thus, for each objective vector $(b, n+2k-b), b \in [2k..n] \cup \{k, n+k\}$, at most one solution can have a Δ value larger than zero, implying that there exist at most $n - 2k + 3 \leq \mu$ solutions in R_1 with $\Delta > 0$.

(2) Any solution $\boldsymbol{x}$ in R_1 satisfies $|\boldsymbol{x}|_1 < k$ or $|\boldsymbol{x}|_1 > n-k$. Note that for the solutions with the number of 1-bits in $[1..k-1]$, a solution with more 1-bits must dominate a solution with less 1-bits. Meanwhile, two solutions with the same number of 1-bits will have a zero Δ value, thus R_1 can only contain at most one solution in $\{\boldsymbol{x} \mid |\boldsymbol{x}|_1 \in [1..k-1]\}$ with Δ value larger than zero. Similarly, at most one solution in $\{\boldsymbol{x} \mid |\boldsymbol{x}|_1 \in [n-k+1..n-1]\}$ with Δ value larger than zero belongs to R_1. For solutions with number of 1-bits in $\{0, n\}$ (note that there may exist reduplicative solutions in Q), it is also straightforward to see that at most two of them can have a Δ value larger than zero. By the problem setting $k < n/2$, we have $\mu \geq n - 2k + 3 \geq 4$, thus there exist at most μ solutions in R_1 with Δ value larger than zero.

Combining the above two cases, we show that there exist at most μ solutions in R_1 with Δ value larger than zero, implying that $\boldsymbol{y}$ will still be maintained in the next generation. □

The main proof idea of Theorem 1 is to divide the optimization procedure into two phases, where the first phase aims at finding the inner part F_I^* (in Eq. (2)) of the Pareto front, and the second phase aims at finding the remaining two extreme vectors in the Pareto front, i.e., $F^* \setminus F_I^* = \{(k, n+k), (n+k, k)\}$, corresponding to the two Pareto optimal solutions 0^n and 1^n.

Proof of Theorem 1. We divide the optimization procedure into two phases. The first phase starts after initialization and finishes until all the objective vectors in the inner part F_I^* of the Pareto front has been found; the second phase starts after the first phase and finishes when the whole Pareto front is found. We will show that the expected number of generations of the two phases are $O(\mu(n \log n + k^k))$ and $O(\mu n^k)$, respectively, leading to the theorem.

For the first phase, we consider two cases.

(1) At least one solution in the inner part S_I^* of the Pareto set exists in the initial population. Let

$$D_1 = \{\boldsymbol{x} \in P \mid (H^+(\boldsymbol{x}) \cap P = \emptyset \wedge H^+(\boldsymbol{x}) \cap S_I^* \neq \emptyset) \vee (H^-(\boldsymbol{x}) \cap P = \emptyset \wedge H^-(\boldsymbol{x}) \cap S_I^* \neq \emptyset)\},$$

where $H^+(\boldsymbol{x}) := \{\boldsymbol{x}' \mid |\boldsymbol{x}'|_1 = |\boldsymbol{x}|_1 + 1\}$ and $H^-(\boldsymbol{x}) := \{\boldsymbol{x}' \mid |\boldsymbol{x}'|_1 = |\boldsymbol{x}|_1 - 1\}$ denote the Hamming neighbours of $\boldsymbol{x}$ with one more 1-bit and one less 1-bit, respectively. Intuitively, D_1 denotes the set of solutions in P whose Hamming neighbour is Pareto optimal but not contained by P. Then, by selecting a solution $\boldsymbol{x} \in D_1$, and flipping one of the 0-bits or one of the 1-bits, a new objective vector in F_I^* can be obtained. By Lemma 1, one solution corresponding to the new objective vector will always be maintained in the population. By repeating the above procedure, the whole set F_I^* can be found. Note that the probability of selecting a specific solution in P is $1/\mu$, and the probability of flipping one of the 0-bits (or 1-bits) is $(n - |\boldsymbol{x}|_1) \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq (n - |\boldsymbol{x}|_1)/(en)$ (or $(|\boldsymbol{x}|_1/n) \cdot (1-1/n)^{n-1} \geq |\boldsymbol{x}|_1/(en)$). Thus, the total expected number of generations for finding F_I^* is at most $\sum_{i=k+1}^{|\boldsymbol{x}|_1} (\mu e n)/i + \sum_{i=k+1}^{n-|\boldsymbol{x}|_1} (\mu e n)/i = O(\mu n \log n)$.

(2) Any solution in the initial population has at most $(k-1)$ 1-bits or at least $(n-k+1)$ 1-bits. Without loss of generality, we can assume that one solution $\boldsymbol{y}$ has at most $(k-1)$ 1-bits. Then, selecting $\boldsymbol{y}$ and flipping $(k - |\boldsymbol{y}|_1)$ 0-bits can generate a solution in S_I^*, whose probability is at least

$$\frac{1}{\mu} \cdot \frac{\binom{n-|\boldsymbol{y}|_1}{k-|\boldsymbol{y}|_1}}{n^{k-|\boldsymbol{y}|_1}} \cdot \left(1 - \frac{1}{n}\right)^{n-k+|\boldsymbol{y}|_1} \geq \frac{1}{e\mu} \cdot \frac{\binom{n-|\boldsymbol{y}|_1}{k-|\boldsymbol{y}|_1}}{n^{k-|\boldsymbol{y}|_1}}.$$

Let $g(i) = \binom{n-i}{k-i}/n^{k-i}, 0 \leq i \leq k-1$, then we have

$$\frac{g(i+1)}{g(i)} = \frac{\binom{n-i-1}{k-i-1}}{\binom{n-i}{k-i}} \cdot n = \frac{n(k-i)}{(n-i)} \geq 1,$$

implying

$$\frac{\binom{n-|\boldsymbol{y}|_1}{k-|\boldsymbol{y}|_1}}{n^{k-|\boldsymbol{y}|_1}} = g(|\boldsymbol{y}|_1) \geq g(0) = \frac{\binom{n}{k}}{n^k} \geq \left(\frac{n}{k}\right)^k \cdot \frac{1}{n^k} = \frac{1}{k^k}.$$

Thus, the expected number of generations for finding a solution in S_I^* is at most $e\mu k^k$. By Lemma 1, the generated solution must be included into the population. Thus, combining the analysis for case (1), we can derive that the total expected number of generations for finding F_I^* is $O(\mu(n \log n + k^k))$.

For the second phase, we need to find the two extreme solutions 1^n and 0^n. To find 1^n (or 0^n), it is sufficient to select the solution in the population P with $(n-k)$ 1-bits (or k 1-bits) and flip its k 0-bits (or k 1-bits), whose probability is $(1/\mu) \cdot (1/n^k) \cdot (1 - 1/n)^{n-k} \geq 1/(e\mu n^k)$. Thus, the expected number of generations is $O(e\mu n^k)$.

Combining the derived upper bounds on the expected number of generations of the two phases, the theorem holds. $\square$

The proof idea of Theorem 2 is that all the solutions in the initial population belong to the inner part S_I^* (in Eq. (1)) of the Pareto set with probability $\Theta(1)$, and then SMS-EMOA requires $\Omega(n^k)$ expected number of generations to find the two extreme Pareto optimal solutions 1^n and 0^n.

Proof of Theorem 2. Let G denote the event that all the solutions in the initial population belong to S_I^*, i.e., for any solution $\boldsymbol{x}$ in the initial population, $k \leq |\boldsymbol{x}|_1 \leq n-k$. We first show that event G happens with probability $1 - o(1)$. For an initial solution $\boldsymbol{y}$, it is generated uniformly at random, i.e., each bit in $\boldsymbol{y}$ can be 1 or 0 with probability $1/2$, respectively. Thus, the expected number of 1-bits in $\boldsymbol{y}$ is exactly $n/2$. By Hoeffding's inequality and the condition $n - 2k = \Theta(n)$ of the theorem, we have

$$\Pr\left(\left||\boldsymbol{y}|_1 - \frac{n}{2}\right| > \frac{n}{2} - k\right) < 2e^{-2(n/2-k)^2/n} = e^{-\Theta(n)}.$$

Then, we can derive that

$$\Pr\left(\forall \boldsymbol{x} \text{ in the initial population}, \left||\boldsymbol{x}|_1 - \frac{n}{2}\right| \leq \frac{n}{2} - k\right) \geq \left(1 - e^{-\Theta(n)}\right)^\mu \geq 1 - \mu \cdot e^{-\Theta(n)} = 1 - o(1),$$

where the last inequality holds by Bernoulli's inequality, and the equality holds by the condition $\mu = poly(n)$.

Next we show that given event G, the expected number of generations for finding the whole Pareto front is at least n^k. Starting from the initial population, if a solution $\boldsymbol{x}$ with $1 \leq |\boldsymbol{x}|_1 \leq k-1$ or $n-k+1 \leq |\boldsymbol{x}|_1 \leq n-1$ is generated in some generation, it must be deleted because it is dominated by any of the solutions in the current population. Thus, the extreme solution 1^n can only be generated by selecting a solution in S_I^* and flipping all the 0-bits, whose probability is at most $1/n^k$. Thus, the expected number of generations for finding 1^n is at least n^k.

Combining the above analyses, the expected number of generations for finding the whole Pareto front is at least $(1 - o(1)) \cdot n^k = \Omega(n^k)$. $\square$

4 Analysis of SMS-EMOA Using Stochastic Population Update

In the previous section, we have proved that the expected number of generations of SMS-EMOA is $O(\mu n^k)$ and $\Omega(n^k)$ for solving OneJumpZeroJump. Next, we will show that by employing stochastic population update in Algorithm 3, instead of the original population update procedure in Algorithm 2, SMS-EMOA can use much less time to find the whole Pareto front.

In particular, we prove in Theorem 3 that the expected number of generations of SMS-EMOA using stochastic population update is $O(\mu n^k \cdot \min\{1, \mu/2^{k/4}\})$ for solving the OneJumpZeroJump problem when $2(n-2k+4) \le \mu$. Recall from Theorem 2 that the expected number of generations of the original SMS-EMOA is $\Omega(n^k)$ when $n-2k = \Theta(n)$ and $n-2k+3 \le \mu = poly(n)$. Thus, when $k = \Omega(n) \wedge k = n/2 - \Theta(n)$ and $2(n-2k+4) \le \mu = poly(n)$, using stochastic population update can bring an acceleration of at least $\Omega(2^{k/4}/\mu^2)$, implying an exponential acceleration. The main reason for the acceleration is that introducing randomness into the population update procedure allows a bad solution, i.e., a solution $\boldsymbol{x}$ with $|\boldsymbol{x}|_1 \in [1..k-1] \cup [n-k+1..n-1]$, to be included into the population with some probability, thus making SMS-EMOA much easier to generate the two extreme Pareto optimal solutions 0^n and 1^n.

Theorem 3. *For SMS-EMOA solving OneJumpZeroJump, if using stochastic population update in Algorithm 3, and a population size μ such that $2(n-2k+4) \le \mu$, then the expected number of generations for finding the Pareto front is $O(\mu n^k \cdot \min\{1, \mu/2^{k/4}\})$.*

Before proving Theorem 3, we first present Lemma 2, which shows that given proper value of μ, an objective vector in the Pareto front will always be maintained once it has been found; any solution (even the worst solution) in the collections of the current population and the newly generated offspring solution can survive in the population update procedure with probability at least $1/2$.

Lemma 2. *For SMS-EMOA solving OneJumpZeroJump, if using stochastic population update in Algorithm 3, and a population size μ such that $2(n-2k+4) \le \mu$, then*

(1) an objective vector $\boldsymbol{f}^*$ in the Pareto front will always be maintained once it has been found;

(2) any solution in $P \cup \{\boldsymbol{x}'\}$ can be maintained in the next population with probability at least $1/2$, where P denotes the current population and $\boldsymbol{x}'$ denotes the offspring solution produced in the current generation.

Proof. The proof of the first clause is similar to that of Lemma 1, and we only need to consider the case that only one solution $\boldsymbol{x}^*$ corresponding to $\boldsymbol{f}^*$ exists in $Q = P \cup \{\boldsymbol{x}'\}$. By the proof of Lemma 1, there exist at most $n-2k+3$ solutions in R_1 with Δ value larger than zero. Note that the removed solution is chosen from $\lfloor(\mu+1)/2\rfloor \ge n-2k+4$ solutions in Q, thus $\boldsymbol{x}^*$ will not be removed because it is one of the best $n-2k+3$ solutions. Then, the first clause holds.

The second clause holds, because for any solution in $P \cup \{\boldsymbol{x}'\}$, it can be removed only if it is chosen for competition, whose probability is at most $\lfloor(\mu+1)/2\rfloor/(\mu+1) \le 1/2$. □

The basic proof idea of Theorem 3 is similar to that of Theorem 1, i.e., dividing the optimization procedure into two phases, which are to find F_I^* and $F^* \setminus F_I^* = \{(k, n+k), (n+k, k)\}$, respectively. However, the analysis for the second phase is a little more sophisticated here, because dominated solutions can be included into the population when using stochastic population update, leading to a more complicated behavior of SMS-EMOA.

We will use the additive drift analysis tool in Lemma 3 to derive an upper bound on the expected number of generations of the second phase. Because the population of SMS-EMOA in the $(t+1)$-th generation only depends on the t-th population, its process can be naturally modeled as a Markov chain. Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ and $\xi_{\hat{t}} = \mathrm{x} \in \mathcal{X}$, we define its *first hitting time* as $\tau = \min\{t \mid \xi_{\hat{t}+t} \in \mathcal{X}^*, t \ge 0\}$, where $\mathcal{X}$ and $\mathcal{X}^*$ denote the state space and target state space of the Markov chain, respectively. For the analysis in Theorem 3, $\mathcal{X}$ denotes the set of all the populations after phase 1, and $\mathcal{X}^*$ denotes the set of all the populations which contain the Pareto optimal solution 1^n (or 0^n). Let $\mathrm{E}(\cdot)$ denote the expectation of a random variable. The mathematical expectation of τ, $\mathrm{E}(\tau \mid \xi_{\hat{t}} = \mathrm{x}) = \sum_{i=0}^{+\infty} i \cdot \mathrm{P}(\tau = i \mid \xi_{\hat{t}} = \mathrm{x})$, is called the *expected first hitting time* (EFHT) starting from $\xi_{\hat{t}} = \mathrm{x}$. The additive drift as presented in Lemma 3 is used to derive upper bounds on the EFHT of Markov chains. To use it, a function $V(\mathrm{x})$ has to be constructed to measure the distance of a state x to the target state space $\mathcal{X}^*$, where $V(\mathrm{x} \in \mathcal{X}^*) = 0$ and $V(\mathrm{x} \notin \mathcal{X}^*) > 0$. Then, we need to investigate the progress on the distance to $\mathcal{X}^*$ in each step, i.e., $\mathrm{E}(V(\xi_t) - V(\xi_{t+1}) \mid \xi_t)$. An upper bound on the EFHT can be derived through dividing the initial distance by a lower bound on the progress.

Lemma 3 (Additive Drift [He and Yao, 2001])**.** *Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ and a distance function $V(\cdot)$, if for any $t \ge 0$ and any ξ_t with $V(\xi_t) > 0$, there exists a real number $c > 0$ such that $\mathrm{E}(V(\xi_t) - V(\xi_{t+1}) \mid \xi_t) \ge c$, then the EFHT satisfies that $\mathrm{E}(\tau \mid \xi_0) \le V(\xi_0)/c$.*

Proof of Theorem 3. Similar to the proof of Theorem 1, we divide the optimization procedure into two phases. That is, the first phase starts after initialization and finishes until all the objective vectors in F_I^* have been found; the second phase starts after the first phase and finishes when 0^n and 1^n are also found. The analysis of the first phase is the same as that of Theorem 1, because the objective vectors in F_I^* will always be maintained by Lemma 2. That is, the expected number of generations of phase 1 is $O(\mu(n \log n + k^k))$.

Now we analyze the second phase. Without loss of generality, we only consider the expected number of generations for finding 1^n, and the same bound holds for finding 0^n analogously. We use Lemma 3, i.e., additive drift analysis, to prove. Note that the process of SMS-EMOA can be directly modeled as a Markov chain by letting the state of the chain represent a population of SMS-EMOA. Furthermore, the target space consists of all the populations which contain 1^n. In the following, we don't distinguish a state from its corresponding population. First, we construct the distance function

$$V(P) = \begin{cases} 0 & \text{if } \max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_1 = n, \\ e\mu n^{k/2} & \text{if } n - \frac{k}{2} \le \max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_1 \le n-1, \\ e\mu n^{k/2} + 1 & \text{if } n-k \le \max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_1 < n - \frac{k}{2}. \end{cases}$$

It is easy to verify that $V(P) = 0$ if and only if $1^n \in P$.

Then, we examine $\mathrm{E}(V(\xi_t) - V(\xi_{t+1}) \mid \xi_t = P)$ for any P with $1^n \notin P$. Assume that currently $\max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_1 = q$, where $n-k \le q \le n-1$. We first consider the case that $n - k/2 \le q \le n-1$. To make V decrease, it is sufficient to select the solution in P with q 1-bits and flip its remaining $(n-q)$

0-bits, whose probability is $(1/\mu)\cdot(1/n^{n-q})\cdot(1-1/n)^q \geq 1/(e\mu n^{n-q}) \geq 1/(e\mu n^{k/2})$, where the last inequality is by $n-k/2 \leq q$. Note that the newly generated solution is 1^n, which must be included in the population. In this case, V can decrease by $e\mu n^{k/2}$. To make V increase, the solution in P with q 1-bits needs to removed in the next generation, whose probability is at most $1/2$ by Lemma 2. In this case, V can increase by $e\mu n^{k/2}+1-e\mu n^{k/2}=1$. Thus, we have

$$\mathrm{E}(V(\xi_t)-V(\xi_{t+1}) \mid \xi_t = P) \geq \frac{e\mu n^{k/2}}{e\mu n^{k/2}} - \frac{1}{2}\cdot 1 \geq \frac{1}{2}. \quad (3)$$

Now we consider the case that $n-k \leq q < n-k/2$. Note that in this case, V cannot increase, thus we only need to consider the decrease of V. We further consider two subcases.
(1) $q > n-3k/4$. To make V decrease by 1, it is sufficient to select the solution with q 1-bits, flip $k/4$ 0-bits among the $(n-q)$ 0-bits, and include the newly generated solution into the population, whose probability is at least $(1/\mu)\cdot(\binom{n-q}{k/4}/n^{k/4})\cdot(1-1/n)^{n-k/4}\cdot(1/2) \geq \binom{k/2}{k/4}/(2e\mu n^{k/4})$. Thus, $\mathrm{E}(V(\xi_t)-V(\xi_{t+1}) \mid \xi_t = P) \geq \binom{k/2}{k/4}/(2e\mu n^{k/4})$.
(2) $q \leq n-3k/4$. To make V decrease by 1, it is sufficient to select the solution with q 1-bits, flip $(n-k/2-q)$ 0-bits among the $(n-q)$ 0-bits, and include the newly generated solution into the population, whose probability is at least

$$\frac{1}{\mu}\cdot\frac{\binom{n-q}{n-k/2-q}}{n^{n-k/2-q}}\cdot\left(1-\frac{1}{n}\right)^{k/2+q}\cdot\frac{1}{2}$$
$$\geq \frac{\binom{n-q}{k/2}}{2e\mu n^{k/2}} \geq \frac{\binom{3k/4}{k/2}}{2e\mu n^{k/2}} \geq \frac{\binom{k/2}{k/4}}{2e\mu n^{k/2}},$$

where the first inequality is by $n-k \leq q$. Thus, $\mathrm{E}(V(\xi_t)-V(\xi_{t+1}) \mid \xi_t = P) \geq \binom{k/2}{k/4}/(2e\mu n^{k/2})$.
Combining subcases (1) and (2), we can derive

$$\mathrm{E}(V(\xi_t)-V(\xi_{t+1}) \mid \xi_t = P) \geq \frac{\binom{k/2}{k/4}}{2e\mu n^{k/2}} \geq \frac{2^{k/4}}{2e\mu n^{k/2}}. \quad (4)$$

By Eqs. (3) and (4), we have $\mathrm{E}(V(\xi_t)-V(\xi_{t+1}) \mid \xi_t = P) \geq 2^{k/4}/(2e\mu n^{k/2})$. Then, by Lemma 3 and $V(P) \leq e\mu n^{k/2}+1$, the expected number of generations for finding 1^n is at most $(e\mu n^{k/2}+1)\cdot(2e\mu n^{k/2})/2^{k/4} = O(\mu^2 n^k/2^{k/4})$.

Thus, combining the two phases, the expected number of generations for finding the whole Pareto front is $O(\mu(n\log n+k^k)) + O(\mu^2 n^k/2^{k/4}) = O(\mu^2 n^k/2^{k/4})$, where the equality holds by $2 \leq k < n/2$.

Now we analyze the expected number of generations from another perspective. To generate 1^n, it is sufficient to select a solution with $(n-k)$ 1-bits and flip the remaining k 0-bits, whose probability is $(1/\mu)\cdot(1/n^k)\cdot(1-1/n)^{n-k} \geq 1/(e\mu n^k)$. Thus, the expected number of generations of phase 2 can also be upper bounded by $O(\mu n^k)$, implying that the total expected number of generations for finding the Pareto front is $O(\mu(n\log n+k^k)) + O(\mu n^k) = O(\mu n^k)$.

Thus, the expected number of generations of SMS-EMOA using stochastic population update for finding the Pareto front is $\min\{O(\mu^2 n^k/2^{k/4}), O(\mu n^k)\} = O(\mu n^k \cdot \min\{1, \mu/2^{k/4}\})$, implying that the theorem holds. □

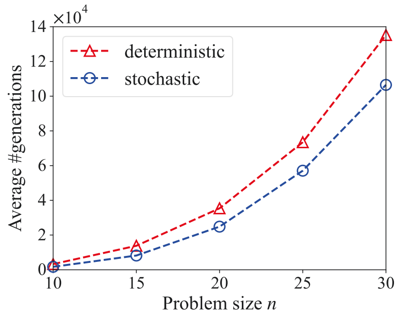

Figure 3: Average #generations of SMS-EMOA using deterministic and stochastic population update methods for solving the OneJumpZeroJump problem.

5 Experiments

In the previous sections, we have proved that the stochastic population update method can bring significant acceleration for large k. However, it is unclear whether it can still perform better for small k. We empirically examine this case here.

Specifically, we compare the number of generations of SMS-EMOA for solving OneJumpZeroJump, when the two population update methods are used, respectively. We set k to 2, the problem size n from 10 to 30, with a step of 5, and the population size $\mu = 2(n-2k+4)$, as suggested in Theorem 3. For each n, we run SMS-EMOA 1000 times independently, and record the number of generations until the Pareto front is found. Figure 3 shows the average number of generations of the 1000 runs. Note that the standard deviation is not included because it is very close to the mean and may make the figure look cluttered. We can observe that the stochastic population update method can bring a clear acceleration. Note that using non-dominated sorting and hypervolume indicator to rank solutions may also be time-consuming, thus the stochastic population update method which only requires to compare $\lfloor |P \cup \{\boldsymbol{x}'\}|/2 \rfloor = \lfloor(\mu+1)/2\rfloor$ solutions in each generation can make the optimization procedure even faster.

6 Conclusion

In this paper, we challenge a common practice for a key component in MOEAs, population update, through rigorous theoretical analysis. Existing MOEAs always update the population in a deterministic manner, while we prove that for the well-known SMS-EMOA solving the bi-objective problem OneJumpZeroJump, introducing randomness into the population update procedure can significantly decrease the expected running time. Though being proved in a special case, this finding may hold more generally. In fact, our results echo some recent empirical studies [Liang *et al.*, 2023], where a simple non-elitist MOEA (which is of the stochastic population update nature) has been found to outperform SMS-EMOA on popular practical problems like the knapsack problem [Zitzler and Thiele, 1999] and NK-landscape [Aguirre and Tanaka, 2004].

References

[Aguirre and Tanaka, 2004] H. E. Aguirre and K. Tanaka. Insights on properties of multiobjective MNK-landscapes. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'04)*, pages 196–203, 2004.

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

[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 and Qian, 2022] C. Bian and C. Qian. Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature (PPSN'22)*, pages 428–441, Dortmund, Germany, 2022.

[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 International Joint Conference on Artificial Intelligence (IJCAI'18)*, pages 1405–1411, Stockholm, Sweden, 2018.

[Brockhoff *et al.*, 2008] D. Brockhoff, T. Friedrich, and F. Neumann. Analyzing hypervolume indicator based algorithms. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN'08)*, pages 651–660, Dortmund, Germany, 2008.

[Coello Coello and Lamont, 2004] C. A. Coello Coello and G. B. Lamont. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004.

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

[Dang *et al.*, 2023] D.-C. Dang, A. Opris, B. Salehi, and D. Sudholt. A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. *CORR abs/2301.13687*, 2023.

[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.

[Deb, 2011] K. Deb. Multi-objective optimisation using evolutionary algorithms: An introduction. In L. Wang, A. H. C. Ng, and K. Deb, editors, *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 3–34. Springer, 2011.

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

[Doerr and Qu, 2022a] B. Doerr and Z. Qu. A first runtime analysis of the NSGA-II on a multimodal problem. In *Proceedings of the 17th International Conference on Parallel Problem Solving from Nature (PPSN'22)*, pages 399–412, Dortmund, Germany, 2022.

[Doerr and Qu, 2022b] B. Doerr and Z. Qu. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. *CORR abs/2209.13974*, 2022.

[Doerr and Qu, 2022c] B. Doerr and Z. Qu. Runtime analysis for the NSGA-II: Provable speed-ups from crossover. *CORR abs/2208.08759*, 2022.

[Doerr and Zheng, 2021] B. Doerr and W. Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Proceedings of the 35-th AAAI Conference on Artificial Intelligence (AAAI'21)*, pages 12293–12301, Virtual, 2021.

[Doerr *et al.*, 2013] B. Doerr, B. Kodric, and M. Voigt. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC'13)*, pages 432–439, Cancun, Mexico, 2013.

[Doerr *et al.*, 2016] B. Doerr, W. Gao, and F. Neumann. Runtime analysis of evolutionary diversity maximization for OneMinMax. In *Proceedings of the 18th ACM Conference on Genetic and Evolutionary Computation (GECCO'16)*, pages 557–564, 2016.

[Eiben and Smith, 2015] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2015.

[Friedrich *et al.*, 2010] T. Friedrich, N. Hebbinghaus, and F. Neumann. Plateaus can be harder in multi-objective optimization. *Theoretical Computer Science*, 411(6):854–864, 2010.

[Friedrich *et al.*, 2011] T. Friedrich, C. Horoba, and F. Neumann. Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 412(17):1546–1556, 2011.

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

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

[Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addion-Wesley, 1989.

[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.

[Horoba, 2009] C. Horoba. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In *Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA'09)*, pages 113–120, Orlando, FL, 2009.

[Huang *et al.*, 2021] Z. Huang, Y. Zhou, C. Luo, and Q. Lin. A runtime analysis of typical decomposition approaches in MOEA/D framework for many-objective optimization problems. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, pages 1682–1688, Virtual, 2021.

[Laumanns *et al.*, 2004a] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing*, 3:37–51, 2004.

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

[Liang *et al.*, 2023] Z. Liang, M. Li, and P. K. Lehre. Non-elitist evolutionary multi-objective optimisation: Proof-of-principle results. *CORR abs/2305.16870*, 2023.

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

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

[Nguyen *et al.*, 2015] A. Q. Nguyen, A. M. Sutton, and F. Neumann. Population size matters: Rigorous runtime results for maximizing the hypervolume indicator. *Theoretical Computer Science*, 561:24–36, 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, K. Tang, and Z.-H. Zhou. Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In *Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN'16)*, pages 835–846, Edinburgh, Scotland, 2016.

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

[Zheng and Doerr, 2022] W. Zheng and B. Doerr. Better approximation guarantees for the NSGA-II by using the current crowding distance. In *Proceedings of the 24th ACM Conference on Genetic and Evolutionary Computation (GECCO'22)*, pages 611–619, Boston, MA, 2022.

[Zheng *et al.*, 2022] W. Zheng, Y. Liu, and B. Doerr. A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*, pages 10408–10416, Virtual, 2022.

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

[Zitzler and Künzli, 2004] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN'04)*, pages 832–842, Birmingham, UK, 2004.

[Zitzler and Thiele, 1999] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.