A Theoretical Perspective on Why Stochastic Population Update Needs an Archive in Evolutionary Multi-objective Optimization

Shengjie Ren[1,2], **Zimin Liang**[3], **Miqing Li**[3] and **Chao Qian**[1,2]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[2]School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
[3]School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K.
rensj@lamda.nju.edu.cn, zxl525@student.bham.ac.uk, m.li.8@bham.ac.uk, qianc@nju.edu.cn

Abstract

Evolutionary algorithms (EAs) have been widely applied to multi-objective optimization due to their population-based nature. Population update, a key component in multi-objective EAs (MOEAs), is usually performed in a greedy, deterministic manner. However, recent studies have questioned this practice and shown that stochastic population update (SPU), which allows inferior solutions have a chance to be preserved, can help MOEAs jump out of local optima more easily. Nevertheless, SPU risks losing high-quality solutions, potentially requiring a large population. Intuitively, a possible solution to this issue is to introduce an archive that stores the best solutions ever found. In this paper, we theoretically show that using an archive allows a small population and may enhance the search performance of SPU-based MOEAs. We examine two classic algorithms, SMS-EMOA and NSGA-II, on the bi-objective problem OneJumpZeroJump, and prove that using an archive can reduce the expected running time upper bound (even exponentially). The comparison between SMS-EMOA and NSGA-II also suggests that the $(\mu + \mu)$ update mode may be more suitable for SPU than the $(\mu + 1)$ update mode. We also validate our findings empirically. We hope this work may provide theoretical support to explore different ideas of designing algorithms in evolutionary multi-objective optimization.

1 Introduction

Multi-objective optimization deals with scenarios where multiple objectives must be optimized simultaneously. They are very common 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 optimal trade-offs between these objectives, called Pareto optimal solutions. The objective vectors of these solutions form the Pareto front. The goal of multi-objective optimization is to find the Pareto front or a good approximation of it.

Evolutionary algorithms (EAs), a kind of randomized heuristic optimization algorithms inspired by natural evolution, have been found well-suited to MOPs due to their population-based nature. Their widespread applications are across various real-world domains [Deb, 2001; Zhou *et al.*, 2019]. Notably, there have been developed a multitude of well-established multi-objective EAs (MOEAs), including the non-dominated sorting genetic algorithm II (NSGA-II) [Deb *et al.*, 2002], multi-objective evolutionary algorithm based on decomposition (MOEA/D) [Zhang and Li, 2007], and $\mathcal{S}$ metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA) [Beume *et al.*, 2007].

In MOEAs, a key component is population update (aka environmental selection or population maintenance). It aims to select a set of promising solutions from the current population and newly generated solutions, which serves as a reservoir to generate high-quality solutions in subsequent generations. In most existing MOEAs, the population update is performed in a greedy and deterministic manner, with the best solutions (non-dominated solutions) always being preserved. This is based on the assumption that higher-quality solutions are more likely to generate better offspring. However, this is not always the case, particularly in rugged problem landscapes with many local optima, where solutions can easily get trapped in MOEAs. Repetitively exploring such local-optimal solutions may not help. Indeed, recent studies show that mainstream MOEAs (e.g., NSGA-II and SMS-EMOA) can easily stagnate, and even more, their population may end up in a very different area at a time [Li *et al.*, 2023].

Very recently, Bian *et al.* [2025] analytically showed that introducing randomness in the population update process of MOEAs (called stochastic population update, SPU) can help the search. Specifically, the study proved that for SMS-EMOA solving the common benchmark problem OneJumpZeroJump, when $k = n/2 - \Omega(n)$, using SPU can bring an acceleration of $\Omega(2^{k/2}/(\sqrt{k}\mu^2)) = \Omega(2^{k/2}/(\sqrt{k}(n-2k+4)^2))$ on the expected running time, where n denotes the problem size, k ($2 \le k < n/2$) denotes the parameter of OneJumpZeroJump, and μ denotes the population size. Subsequently, Zheng and Doerr [2024b] extended SMS-EMOA to solve a many-objective problem, mOneJumpZeroJump, showing that the same SPU can bring an acceleration of $\Theta(2^k/\mu) = \Theta(2^k/(2n/m - 2k + 3)^{m/2})$ as well, where m is the number of objectives. These works echoed the empirical studies that show the benefit of considering non-elitism in MOEAs [Tanabe and Ishibuchi, 2019; Liang *et al.*, 2023a].

	Stochastic Population Update	**Stochastic Population Update + Archive**
SMS-EMOA	$O(\mu n^k \cdot \min\{1, \sqrt{k}\mu/2^{k/2}\})$ [Bian *et al.*, 2025] $O(\mu n^k \cdot \min\{1, \mu/2^k\})$ [Zheng and Doerr, 2024b] $[\mu \geq 2(n-2k+4); p_s = 1/2]$	$O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$ [Theorem 3] $[\mu \geq 3; C = e\mu/(p_s(1-p_c))]$
	$O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$ [Theorem 1] $[\mu \geq (n-2k+4)/(1-p_s); C = e\mu/(p_s(1-p_c))]$	
NSGA-II	$O(\mu \sqrt{k}(n/2)^k)$ [Bian *et al.*, 2025] $[\mu \geq 8(n-2k+3); p_s = 1/4; k > 8e^2]$	$O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$ [Theorem 5] $[\mu \geq 5; C = e/(p_s(1-p_c))]$
	$O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$ [Theorem 4] $[\mu \geq 4(n-2k+3)/(1-2p_s); C = e/(p_s(1-p_c))]$	

Table 1: The expected number of fitness evaluations of SMS-EMOA and NSGA-II for solving OneJumpZeroJump when using SPU alone, or with an archive, where n denotes the problem size, k ($2 \leq k < n/2$) denotes the parameter of OneJumpZeroJump, μ denotes the population size, p_c denotes the probability of using crossover, and p_s denotes the proportion of the current population and offspring solution(s) that SPU selects to preserve directly. The required ranges of p_c are: $1 - p_c = \Omega(1)$ for Theorems 1 and 4; $p_c = \Theta(1)$ for Theorems 3 and 5. The required ranges of p_s are: $p_s \in [1/(\mu+1), 1-1/o(\mu))$ for Theorem 1; $p_s \in [1/(\mu+1), (\mu-2)/(\mu+1)]$ for Theorem 3; $p_s \in [1/(2\mu), 1/2 - o(1/\mu))$ for Theorem 4; $p_s \in [1/(2\mu), (\mu-4)/(2\mu)]$ for Theorem 5.

SPU in [Bian *et al.*, 2025; Zheng and Doerr, 2024b] introduces randomness by randomly selecting a proportion p_s of the combined set of the current population and offspring solution(s) to be directly preserved into the next generation. This essentially gives a chance for inferior solutions to survive, which enables the evolutionary search to go along inferior regions which may be close to Pareto optimal regions. However, a cost of this method is that there is less space for the best solutions in the population. Some very best solutions to the problem (e.g., globally non-dominated solutions), even found by an MOEA, may be discarded during the population update process. This necessitates a large population used. Unfortunately, when the population size is large, the benefit of using SPU may vanish. This is because the benefit of SPU comes from the operation on inferior solutions while the large population size will lead to a small probability of selecting these solutions. For example, for SMS-EMOA solving OneJumpZeroJump, the acceleration of $\Omega(2^{k/2}/(\sqrt{k}\mu^2))$ brought by SPU [Bian *et al.*, 2025] will vanish when the population size μ is exponential w.r.t. k, e.g., $k = \log n$; for SMS-EMOA solving mOneJumpZeroJump, the acceleration of $\Theta(2^k/(2n/m - 2k + 3)^{m/2})$ [Zheng and Doerr, 2024b] will vanish when the number m of objectives is large, e.g., $m \geq k$, because the population size $\mu = (2n/m - 2k + 3)^{m/2}$ increases rapidly with m. This dilemma impacts both the effectiveness and practicality of using SPU in MOEAs.

Intuitively, a possible solution to this issue is to use an archive to store the best solutions ever found. In fact, in the area of MOEAs, this approach has become a popular practice [Li *et al.*, 2024]. Since the formalization of the archiving problem in the early 2000s [Knowles and Corne, 2003], there has been increasing interest and feasibility to use (even unbounded) archives in MOEAs, as seen in e.g. [Fieldsend *et al.*, 2003; Krause *et al.*, 2016; Brockhoff and Tušar, 2019; Ishibuchi *et al.*, 2020]. In this paper, we analytically show that incorporating an unbounded archive into SPU can reduce the population size and significantly enhance acceleration. Specifically, we compare the expected running time of two well-established MOEAs, SMS-EMOA and NSGA-II, with SPU for solving OneJumpZeroJump, when an archive is used or not. The results are shown in Table 1. Our contributions can be summarized as follows.

- We theoretically show that incorporating an archive mechanism with SPU can reduce the upper bound on the expected running time (even exponentially). For example, comparing Theorems 1 and 3 in Table 1, the expected running time of SMS-EMOA with SPU for solving OneJumpZeroJump, no matter whether an archive is used or not, is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e\mu/(p_s(1-p_c))$. The key difference is that using an archive allows a constant population size, resulting in a significantly smaller C and thus reducing the upper bound significantly. Note that Bian *et al.* [2024] recently proved the effectiveness of using an archive (bringing polynomial acceleration) for MOEAs, while our analysis for MOEAs with SPU reveals that even exponential acceleration can be obtained.
- Comparing the results of Theorems 1 and 4 in Table 1, we can find that the upper bound of NSGA-II is smaller than SMS-EMOA when using SPU. Our analysis reveals that the benefit of NSGA-II is due to its $(\mu + \mu)$ update mode, which selects each solution in the current population for reproduction and thus makes exploring promising dominated solutions easier.
- In addition, our derived running time bounds for MOEAs with SPU in the second column of Table 1 are significantly better than the previously known ones [Bian *et al.*, 2025; Zheng and Doerr, 2024b]. This improvement stems from the analysis method of considering both the number and size of jumps across gap of dominated solutions. The method happens to share similarities with that of proving Lemma 12 in [Doerr and Lutzeyer, 2024]. Moreover, our running time bounds are more general, as we consider variable survival probability p_s and crossover probability p_c.

We also validate our theoretical findings through an empirical study on the artificial OneJumpZeroJump problem and the multi-objective travelling salesperson problem (MOTSP) [Ribeiro *et al.*, 2002]. The results show that combining SPU with an archive leads to the best performance for both SMS-EMOA and NSGA-II. Furthermore, the results on the MOTSP show that with the SPU method, NSGA-II always performs better than SMS-EMOA. These results confirm our theoretical findings.

Finally, we give a brief overview about the running time analysis of MOEAs. Over the last decade, there has been an increasing interest for the evolutionary theory community to study MOEAs. Early theoretical works [Laumanns *et al.*, 2004a; Laumanns *et al.*, 2004b; Neumann, 2007; Giel and Lehre, 2010; Neumann and Theile, 2010; Doerr *et al.*, 2013; Qian *et al.*, 2013; Qian *et al.*, 2016; Bian *et al.*, 2018] mainly focus on analyzing the expected running time of a simple MOEA like GSEMO/SEMO. Recently, researchers have begun to examine practical MOEAs. Huang *et al.* [2021] investigated MOEA/D, assessing the effectiveness of different decomposition methods. Zheng *et al.* [2022] conducted the first theoretical analysis of NSGA-II. Bian *et al.* [2025] analyzed the running time of SMS-EMOA and showed that SPU can bring acceleration. Moreover, Wietheger and Doerr [2023] demonstrated that NSGA-III [Deb and Jain, 2014] exhibits superior performance over NSGA-II in solving the tri-objective problem 3OneMinMax. Ren *et al.* [2024a] analyzed the running time of SPEA2 on three m-objective problems. Some other works on well-established MOEAs include [Bian and Qian, 2022; Zheng and Doerr, 2022; Zheng and Doerr, 2024a; Cerf *et al.*, 2023; Dang *et al.*, 2023a; Dang *et al.*, 2023b; Doerr and Qu, 2023b; Doerr and Qu, 2023c; Doerr *et al.*, 2024; Doerr *et al.*, 2025; Lu *et al.*, 2024; Opris *et al.*, 2024; Opris, 2025; Ren *et al.*, 2024b].

2 Multi-objective Optimization

Multi-objective optimization aims to optimize two or more objective functions simultaneously, as shown in Definition 1. In this paper, we focus on maximization, while minimization can be defined similarly. The objectives are typically conflicting, meaning that there is no canonical complete order in the solution space $\mathcal{X}$. To compare solutions, we use the *domination* relationship in Definition 2. A solution is *Pareto optimal* if no other solution in $\mathcal{X}$ dominates it. The set of objective vectors corresponding to all 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 $f_1, f_2, \dots, f_m$, 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. *Let $\boldsymbol{f} = (f_1, f_2, \dots, f_m) : \mathcal{X} \to \mathbb{R}^m$ 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 $1 \le i \le m, f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{y})$;*
- *$\boldsymbol{x}$ 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 $\boldsymbol{y} \succeq \boldsymbol{x}$.*

Note that the notions of "weakly dominate" and "dominate" are also called "dominate" and "strictly dominate" in some works [Cerf *et al.*, 2023; Wietheger and Doerr, 2023].

Next, we introduce the benchmark problem OneJumpZeroJump studied in this paper. The OneJumpZeroJump problem as presented in Definition 3 is constructed based on the Jump problem [Doerr and Neumann, 2020], and has been widely used in MOEAs' theoretical analyses [Doerr and Zheng, 2021; Doerr and Qu, 2023a; Lu *et al.*, 2024; Ren *et al.*, 2024b]. Its first objective is the same as the Jump problem, which aims 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. The second objective is isomorphic to the first, with the roles of 1-bits and 0-bits reversed. 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. We use $F_I^* = \{(a, n+2k-a) \mid a \in [2k..n]\}$ to denote the inner part of the Pareto front.

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

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

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

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

3 Running Time Analysis of SMS-EMOA

In this section, we prove that using an archive can bring (even exponential) speedup for the well-established MOEA, SMS-EMOA [Beume *et al.*, 2007], with stochastic population update (SPU) solving the OneJumpZeroJump problem. In Section 3.1, we first introduce SMS-EMOA with SPU, and its expected running time for solving OneJumpZeroJump, which is much tighter than the previous results [Bian *et al.*, 2025; Zheng and Doerr, 2024b]. Then, in Section 3.2, we prove that using an archive can significantly reduce the upper bound on the expected running time. Due to space limitation, all mathematical proofs could only be sketched or had to be omitted completely. The full proof is given in the supplementary.

3.1 SMS-EMOA with SPU

SMS-EMOA presented in Algorithm 1 is a popular MOEA, which employs non-dominated sorting and hypervolume indicator to update the population. It starts from an initial population of μ solutions (line 1). In each generation, it randomly selects a solution $\boldsymbol{x}$ from the current population (line 3) for reproduction. With probability p_c, it selects another solution $\boldsymbol{y}$ and applies one-point crossover on $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate an offspring solution $\boldsymbol{x}'$ (lines 4–7); otherwise, $\boldsymbol{x}'$ is set as the copy of $\boldsymbol{x}$ (line 9). Note that one-point crossover selects a random point $i \in \{1, 2, \dots, n\}$ and exchanges the first i bits of

Algorithm 1 SMS-EMOA

Input: objective function $f_1, f_2 \cdots, f_m$, 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: select a solution $\boldsymbol{x}$ from P uniformly at random;
4: sample u from the uniform distribution over $[0,1]$;
5: **if** $u < p_c$ **then**
6: select a solution $\boldsymbol{y}$ from P uniformly at random;
7: apply one-point crossover on $\boldsymbol{x}$ and $\boldsymbol{y}$ to generate $\boldsymbol{x}'$
8: **else**
9: set $\boldsymbol{x}'$ as the copy of $\boldsymbol{x}$
10: **end if**
11: apply bit-wise mutation on $\boldsymbol{x}'$ to generate $\boldsymbol{x}''$;
12: $P \leftarrow$ POPULATION UPDATE$(P \cup \{\boldsymbol{x}''\})$
13: **end while**
14: **return** P

two parent solutions, which actually produces two new solutions, but the algorithm only picks the one that consists of the first part of the first parent solution and the second part of the second parent solution. Afterwards, bit-wise mutation flips each bit of $\boldsymbol{x}'$ with probability $1/n$ to produce an offspring $\boldsymbol{x}''$ (line 11). Then, the worst solution in $P \cup \{\boldsymbol{x}''\}$, the union of the current population and offspring, is removed (line 12) using the POPULATION UPDATE OF SMS-EMOA subroutine described in Algorithm 2. The subroutine first partitions the solution set Q (where $Q = P \cup \{\boldsymbol{x}''\}$) into non-dominated sets $R_1, R_2, \ldots, R_v$, where R_1 contains all non-dominated solutions in Q, and R_i ($i \geq 2$) contains all non-dominated solutions in $Q \setminus \cup_{j=1}^{i-1} R_j$. A solution $\boldsymbol{z} \in R_v$ is then removed by minimizing $\Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v) := HV_{\boldsymbol{r}}(R_v) - HV_{\boldsymbol{r}}(R_v \setminus \{\boldsymbol{x}\})$, where $HV_{\boldsymbol{r}}(X)$ denotes the hypervolume of the solution set X with respect to a reference point $\boldsymbol{r} \in \mathbb{R}^m$ ($\forall i, r_i \leq \min_{\boldsymbol{x} \in \mathcal{X}} f_i(\boldsymbol{x})$), i.e., the volume of the objective space between the reference point and the objective vectors of the solution set. A larger hypervolume indicates better approximation of the Pareto front in terms of convergence and diversity. For bi-objective problems, as defined in the original SMS-EMOA [Beume *et al.*, 2007], the algorithm omits the reference point and directly preserves the two boundary points, allowing the hypervolume to be calculated accordingly.

In [Bian *et al.*, 2025], the SPU method is introduced and shown to be beneficial for the search of MOEAs. During population updates, SPU randomly selects a proportion p_s of the current population and the offspring solution(s) to directly survive into the next generation and the removed part is selected from the rest solutions. This implies that each solution, including the worst solution in the population, has at least a probability p_s of surviving to the next generation. Specifically, SPU OF SMS-EMOA as presented in Algorithm 3 is used to replace the original POPULATION UPDATE procedure in line 12 of Algorithm 1. Note that p_s is set to $1/2$ in [Bian *et al.*, 2025], while we consider a general p_s here.

The expected running time of SMS-EMOA with SPU for

Algorithm 2 POPULATION UPDATE OF SMS-EMOA (Q)

Input: a set Q of solutions, and a reference point $\boldsymbol{r} \in \mathbb{R}^m$
Output: $|Q|-1$ solutions from Q

partition Q into non-dominated sets $R_1, R_2, \ldots, R_v$;
let $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$;
return $Q \setminus \{\boldsymbol{z}\}$

Algorithm 3 SPU OF SMS-EMOA (Q)

Input: a set Q of solutions, and a reference point $\boldsymbol{r} \in \mathbb{R}^m$
Output: $|Q|-1$ solutions from Q

$Q' \leftarrow \lfloor |Q| \cdot (1-p_s) \rfloor$ solutions uniformly and randomly selected from Q without replacement;
partition Q' into non-dominated sets $R_1, R_2, \ldots, R_v$;
let $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$;
return $Q \setminus \{\boldsymbol{z}\}$

solving OneJumpZeroJump has been proven to be $O(\mu n^k \cdot \min\{1, \sqrt{k}\mu/2^{k/2}\})$ [Bian *et al.*, 2025], which is better than that, i.e., $O(\mu n^k)$, of the original SMS-EMOA. Intuitively, by by introducing randomness into the population update, the evolutionary search has a chance to go along inferior regions which are close to Pareto optimal regions, thereby making the search easier. Here, we re-prove a tighter upper bound on the expected running time of SMS-EMOA with SPU for solving OneJumpZeroJump, as shown in Theorem 1. It is also more general, as it considers a survival probability $p_s \in [1/(\mu+1), 1-1/o(\mu))$, rather than just $p_s = 1/2$ as in [Bian *et al.*, 2025]. Note that the running time of EAs is often measured by the number of fitness evaluations.

Theorem 1. *For SMS-EMOA solving OneJumpZeroJump with $n - 2k = \Omega(n)$, if using SPU with survival probability $p_s \in [1/(\mu+1), 1-1/o(\mu))$, the crossover probability $1 - p_c = \Omega(1)$, and the population size $\mu \geq (n-2k+4)/(1-p_s)$, then the expected running time for finding the whole Pareto front is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e\mu/(p_s(1-p_c))$.*

The proof of Theorem 1 needs Lemma 2, which shows that given a proper value of μ, an objective vector on the Pareto front will always be maintained once it has been found. The reason is that in Algorithm 3, the removed solution is selected from $\lfloor (\mu+1) \cdot (1-p_s) \rfloor \geq n-2k+4$ solutions in Q. For each objective vector on the Pareto front, whose size is $n - 2k + 3$, only one solution has positive Δ-value. This ensures that these solutions rank among the top $n - 2k + 3$ solutions, and thus will not be removed.

Lemma 2. *For SMS-EMOA solving OneJumpZeroJump, if using SPU with survival probability $p_s \in [1/(\mu+1), 1-1/o(\mu))$, and the population size $\mu \geq (n-2k+4)/(1-p_s)$, then an objective vector $\boldsymbol{f}^*$ on the Pareto front will always be maintained once it has been found.*

Proof Sketch of Theorem 1. We divide the optimization procedure into two phases: the first phase starts after initialization and finishes until the inner part F_I^* of the Pareto front is found; the second phase starts after the first phase and finishes until the extreme Pareto optimal solution 1^n is found.

Note that the analysis for finding 0^n holds similarly. Since Lemma 2 ensures the maintenance of objective vectors in F_I^*, making the first-phase analysis similar to that of Theorem 1 in [Bian *et al.*, 2025], we can derive that the expected running time of the first phase is $O(\mu n \log n)$.

Next, we consider the second phase. By employing SPU, any solution (including dominated ones) can survive into the next generation with probability at least p_s. This means that the population can preserve some dominated solutions to gradually reach 1^n. We assume a "jump" to be an event where a solution $\boldsymbol{x}$ with $|\boldsymbol{x}|_1 \in [n-k..n-1]$ is selected, and a new dominated solution closer to 1^n is generated and preserved. Thus, 1^n can be reached more easily through multiple jumps across the gap of dominated solution set (i.e., $\{\boldsymbol{x} \mid |\boldsymbol{x}|_1 \in [n-k+1..n-1]\}$). We refer to the dominated solutions along the multiple jumps as "stepping stones" and assume that 1^n can be reached through M stepping stones. Then, we consider $M+1$ consecutive jumps, which start from a solution $\boldsymbol{x}$ with $n-k$ 1-bits, and continue to generate the next stepping-stone solution until finding 1^n. Any failure during the intermediate jumps will result in restarting the process from the solution with $n-k$ 1-bits. By enumerating all possible jump sizes (the sum of which is required to be k), we prove that the $M+1$ consecutive jumps can happen with probability at least $(1-p_c)(M+1)^k/(e\mu n^k C^M)$, where $C = e\mu/(p_s(1-p_c))$. Thus, 1^n can be found after at most $e\mu n^k C^M/((1-p_c)(M+1)^k)$ trials in expectation. As each trial requires up to $M+1$ generations and $1-p_c = \Omega(1)$, the expected number of generations for finding 1^n is

$$(M+1) \cdot \frac{e\mu n^k \cdot C^M}{(1-p_c)(M+1)^k} = O\left(\frac{\mu n^k \cdot C^M}{(M+1)^{k-1}}\right). \quad (1)$$

Finally, we minimize this upper bound by taking $M = \lceil (k-1)/\ln C - 1 \rceil$ when $k > e \ln C$, and $M = 0$ when $k \le e \ln C$. This leads to that the expected number of generations of the second phase is at most $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$.

As SMS-EMOA generates one solution per generation, its running time is equal to the number of generations. Combining the two phases, the total expected running time is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $O(\mu n \log n)$ required by the first phase is dominated, and $C = e\mu/(p_s(1-p_c))$. □

Under the same conditions as the previous results $O(\mu n^k \cdot \min\{1, \sqrt{k}\mu/2^{k/2}\})$ in [Bian *et al.*, 2025] and $O(\mu n^k \cdot \min\{1, \mu/2^k\})$ in [Zheng and Doerr, 2024b], where SPU is used with $p_s = 1/2$ and crossover probability $p_c = 0$, our bound in Theorem 1 becomes $O(\mu n^k \cdot (e \ln(2e\mu)/k)^{k-1})$. Since the bound in [Zheng and Doerr, 2024b] is tighter than that in [Bian *et al.*, 2025], we focus on comparing our result only with [Zheng and Doerr, 2024b]. When $k > 2e \ln(2e\mu)$, our bound $O(\mu n^k \cdot (e \ln(2e\mu)/k)^{k-1})$ brings an improvement ratio of $\Theta(\mu(k/(2e \ln(2e\mu)))^{k-1})$, which can be exponential when k is large, e.g., $k = n/8$. For a very small range $\log \mu \le k < 2e \ln(2e\mu)$, our bound shows no advantage, which is because to minimize Eq. (1), we choose $M = \lceil (k-1)/\ln C - 1 \rceil$ to round up to the nearest integer, leading to an over-relaxation. If we instead set $M = 1$, the same bound can be obtained. The reason of our better bound is in the second phase of finding the Pareto optimal solution 1^n: 1) We consider $M+1$ "jumps" across the gap between dominated solutions, and account for all possible jump sizes to find 1^n; 2) We select the optimal number of jumps, i.e., $M = \lceil (k-1)/\ln C - 1 \rceil$. Note that Bian *et al.* [2025] considered only two fixed-size jumps, and although Zheng and Doerr [2024b] accounted for all possible jump sizes, they also reduced the process to two jumps.

3.2 An Archive is Provably Helpful

In the last section, we have proved that for SMS-EMOA with SPU solving OneJumpZeroJump, the expected running time is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e\mu/(p_s(1-p_c))$. From the analysis, we find that SPU benefits evolutionary search by exploring inferior regions that are close to Pareto optimal areas, but it also requires a larger population size $\mu \ge (n-2k+4)/(1-p_s)$ to preserve the Pareto optimal solutions discovered. The greater the randomness introduced by SPU (i.e., the larger the value of p_s), the larger the population size needed. Note that a large population size may diminish the benefit of SPU, because it will lead to a very small probability of selecting specific dominated solutions for reproduction, which is required by SPU. For example, when the population size μ is exponential w.r.t. k, e.g., $k = \log n$, the improvement by SPU will vanish, compared to the expected running time $O(\mu n^k)$ without SPU [Bian *et al.*, 2025].

In this section, we theoretically show that the limitation of SPU can be alleviated by using an archive. Once a new solution is generated, the solution will be tested if it can enter the archive. If there is no solution in the archive that dominates the new solution, then the solution will be placed in the archive, and meanwhile those solutions weakly dominated by the new solution will be deleted from the archive. Algorithmic steps incurred by adding an archive in SMS-EMOA are given as follows. In Algorithm 1, an empty set A is initialized in line 1, and the following lines are added after line 11:

if $\nexists \boldsymbol{z} \in A$ such that $\boldsymbol{z} \succ \boldsymbol{x}''$ **then**
  $A \leftarrow (A \setminus \{\boldsymbol{z} \in A \mid \boldsymbol{x}'' \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{x}''\}$
end if

The set A instead of P is returned in the last line.

We prove in Theorem 3 that if using an archive, a population size $\mu \ge 3$ is sufficient to guarantee the same running time bound of SMS-EMOA with SPU as Theorem 1. Note that Theorem 1 requires $\mu \ge (n-2k+4)/(1-p_s)$, which implies that using an archive can allow a small population size and thus bring speedup.

Theorem 3. *For SMS-EMOA solving OneJumpZeroJump with $n - 2k = \Omega(n)$, if using SPU with survival probability $p_s \in [1/(\mu+1), (\mu-2)/(\mu+1)]$, the crossover probability $p_c = \Theta(1)$, the population size $\mu \ge 3$, and using an archive, then the expected running time for finding the whole Pareto front is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e\mu/(p_s(1-p_c))$.*

Proof Sketch of Theorem 3. We divide the optimization procedure into three phases, where the first phase aims at finding the two boundary solutions in the inner part F_I^* of the Pareto front (i.e., a solution with k 1-bits and a solution with $n-k$ 1-bits), the second phase aims at finding the two extreme Pareto optimal solutions 1^n and 0^n, and the third phase aims at finding the remaining objective vectors on the Pareto front.

A key property used in the proof is that the maximum f_1 value among the Pareto optimal solutions in $P \cup \{\boldsymbol{x}''\}$ will not decrease, because the two boundary solutions in R_1 (i.e., the first non-dominated set by non-dominated ranking) are prioritized for preservation into the next generation. For the first phase, we show that the expected number of generations for finding a solution with $n-k$ 1-bits is $O(\mu n \log n)$, by continuously selecting the solution with the maximum f_1 value in the population and flipping a 0-bit to increase the f_1 value. The process for finding another boundary solution with k 1-bits is symmetric. For the second phase, the analysis follows that of the second phase in Theorem 1, leading to an expected running time of $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$ for finding 1^n and 0^n, where $C = e\mu/(p_s(1-p_c))$. For the third phase, since 0^n and 1^n must be maintained in the population P, we consider the case where 0^n or 1^n is selected as a parent, and another solution $\boldsymbol{x} \in P$ is chosen as the second parent. By using one-point crossover, new points on the Pareto front can be continuously generated and added to the archive. Consequently, the expected running time of the third phase is $O(\mu n \log n)$. Combining the three phases, the total expected running time is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $O(\mu n \log n)$ required by the first phase and third phase is dominated and thus omitted. □

For the case without an archive, in order to avoid losing the Pareto optimal solutions found while using SPU, the population size must be large, i.e., $\mu \geq (n-2k+4)/(1-p_s)$. However, for the case with an archive, the population size only needs to be a constant. The main reasons are: 1) Using an archive that stores all the Pareto optimal solutions generated enables the algorithm not to worry about losing Pareto optimal solutions, but only endeavoring to seek new Pareto optimal solutions. 2) In the context of our analysis, a constant population size is sufficient to preserve exploration-favoring solutions, i.e., the two boundary Pareto optimal solutions.

The smaller population size allows a larger probability of selecting inferior solutions that are close to Pareto optimal areas for reproduction, thus leading to speedup. When k is limited, using only SPU may not bring acceleration. For example, when $k = e\ln(8en)$, if using SPU with $p_s = 1/2$, $\mu = 2n$, and $p_c = 1/2$, the expected running time of SMS-EMOA on OneJumpZeroJump is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\}) = O(n^{k+1})$ (where $C = e\mu/(p_s(1-p_c)) = 8en$), implying no acceleration compared to $O(\mu n^k) = O(n^{k+1})$ without SPU [Bian *et al.*, 2025]. However, by adding an archive and reducing the population size μ to 5, while keeping other settings unchanged, the bound reduces to $O(n^k \cdot (e\ln(20e)/k)^{k-1})$, implying a superpolynomial reduction in the upper bound as $k = e\ln(8en)$. Note that we set $\mu = 5$ to satisfy the condition of Theorem 3, i.e., to make $p_s = 1/2 \in [1/(\mu+1), (\mu-2)/(\mu+1)]$. On the other hand, when k is large, using SPU alone can lead to exponential acceleration, and the addition of an archive can further enhance this acceleration. For example, when $k = n/8$, if using SPU with $p_s = 1/2$, $\mu = 2n$, and $p_c = 1/2$, the expected running time is $O(\mu n^k \cdot (8e\ln(8en)/n)^{n/8-1})$, which implies an exponential reduction in the upper bound compared to $O(\mu n^k)$ without SPU. By adding an archive and setting μ to 5, the bound reduces to $O(n^k \cdot (8e\ln(20e)/n)^{n/8-1})$, resulting in an improvement ratio of $\Theta(n(\ln(8en)/\ln(20e))^{n/8-1})$, i.e., an exponential reduction in the upper bound.

We also note that the recent work [Bian *et al.*, 2024] has proven that using an archive can provide polynomial acceleration for MOEAs for the first time. For example, for SMS-EMOA solving OneMinMax, the expected running time is $O(\mu n \log n)$ both with and without an archive, but the archive allows for a constant μ, achieving an acceleration of $\Theta(n)$. Our work gives another theoretical evidence for the effectiveness of using an archive, and further shows that superpolynomial or even exponential acceleration can be achieved. In our analysis with SPU, a sequence of $M+1$ consecutive jumps are required, which implies continuously selecting specific solutions for reproduction; using an archive allows a small population size, which increases the selection probability significantly and thus leads to greater acceleration.

4 Running Time Analysis of NSGA-II

In this section, we show that an archive is also helpful for NSGA-II solving OneJumpZeroJump if using SPU. Specifically, we prove in Theorem 4 that the expected running time of NSGA-II using SPU for solving OneJumpZeroJump is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e/(p_s(1-p_c))$, and the population size μ is required to be at least $4(n-2k+3)/(1-2p_s)$. Theorem 5 shows that using an archive with a constant population size, the same running time bound holds. Their proofs are similar to that of Theorems 1 and 3, respectively. The main difference is that in each generation, the probability of selecting a specific parent solution for reproduction is changed from $1/\mu$ to 1 due to the $(\mu+\mu)$ mode and fair selection employed by NSGA-II, making the value of C independent of μ. The introduction of NSGA-II, the corresponding SPU and archive mechanisms, and the proof details are provided in the supplementary due to space limitation.

Theorem 4. *For NSGA-II solving OneJumpZeroJump with $n-2k = \Omega(n)$, if using SPU with survival probability $p_s \in [1/(2\mu), 1/2 - o(1/\mu))$, the crossover probability $1-p_c = \Omega(1)$, and the population size $\mu \geq 4(n-2k+3)/(1-2p_s)$, then the expected running time for finding the whole Pareto front is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e/(p_s(1-p_c))$.*

Bian *et al.* [2025] analyzed NSGA-II solving OneJumpZeroJump with $k > 8e^2$, $p_s = 1/4$, and $p_c = 0$, deriving an expected running time of $O(\mu\sqrt{k}(n/2)^k)$. In contrast, Theorem 4 gives a tighter bound $O(\mu n^k \cdot (e\ln(4e)/k)^{k-1})$, with an improvement ratio of $\Theta(\sqrt{k}(k/(2e\ln(4e)))^{k-1})$, which can be exponential when k is large, e.g., $k = n/8$. Our analysis leads to improvement because, in the second phase of finding the extreme Pareto optimal solution 1^n, we consider an optimal number $M+1$ of jumps across the gap of the dominated solution set (i.e., $\{\boldsymbol{x} \mid |\boldsymbol{x}|_1 \in [n-k+1..n-1]\}$), rather than a continuous sequence of k jumps from $|\boldsymbol{x}|_1 = n-k$ to n in [Bian *et al.*, 2025].

Algorithm	Original Algorithm $\mu = 2\|\mathcal{PF}\|$	Only SPU $\mu = 2\|\mathcal{PF}\|$	Only Archive $\mu = \frac{1}{4}\|\mathcal{PF}\|$	SPU+Archive $\mu = \frac{1}{4}\|\mathcal{PF}\|$
SMS-EMOA	4.0923e+4 (5.69e+3) †	3.8018e+4 (6.27e+3) †	2.1939e+4 (2.03e+3) †	**1.2201e+4 (1.66e+3)**
NSGA-II	4.0017e+4 (2.55e+3) †	3.2850e+4 (3.03e+3) †	1.9363e+4 (2.21e+3) †	**9.8081e+3 (1.49e+3)**

"†" indicates that the result is significantly different from that of the SPU + archive algorithm (last column), at a 95% confidence by the Wilcoxon rank-sum test.

Table 2: The IGD [Coello Coello and Reyes Sierra, 2004] results (mean and standard deviation) of the four variants of NSGA-II and SMS-EMOA on the 100-cities MOTSP instance clusAB from TSPLIB [Reinelt, 1991]. For each MOEA, the best mean is highlighted in bold.

Theorem 5. *For NSGA-II solving OneJumpZeroJump with $n - 2k = \Omega(n)$, if using SPU with survival probability $p_s \in [1/(2\mu), (\mu-4)/(2\mu)]$, the crossover probability $p_c = \Theta(1)$, the population size $\mu \geq 5$, and using an archive, then the expected running time for finding the whole Pareto front is $O(\mu n^k \cdot \min\{1, (e \ln C/k)^{k-1}\})$, where $C = e/(p_s(1-p_c))$.*

Comparing Theorems 4 and 5, we find that if using SPU, the expected running time for NSGA-II solving OneJumpZeroJump with or without an archive is both $O(\mu n^k \cdot (e \ln C/k)^{k-1})$, where $C = e/(p_s(1-p_c))$. The difference is that using an archive allows for a small constant population size μ, which can bring an acceleration factor of $\Theta(n/(1-2p_s))$, as μ is required to be at least $4(n-2k+3)/(1-2p_s)$ if not using an archive. Compared to SMS-EMOA, using an archive does not lead to an exponential reduction in the upper bound for NSGA-II. The reason is that NSGA-II employs the $(\mu+\mu)$ update mode and fair selection, allowing it to select and explore all solutions in each generation, thereby alleviating selection pressure. This results in a smaller constant $C = e/(p_s(1-p_c))$ independent of μ in Theorems 4 and 5, compared to $C = e\mu/(p_s(1-p_c))$ in Theorems 1 and 3. This also implies that NSGA-II achieves a smaller expected running time than SMS-EMOA, suggesting that the $(\mu+\mu)$ mode may be more suitable for SPU than the $(\mu+1)$ mode.

5 Experiments

In this section, we conduct experiments on OneJumpZeroJump and the well-known multi-objective travelling salesman problem (MOTSP) [Ribeiro *et al.*, 2002]. Table 3 presents the results of NSGA-II solving the OneJumpZeroJump problem with size $n \in \{10, 15, 20, 25, 30\}$ and $k = 3$. The settings are: population size $\mu = 8$ with an archive, and $\mu = 8(n-2k+3)$ without; survival probability $p_s = 1/2$ when using SPU; and crossover probability $p_c = 1/2$. We present the average number of fitness evaluations over 200 runs for three configurations: SPU only, archive only, and SPU + archive. Results show that using SPU+archive significantly reduces running time compared to using only SPU, and using only archive. We have similar results on SMS-EMOA in the supplementary.

For the MOTSP, the Pareto front size $|\mathcal{PF}|$ is known, allowing us to set commensurable population sizes μ for the compared MOEAs. We conduct experiments on two 100-cities instances (i.e., clusAB and kroAB) of MOTSP [1] using NSGA-II and SMS-EMOA under four scenarios with varying population sizes: 1) the original algorithms, 2) using only SPU, 3) using only an archive, and 4) using SPU and an archive. For a fair comparison, each scenario used $1,000,000$ fitness evaluations over 30 runs. Table 2 presents the results of a widely-used quality indicator, Inverted Generational Distance (IGD) [Coello Coello and Reyes Sierra, 2004] on the instance clusAB. We used IGD since it can measure how well the obtained solution set represents the Pareto front [Li *et al.*, 2020]. Table 2 shows that using SPU and an archive simultaneously can lead to the smallest value of IGD for both SMS-EMOA and NSGA-II, and statistically outperforms the other three scenarios. Moreover, with the SPU method, NSGA-II always achieves smaller IGD values than SMS-EMOA. Similar results can be observed on the kroAB instance, along with additional experiments using smaller population sizes ($\mu = |\mathcal{PF}|/6$, $|\mathcal{PF}|/8$, and $|\mathcal{PF}|/10$). These findings further validate our theoretical analysis, with full experimental details available in the supplementary.

[1] https://webia.lip6.fr/~lustt/Research.html

size n	SPU	Archive	SPU+Archive
10	5212.48	11733.52	5125.52
15	35653.92	57941.48	25808.24
20	106424.76	183201.24	65398.97
25	239294.88	415550.16	146384.08
30	479311.56	767354.84	284114.12

Table 3: Average number of fitness evaluations over 200 independent runs for NSGA-II solving OneJumpZeroJump with $k = 3$.

6 Conclusion

This paper analytically shows that SPU in MOEAs needs an archive to better leverage its exploration ability. We prove that for NSGA-II and SMS-EMOA solving OneJumpZeroJump, introducing an archive for SPU can address the dilemma of large population size and may provide an extra exponential speedup. The reason is that SPU requires a large population to preserve the best solutions found, while an archive enables a small population size, increasing the chance of selecting inferior but undeveloped solutions. We also find that the $(\mu+\mu)$ update mode may be more suitable for SPU than the $(\mu+1)$ update mode. Another contribution lies in improving the running time bounds for SMS-EMOA and NSGA-II solving OneJumpZeroJump using SPU. These theoretical findings are empirically validated on OneJumpZeroJump and the MOTSP. We hope our work may provide some theoretical evidence for the attempts of designing new MOEAs that separate the exploration (via the evolutionary population) and elitist solution preservation (via an external archive), such as in non-elitist or less elitist MOEAs [Tanabe and Ishibuchi, 2019; Liang *et al.*, 2023b].

Acknowledgements

This work was supported by the National Science and Technology Major Project (2022ZD0116600), the National Science Foundation of China (62276124), and the Fundamental Research Funds for the Central Universities (14380020). Chao Qian is the corresponding author. The supplementary is available at arXiv.

References

[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 *PPSN*, 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 *IJCAI*, pages 1405–1411, Stockholm, Sweden, 2018.

[Bian *et al.*, 2024] C. Bian, S. Ren, M. Li, and C. Qian. An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In *IJCAI*, pages 6905–6913, Jeju Island, South Korea, 2024.

[Bian *et al.*, 2025] C. Bian, Y. Zhou, M. Li, and C. Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. *AIJ*, 341:104308, 2025.

[Brockhoff and Tušar, 2019] D. Brockhoff and T. Tušar. Benchmarking algorithms from the platypus framework on the biobjective bbob-biobj testbed. In *GECCO*, pages 1905–1911, Prague, Czech Republic, 2019.

[Cerf *et al.*, 2023] S. Cerf, B. Doerr, B. Hebras, Y. Kahane, and S. Wietheger. The first proven performance guarantees for the non-dominated sorting genetic algorithm II (NSGA-II) on a combinatorial optimization problem. In *IJCAI*, pages 5522–5530, Macao, SAR, China, 2023.

[Coello Coello and Reyes Sierra, 2004] C. A. Coello Coello and M. Reyes Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *MICAI*, pages 688–697, Mexico City, Mexico, 2004.

[Dang *et al.*, 2023a] D.-C. Dang, A. Opris, B. Salehi, and D. Sudholt. Analysing the robustness of NSGA-II under noise. In *GECCO*, pages 642–651, Lisbon, Portugal, 2023.

[Dang *et al.*, 2023b] 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. In *AAAI*, pages 12390–12398, Washington, DC, 2023.

[Deb and Jain, 2014] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[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, 2001] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[Doerr and Lutzeyer, 2024] B. Doerr and J. F. Lutzeyer. Hyper-heuristics can profit from global variation operators. *CORR abs/2407.14237*, 2024.

[Doerr and Neumann, 2020] B. Doerr and F. Neumann. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Springer, 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 *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 *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 *AAAI*, 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 *CEC*, pages 432–439, Cancun, Mexico, 2013.

[Doerr *et al.*, 2024] B. Doerr, T. Ivan, and M. S. Krejca. Speeding up the NSGA-II with a simple tie-breaking rule. *CORR abs/11931*, 2024.

[Doerr *et al.*, 2025] B. Doerr, M. S. Krejca, and G. Rudolph. Runtime analysis for multi-objective evolutionary algorithms in unbounded integer spaces. In *AAAI*, pages 26955–26963, Philadelphia, PA, 2025.

[Fieldsend *et al.*, 2003] J. E. Fieldsend, R. M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, 2003.

[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.

[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 *IJCAI*, pages 1682–1688, Virtual, 2021.

[Ishibuchi *et al.*, 2020] H. Ishibuchi, L. M. Pang, and K. Shang. A new framework of evolutionary multi-objective algorithms with an unbounded external archive. In *ECAI 2020*, pages 283–290, Santiago, Spain, 2020.

[Knowles and Corne, 2003] J. Knowles and D. Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[Krause *et al.*, 2016] O. Krause, T. Glasmachers, N. Hansen, and C. Igel. Unbounded population MO-CMA-ES for the bi-objective bbob test suite. In *GECCO*, pages 1177–1184, Denver, CO, 2016.

[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.

[Li *et al.*, 2020] M. Li, T. Chen, and X. Yao. How to evaluate solutions in pareto-based search-based software engineering: A critical review and methodological guidance. *IEEE Transactions on Software Engineering*, 48(5):1771–1799, 2020.

[Li *et al.*, 2023] M. Li, X. Han, and X. Chu. MOEAs are stuck in a different area at a time. In *GECCO*, pages 303–311, Lisbon, Portugal, 2023.

[Li *et al.*, 2024] M. Li, M. López-Ibáñez, and X. Yao. Multi-objective archiving. *IEEE Transactions on Evolutionary Computation*, 28(3):696–717, 2024.

[Liang *et al.*, 2023a] Z. Liang, M. Li, and P. K. Lehre. Non-elitist evolutionary multi-objective optimisation: Proof-of-principle results. In *GECCO*, pages 383–386, Lisbon, Portugal, 2023.

[Liang *et al.*, 2023b] Z. Liang, M. Li, and P. K. Lehre. Non-elitist evolutionary multi-objective optimisation: Proof-of-principle results. In *GECCO*, pages 383–386, Lisbon, Portugal, 2023.

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

[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 *PPSN*, 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.

[Opris *et al.*, 2024] A. Opris, D.-C. Dang, and D. Sudholt. Runtime analyses of NSGA-III on many-objective problems. In *GECCO*, pages 1596–1604, Melbourne, Australia, 2024.

[Opris, 2025] A. Opris. A many-objective problem where crossover is provably indispensable. In *AAAI*, pages 27108–27116, Philadelphia, PA, 2025.

[Qian *et al.*, 2013] C. Qian, Y. Yu, and Z.-H. Zhou. An analysis on recombination in multi-objective evolutionary optimization. *AIJ*, 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 *PPSN*, pages 835–846, Edinburgh, Scotland, 2016.

[Reinelt, 1991] G. Reinelt. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4), 1991.

[Ren *et al.*, 2024a] S. Ren, C. Bian, M. Li, and C. Qian. A first running time analysis of the strength Pareto evolutionary algorithm 2 (SPEA2). In *PPSN*, pages 295–312, Hagenberg, Austria, 2024.

[Ren *et al.*, 2024b] S. Ren, Z. Qiu, C. Bian, M. Li, and C. Qian. Maintaining diversity provably helps in evolutionary multimodal optimization. In *IJCAI*, pages 7012–7020, Jeju Island, South Korea, 2024.

[Ribeiro *et al.*, 2002] C. C. Ribeiro, P. Hansen, P. C. Borges, and M. P. Hansen. A study of global convexity for a multiple objective travelling salesman problem. *Essays and Surveys in Metaheuristics*, pages 129–150, 2002.

[Tanabe and Ishibuchi, 2019] R. Tanabe and H. Ishibuchi. Non-elitist evolutionary multi-objective optimizers revisited. In *GECCO*, pages 612–619, Prague, Czech Republic, 2019.

[Wietheger and Doerr, 2023] S. Wietheger and B. Doerr. A mathematical runtime analysis of the non-dominated sorting genetic algorithm III (NSGA-III). In *IJCAI*, pages 5657–5665, Macao, SAR, China, 2023.

[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 *GECCO*, pages 611–619, Boston, MA, 2022.

[Zheng and Doerr, 2024a] W. Zheng and B. Doerr. Runtime analysis for the NSGA-II: Proving, quantifying, and explaining the inefficiency for many objectives. *IEEE Transactions on Evolutionary Computation*, 28(5):1442–1454, 2024.

[Zheng and Doerr, 2024b] W. Zheng and B. Doerr. Runtime analysis of the SMS-EMOA for many-objective optimization. In *AAAI*, pages 20874–20882, Vancouver, Canada, 2024.

[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 *AAAI*, 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, 2019.