An Archive Can Bring Provable Speed-ups in Multi-Objective Evolutionary Algorithms

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

Abstract

In the area of multi-objective evolutionary algorithms (MOEAs), there is a trend of using an archive to store non-dominated solutions generated during the search. This is because 1) MOEAs may easily end up with the final population containing inferior solutions that are dominated by other solutions discarded during the search process and 2) the population that has a commensurable size of the problem's Pareto front is often not practical. In this paper, we theoretically show, for the first time, that using an archive can guarantee speed-ups for MOEAs. Specifically, we prove that for two well-established MOEAs (NSGA-II and SMS-EMOA) on two commonly studied problems (OneMinMax and LeadingOnesTrailingZeroes), using an archive brings a polynomial acceleration on the expected running time. The reason is that with an archive, the size of the population can reduce to a small constant; there is no need for the population to keep all the Pareto optimal solutions found. This contrasts existing theoretical studies for MOEAs where a population with a commensurable size of the problem's Pareto front is needed. The findings in this paper not only provide a theoretical confirmation for an increasingly popular practice in the design of MOEAs, but can also be beneficial to the theory community towards studying more practical MOEAs.

1 Introduction

Multi-objective optimization refers to an optimization scenario of considering multiple objectives simultaneously. Since the objectives of a multi-objective optimization problem (MOP) are usually conflicting, there does not exist a single optimal solution, but a set of trade-off solutions, called Pareto optimal solution set (or Pareto front in the objective space). Evolutionary algorithms (EAs), a kind of randomized heuristic optimization algorithms inspired by natural evolution, have been found well-suited to MOPs. Their population-based search mechanism can approximate a set of Pareto optimal solutions within one execution, with one solution representing a different trade-off between the objectives. Over the last decades, there have been a lot of well-known multi-objective EAs (MOEAs) developed, such as 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 the MOEA area, recently there is a trend of using an archive to store non-dominated solutions generated during the search [Li *et al.*, 2023b]. A major reason for this practice is that the population in MOEAs may fail to preserve high-quality solutions found even with elite preservation [Knowles and Corne, 2004]. During the evolutionary process, it is highly likely that there are more non-dominated solutions generated than the capacity of the population. As such, a population truncation needs to take place to remove excess non-dominated solutions (e.g., on the basis of their crowdedness in the population). However, the population later may accept new solutions which are non-dominated to the current population but being dominated by the ones removed previously (due to the dynamics of the evolution, for example, some area becomes sparse again). Consequently, the final population may have a large portion of dominated solutions. This applies to all practical MOEAs [Li *et al.*, 2023b]. For example, it has been reported in [Li and Yao, 2019] that on some problems (such as DTLZ7 [Deb *et al.*, 2005]), nearly half of the final population of NSGA-II and MOEA/D are not globally optimal (i.e., being dominated by other solutions which were discarded during their evolutionary process). This unwelcome issue has been frequently observed in various scenarios, from synthetic test suites [Laumanns *et al.*, 2002; Fieldsend *et al.*, 2003; Li and Yao, 2019] to real-world problems [Fieldsend, 2017; Chen *et al.*, 2019].

An easy way to fix the above issue is to use an (unbounded) archive that stores all non-dominated solutions found, leaving the search focused on finding new non-dominated solutions. This is indeed an increasingly popular practice [Li *et al.*, 2023b], given the capacity of today's computers that storing millions of solutions does not pose a problem. Various studies of using an unbounded archive emerged [Li *et al.*, 2023b], including to store high-quality solutions [Dubois-Lacoste *et al.*, 2015; Ishibuchi *et al.*, 2020; Zhang *et al.*, 2023], to incorporate it into MOEAs as an important al-

	OneMinMax	LeadingOnesTrailingZeroes
Original NSGA-II, namely, the one without using an archive ([Bian and Qian, 2022])	$O(\mu n \log n)$ [$\mu \geq 2(n+1)$]	$O(\mu n^2)$ [$\mu \geq 2(n+1)$]
NSGA-II using an archive (this paper)	$O(\mu n \log n)$ [Thm 1, $\mu \geq 4$]	$O(\mu n^2 + \mu^2 n \log n)$ [Thm 2, $\mu \geq 4$]
Original SMS-EMOA, namely, the one without using an archive ([Zheng and Doerr, 2024])	$O(\mu n \log n)$ [$\mu \geq n+1$]	$O(\mu n^2)$ [$\mu \geq n+1$]
SMS-EMOA using an archive (this paper)	$O(\mu n \log n)$ [Thm 3, $\mu \geq 2$]	$O(\mu n^2 + \mu^2 n \log n)$ [Thm 4, $\mu \geq 2$]

Table 1: The expected number of fitness evaluations of NSGA-II and SMS-EMOA for solving the OneMinMax and LeadingOnesTrailingZeroes problems when an archive is used or not, where n denotes the problem size, and μ denotes the population size.

gorithm component [Wang *et al.*, 2019; Li *et al.*, 2021], to benchmark different MOEAs [Tanabe and Oyama, 2017; Brockhoff and Tušar, 2019], to use it to identify if the search stagnates [Li *et al.*, 2023a], and to design efficient data structure for it [Glasmachers, 2017; Jaszkiewicz and Lust, 2018; Fieldsend, 2020; Nan *et al.*, 2020]. An archive (even a bounded one) can significantly improve the performance of MOEAs, as shown empirically in [Bezerra *et al.*, 2019].

In this paper, we theoretically show that using an archive can bring speed-ups for MOEAs. Specifically, we study the expected running time of two well-established MOEAs, NSGA-II and SMS-EMOA, when using an archive to store non-dominated solutions generated during the search. We consider two bi-objective optimization problems, OneMinMax and LeadingOnes-TrailingZeroes, which were commonly used in theoretical studies of MOEAs [Laumanns *et al.*, 2004; Doerr *et al.*, 2013; Nguyen *et al.*, 2015; Bian and Qian, 2022; Zheng and Doerr, 2023a].

The results are given in Table 1. As can be seen in the table, using an archive allows a small constant population size, which brings an acceleration of $\Theta(n)$ on the expected running time. Note that the original algorithms without using an archive require the population size μ to be at least the same size of the problem's Pareto front (i.e., $\mu \geq 2(n+1)$ for NSGA-II and $\mu \geq n+1$ for SMS-EMOA, where $n+1$ is the size of the Pareto front), otherwise Pareto optimal solutions can be lost even if they have been found previously. Using an archive that stores all the Pareto optimal solutions generated enables the algorithms not to worry about losing Pareto optimal solutions, but only endeavoring to seek new Pareto optimal solutions, thus speeding up the search process.

Over the past two decades, there is a substantial number of theoretical studies in the area of MOEAs, particularly regarding their running time complexity analyses. It starts with the analysis of a simple evolutionary multi-objective optimizer (SEMO) and its global variant, GSEMO, for solving multi-objective synthetic and combinatorial optimization problems [Giel, 2003; Laumanns *et al.*, 2004; Neumann, 2007; Giel and Lehre, 2010; Neumann and Theile, 2010; Doerr *et al.*, 2013; Qian *et al.*, 2013; Bian *et al.*, 2018]. On the other side, based on SEMO and GSEMO, the effectiveness of several parent selection and reproduction methods has also been studied [Laumanns *et al.*, 2004; Friedrich *et al.*, 2011; Qian *et al.*, 2013; Qian *et al.*, 2016; Doerr and Zheng, 2021].

Recently, attention is being shifted to the analysis of practical MOEAs. The expected running time of $(\mu+1)$ SIBEA, i.e., an 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]. Very recently, Zheng and Doerr [2023a] analyzed the expected running time of NSGA-II, for the first time, by considering the bi-objective OneMinMax and LeadingOnes-TrailingZeroes problems. Since then, the effectiveness of different components and mechanisms in NSGA-II, e.g., crowding distance [Zheng and Doerr, 2022], stochastic tournament selection [Bian and Qian, 2022], fast mutation [Doerr and Qu, 2023a], crossover [Dang *et al.*, 2023b; Doerr and Qu, 2023c], and diversity maintenance [Ren *et al.*, 2024], has also been analyzed. More results on NSGA-II include [Cerf *et al.*, 2023; Doerr and Qu, 2023a; Doerr and Qu, 2023b; Zheng and Doerr, 2023b]. Furthermore, the expected running time of other well-established MOEAs has also been analyzed, e.g., MOEA/D [Huang *et al.*, 2019; Huang and Zhou, 2020; Huang *et al.*, 2021], SMS-EMOA [Bian *et al.*, 2023; Ren *et al.*, 2024; Zheng and Doerr, 2024], and NSGA-III [Wietheger and Doerr, 2023], in addition to the analysis of them under different optimization models such as under noise [Dang *et al.*, 2023a; Dinot *et al.*, 2023] and the interactive model [Lu *et al.*, 2024].

Yet, all the above work regarding practical MOEAs needs a population with a commensurable size of the problem's Pareto front. This may not be very practical since one may not be able to know the size of the problem's Pareto front before the optimization. The proposed work in this paper addresses this issue and proves that a small population, with an archive, even works better. This result not only provides a theoretical confirmation for an increasingly popular practice in the development of MOEAs, but can also be beneficial to the theory community towards studying more practical MOEAs.

2 Preliminaries

In this section, we first give basic concepts in multi-objective optimization, which is followed by the considered algorithms NSGA-II and SMS-EMOA, and the archive mechanism. Lastly, we describe the OneMinMax and LeadingOnesTrailingZeroes problems studied in this paper.

2.1 Multi-objective Optimization

Multi-objective optimization aims to optimize two or more objective functions simultaneously, as presented in Definition 1. In this paper, we consider maximization (minimization

can be defined similarly), and pseudo-Boolean functions, i.e., the solution space $\mathcal{X} = \{0,1\}^n$. Since the objectives are usually conflicting, there does not exist 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 it is not dominated by any other solution in $\mathcal{X}$, and the set of objective vectors of all the Pareto optimal solutions is called the *Pareto front*. The goal of multi-objective optimization is to find the Pareto front or its good approximation.

Definition 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 $1 \le i \le m$, $f_i(\boldsymbol{x}) \ge f_i(\boldsymbol{y})$;*
- *$\boldsymbol{x}$* 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 $\boldsymbol{y} \succeq \boldsymbol{x}$.*

2.2 NSGA-II and SMS-EMOA

The NSGA-II algorithm [Deb *et al.*, 2002], as presented in Algorithm 1, is a very popular MOEA which incorporates two substantial features, i.e., non-dominated sorting and crowding distance. It starts from an initial population of μ (without loss of generality, we assume that μ is even) random solutions (line 1). In each generation, NSGA-II employs binary tournament selection to select parent solutions (line 5), which picks two solutions randomly from the population P with replacement, and then selects a better one as the parent solution (ties broken uniformly). Then, one-point crossover is performed on the two parent solutions with probability p_c (lines 6–11), which selects a crossover point $i \in \{1, 2, \ldots, n\}$ uniformly at random, where n is the problem size, and then exchanges the first i bits of two solutions. The bit-wise mutation operator, which flips each bit of a solution independently with probability $1/n$, is then applied to generate offspring solutions (line 12). After a set P' of μ offspring solutions have been generated, the solutions in the current and offspring populations are partitioned into non-dominated sets $R_1, \ldots, R_v$ (line 15), where R_1 contains all the non-dominated solutions in $P \cup P'$, and R_i ($i \ge 2$) contains all the non-dominated solutions in $(P \cup P') \setminus \cup_{j=1}^{i-1} R_j$. Note that a solution is said to be with rank i if it belongs to R_i. Then, the solutions in $R_1, \ldots, R_v$ are added into the next population, until the population size exceeds μ (lines 16–19). For the critical set R_i whose inclusion makes the population size larger than μ, the crowding distance is computed for each of the contained solutions (line 20). Crowding distance reflects the diversity of a solution. For each objective f_j, $1 \le j \le m$, the solutions in R_i are sorted according to their objective values in ascending order, and we assume the sorted list is $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^k$; the crowding distance of the solution $\boldsymbol{x}^l$ with respect to f_j is set to ∞ if $l \in \{1, k\}$, and $(f_j(\boldsymbol{x}^{l+1}) - f_j(\boldsymbol{x}^{l-1}))/(f_j(\boldsymbol{x}^k) - f_j(\boldsymbol{x}^1))$ otherwise. The final crowding distance of a solution is the sum of the crowding distance with respect to each objective. Finally, the solutions in R_i are selected to fill the remaining population slots where the solutions with larger crowding distance are preferred (line 21). Note that when using binary tournament selection in line 5, the selection criterion is based on rank and crowding distance, that is, a solution $\boldsymbol{x}$ is superior to $\boldsymbol{y}$ if $\boldsymbol{x}$ has a smaller rank, or $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same rank but $\boldsymbol{x}$ has a larger crowding distance than $\boldsymbol{y}$. The probability p_c of using crossover is set to 0.9, just as the setting in [Deb *et al.*, 2002].

Algorithm 1 NSGA-II [Deb *et al.*, 2002]

Input: objective functions $f_1, f_2 \ldots, f_m$, population size μ, probability p_c of using crossover

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

The SMS-EMOA algorithm [Beume *et al.*, 2007] as presented in Algorithm 2 is also a popular MOEA, which employs non-dominated sorting and hypervolume indicator to update the population. Starting from an initial population of μ random solutions (line 1), in each generation, it randomly selects a parent solution $\boldsymbol{x}$ from the current population for reproduction (line 3). With probability p_c (similar to NSGA-II, p_c is set to 0.9), 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 actually produces two 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 is applied on $\boldsymbol{x}'$ to generate one offspring solution (line 11). Then, similar to line 15 of Algorithm 1,

Algorithm 2 SMS-EMOA [Beume *et al.*, 2007]

Input: objective functions $f_1, f_2 \ldots, f_m$, population size μ, probability p_c of using crossover

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 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 one solution $\boldsymbol{x}'$
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: partition $P \cup \{\boldsymbol{x}''\}$ into non-dominated sets $R_1, ..., R_v$;
13: let $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in R_v} \Delta_{\boldsymbol{r}}(\boldsymbol{x}, R_v)$;
14: $P \leftarrow (P \cup \{\boldsymbol{x}''\}) \setminus \{\boldsymbol{z}\}$
15: **end while**
16: **return** P

the union of the current population and the newly generated offspring solution is partitioned into non-dominated sets $R_1, \ldots, R_v$ (line 12), and 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 13–14), where $HV_{\boldsymbol{r}}(X) = \Lambda\big(\cup_{\boldsymbol{x} \in X} \{\boldsymbol{f}' \in \mathbb{R}^m \mid \forall 1 \leq i \leq m : r_i \leq f_i' \leq f_i(\boldsymbol{x})\}\big)$ denotes the hypervolume of a solution set X with respect to a reference point $\boldsymbol{r} \in \mathbb{R}^m$ (satisfying $\forall 1 \leq i \leq m, r_i \leq \min_{\boldsymbol{x} \in \mathcal{X}} f_i(\boldsymbol{x})$), i.e., the volume of the objective space between the reference point and the objective vectors of the solution set, and Λ denotes the Lebesgue measure. A larger hypervolume value implies a better approximation with regards to both convergence and diversity. Note that when SMS-EMOA solves bi-objective problems, we use the original setting in [Beume *et al.*, 2007] that the two extreme points (i.e., the objective vectors which contain the largest objective value for some f_i, $i \in \{1, 2\}$) are always kept in the population regardless of their hypervolume loss.

2.3 NSGA-II and SMS-EMOA with an Archive

In the original NSGA-II and SMS-EMOA, there is no archive used. As explained previously, the non-dominated solutions may lose even if they have been found once. Using an archive can easily address this issue. That is, 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. Additional algorithmic steps incurred by adding an archive in NSGA-II and SMS-EMOA are given as follows. For NSGA-II in Algorithm 1, an empty set Q is initialized in line 1, the following lines

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

are added after line 14, and the set Q instead of P is returned in the last line. For SMS-EMOA in Algorithm 2, an empty set Q is also initialized in line 1, the following lines

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

are added after line 11, and the set Q instead of P is returned in the last line.

2.4 OneMinMax and LeadingOnesTrailingZeroes

Now we introduce two bi-objective problems, OneMinMax and LeadingOnesTrailingZeroes considered in this paper. Theses two problems have been widely used in MOEAs' theoretical analyses [Laumanns *et al.*, 2004; Brockhoff *et al.*, 2008; Doerr *et al.*, 2013; Qian *et al.*, 2013; Bian and Qian, 2022; Zheng and Doerr, 2023a].

The OneMinMax problem presented in Definition 3 aims to simultaneously maximize the number of 0-bits and the number of 1-bits of a binary bit string. The Pareto front is $\{(a, n-a) \mid a \in [0..n]\}$, whose size is $n+1$, and the Pareto optimal solution corresponding to $(a, n-a)$, $a \in [0..n]$, is any solution with $(n-a)$ 1-bits. Note that we use $[l..r]$ to denote the set $\{l, l+1, \ldots, r\}$ of integers throughout the paper. We can see that any solution $\boldsymbol{x} \in \{0,1\}^n$ is Pareto optimal for this problem.

Definition 3 (OneMinMax [Giel and Lehre, 2010])**.** *The OneMinMax problem of size n is to find n bits binary strings which maximize $\boldsymbol{f}(\boldsymbol{x}) = (n - \sum_{i=1}^n x_i, \sum_{i=1}^n x_i)$, where x_i denotes the i-th bit of $\boldsymbol{x} \in \{0,1\}^n$.*

The LeadingOnesTrailingZeroes problem presented in Definition 4 aims to simultaneously maximize the number of leading 1-bits and the number of trailing 0-bits of a binary bit string. The Pareto front is $\{(a, n-a) \mid a \in [0..n]\}$, whose size is $n+1$, and the Pareto optimal solution corresponding to $(a, n-a)$, $a \in [0..n]$, is $1^a 0^{n-a}$, i.e., the solution with a leading 1-bits and $n-a$ trailing 0-bits.

Definition 4 (LeadingOnesTrailingZeroes [Laumanns *et al.*, 2004])**.** *The LeadingOnesTrailingZeroes problem of size n is to find n bits binary strings which maximize $\boldsymbol{f}(\boldsymbol{x}) = (\sum_{i=1}^n \prod_{j=1}^i x_j, \sum_{i=1}^n \prod_{j=i}^n (1 - x_j))$, where x_j denotes the j-th bit of $\boldsymbol{x} \in \{0,1\}^n$.*

3 Analysis of NSGA-II with an Archive

In this section, we analyze the expected running time of NSGA-II in Algorithm 1 using an archive. Note that the running time of an EA is usually measured by the number of fitness evaluations, which is often the most time-consuming step in the evolutionary process. We prove in Theorem 1 that the expected number of fitness evaluations of NSGA-II using an archive for solving OneMinMax is $O(\mu n \log n)$, where the population size $\mu \geq 4$. The proof idea is to divide the optimization procedure into two phases, where the first phase aims at finding the two extreme Pareto optimal solutions 1^n

and 0^n, and the second phase aims at finding the remaining objective vectors in the Pareto front.

Theorem 1. *For NSGA-II solving OneMinMax, if using an archive, and a population size μ such that $\mu \geq 4$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n \log n)$.*

Proof. We divide the running process of NSGA-II into two phases. The first phase starts after initialization and finishes until 1^n and 0^n are both found; the second phase starts after the first phase and finishes when the Pareto front is found.

For the first phase, we will prove that the expected number of generations for finding 0^n is $O(n \log n)$, and then the same bound holds for finding 1^n analogously. We first show that the maximal f_1 value of the solutions in the population P, i.e., $\max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_0$, will not decrease, where $|\boldsymbol{x}|_0$ denotes the number of 0-bits in $\boldsymbol{x}$. Let C denote the set of solutions in $P \cup P'$ with the maximal f_1 value, where P' denotes the offspring population. Because of $P \subseteq P \cup P'$ and the definition of C, we have for any $\boldsymbol{x} \in C$, $|\boldsymbol{x}|_0 \geq \max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_0$. Thus, we only need to show that one solution in C will be maintained in the next population. Because all the solutions in C have the maximal f_1 value and the same f_2 value, they cannot be dominated by any solution in $P \cup P'$, implying that they all have rank 1, i.e., belong to R_1 in the non-dominated sorting procedure. If $|R_1| \leq \mu$, all the solutions in C will be maintained in the next population. If $|R_1| > \mu$, the crowding distance of the solutions in R_1 needs to be computed. When the solutions in R_1 are sorted according to f_1 in ascending order, one solution in C (denoted as $\boldsymbol{x}^*$) must be put in the last position and thus has infinite crowding distance. Note that only solutions in the first and the last positions can have infinite crowding distance. As OneMinMax has two objectives, at most four solutions in R_1 can have infinite crowding distance. Thus, $\boldsymbol{x}^*$ is among the best four solutions in R_1 and must be included in the next population (note that $\mu \geq 4$), implying that the maximal f_1 value will not decrease.

Next, we analyze the increase of the maximal f_1 value. Assume that currently the maximal f_1 value is i ($i < n$), i.e., $\max_{\boldsymbol{x} \in P} |\boldsymbol{x}|_0 = i$. When using binary tournament selection to select a parent solution, the competition between the two randomly selected solutions is based on rank and crowding distance (note that the rank and crowding distance here are computed based on the current population P instead of the union of the current population and the offspring population) with ties broken uniformly. As analyzed in the last paragraph, a solution $\boldsymbol{x} \in P$ with $|\boldsymbol{x}|_0 = i$ must have rank 1 and infinite crowding distance. Once $\boldsymbol{x}$ is selected for competition (whose probability is $1/\mu$), it will always win, if the other solution selected for competition has larger rank or finite crowding distance; or win with probability $1/2$, if the other solution has the same rank and crowding distance as $\boldsymbol{x}$, resulting in a tie which is broken uniformly at random. Thus, $\boldsymbol{x}$ can be selected as a parent solution with probability at least $1/(2\mu)$. In the reproduction procedure, a solution with more 0-bits can be generated from $\boldsymbol{x}$ if crossover is not performed (whose probability is $1 - 0.9 = 0.1$) and only one of the 1-bits in $\boldsymbol{x}$ is flipped by bit-wise mutation (whose probability is $((n - |\boldsymbol{x}|_0)/n) \cdot (1-1/n)^{n-1}$). Thus, the probability of generating a solution with more than i 0-bits is at least

$$\frac{1}{2\mu} \cdot 0.1 \cdot \frac{n - |\boldsymbol{x}|_0}{n} \cdot \Big(1 - \frac{1}{n}\Big)^{n-1} \geq \frac{n-i}{20e\mu n}. \tag{1}$$

Because in each generation, $\mu/2$ pairs of parent solutions will be selected for reproduction, the probability of generating a solution with more than i 0-bits is at least

$$\begin{aligned} 1 - \Big(1 - \frac{n-i}{20e\mu n}\Big)^{\mu/2} &\geq 1 - \frac{1}{e^{(n-i)/(40en)}} \\ &\geq 1 - \frac{1}{1 + (n-i)/(40en)} = \Omega\Big(\frac{n-i}{n}\Big), \end{aligned}$$

where the inequalities hold by $1 + a \leq e^a$ for any $a \in \mathbb{R}$. Because the solution with the most number of 0-bits will be maintained in the population, the expected number of generations for increasing the maximal f_1 value to n, i.e., finding 0^n, is at most $\sum_{i=0}^{n-1} O(n/(n-i)) = O(n \log n)$. That is, the expected number of generations of the first phase is $O(n \log n)$.

Now we consider the second phase, and will show that NSGA-II can find the whole Pareto front in $O(n \log n)$ expected numbers of generations. Note that after phase 1, 0^n and 1^n must be maintained in the population P. Let $D = \{j \mid \exists \boldsymbol{x} \in Q, |\boldsymbol{x}|_0 = j\}$, where Q denotes the archive, and we suppose $|D| = i$, i.e., i points in the Pareto front has been found in the archive. Note that $i \geq 2$ as 0^n and 1^n have been found. Next, we consider two cases.

(1) The number of 1^n or the number of 0^n in the current population P is at least $\mu/4$. Without loss of generality, we assume that the number of 0^n is at least $\mu/4$. Then, the probability of selecting 0^n as a parent solution is at least $(1/4)^2 = 1/16$, because it is sufficient to select 0^n twice in binary tournament selection. According to the analysis in the paragraph above Eq. (1), the probability of selecting 1^n as the other parent solution is at least $1/(2\mu)$. After exchanging the first k ($k \in [0..n] \setminus D$) bits of 0^n and 1^n by one-point crossover (the probability is $0.9 \cdot (1/n)$), a solution with k 0-bits can be generated, which can keep unchanged after bit-wise mutation if none of the bits is flipped (the probability is $(1-1/n)^n$). Thus, the probability of generating a new point in the Pareto front is at least

$$\begin{aligned} &\frac{1}{16} \cdot \frac{1}{2\mu} \cdot 0.9 \cdot \frac{n+1-i}{n} \cdot \Big(1 - \frac{1}{n}\Big)^n \\ &\geq \frac{n+1-i}{32\mu n} \cdot 0.9 \cdot \Big(1 - \frac{1}{n}\Big) \cdot \frac{1}{e} \geq \frac{n+1-i}{64e\mu n}, \end{aligned} \tag{2}$$

where the term $n+1-i$ is because there are $|[0..n] \setminus D| = n + 1 - i$ points in the Pareto front to be found, the first inequality holds by $(1-1/n)^{n-1} \geq 1/e$, and the second inequality holds for $n \geq 3$.

(2) The number of 1^n and the number of 0^n in the current population P are both less than $\mu/4$. Then, in one binary tournament selection procedure, the probability of selecting two solutions with the number of 0-bits in $[1..n-1]$ is at least $(\mu - \mu/4 - \mu/4)^2/\mu^2 = 1/4$, and we assume that the winning solution $\boldsymbol{x}$ has j ($j \in [1..n-1]$) 0-bits. If the other selected parent solution is 0^n, then for any $k \in [j+1..n] \setminus D$, there must exist a crossover point k' such that exchanging the first k' bits of $\boldsymbol{x}$ and 0^n can generate a solution with k 0-bits. If the other selected parent solution is 1^n, then for any

$k \in [0..j-1] \setminus D$, there must exist a crossover point k' such that exchanging the first k' bits of $\boldsymbol{x}$ and 1^n can generate a solution with k 0-bits. The newly generated solution can keep unchanged by flipping no bits in bit-wise mutation. Note that the probability of selecting 1^n (or 0^n) as a parent solution is at least $1/(2\mu)$. Thus, similar to Eq. (2), the probability of generating a new point in the Pareto front is at least

$$\frac{1}{4} \cdot \frac{1}{2\mu} \cdot 0.9 \cdot \frac{n+1-|D|}{n} \cdot \Big(1-\frac{1}{n}\Big)^n \geq \frac{n+1-i}{16e\mu n}. \quad (3)$$

By taking the smaller one between Eqs. (2) and (3), and using the fact that NSGA-II produces $\mu/2$ pairs of offspring solutions in each generation, the probability of generating a new point in the Pareto front in each generation is at least

$$1-\Big(1-\frac{n+1-i}{64e\mu n}\Big)^{\mu/2} = \Omega\Big(\frac{n+1-i}{n}\Big).$$

Then, we can derive that the expected number of generations of the second phase (i.e., for finding the whole Pareto front) is at most $\sum_{i=2}^{n} O(n/(n+1-i)) = O(n \log n)$.

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

We prove in Theorem 2 that the expected number of fitness evaluations of NSGA-II using an archive for solving LeadingOnesTrailingZeroes is $O(\mu n^2 + \mu^2 n \log n)$, where the population size $\mu \geq 4$. As the proof of Theorem 1, we divide the optimization procedure into two phases, that is to find the extreme solutions 1^n and 0^n, and to find the whole Pareto front, respectively. But due to the fact that only solutions with the form $1^j 0^{n-j}$ are Pareto optimal for LeadingOnesTrailingZeroes while any solution is Pareto optimal for OneMinMax, their analyses of increasing the maximal f_1 value in the first phase as well as generating new Pareto optimal solutions in the second phase are different. The detailed proof of Theorem 2 is given in the supplementary due to space limitation.

Theorem 2. *For NSGA-II solving LeadingOnesTrailingZeroes, if using an archive, and a population size μ such that $\mu \geq 4$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^2 + \mu^2 n \log n)$.*

The expected number of fitness evaluations of the original NSGA-II (without using an archive) for solving OneMinMax and LeadingOnesTrailingZeroes has been shown to be $O(\mu n \log n)$ and $O(\mu n^2)$, respectively, where the population size $\mu \geq 2(n+1)$ [Bian and Qian, 2022]. Thus, our results in Theorems 1 and 2 show that if a constant population size is used for NSGA-II having an archive, the expected running time can be reduced by a factor of $\Theta(n)$. The main reason for the acceleration is that the archive can preserve all the non-dominated solutions generated so far, which enables NSGA-II to discard solutions that are not critical for finding the Pareto front, and thus to use a small population bringing more efficient exploration of the search space.

4 Analysis of SMS-EMOA with an Archive

In this section, we consider SMS-EMOA in Algorithm 2 using an archive. We prove in Theorems 3 and 4 that the expected number of fitness evaluations of SMS-EMOA using an archive for solving OneMinMax and LeadingOnesTrailingZeroes is $O(\mu n \log n)$ and $O(\mu n^2 + \mu^2 n \log n)$, respectively, where the population size $\mu \geq 2$. Their proofs are similar to that of Theorems 1 and 2, respectively. That is, we divide the optimization procedure into two phases, where the first phase aims at finding 1^n and 0^n, and the second phase aims at finding the whole Pareto front. The main difference of the proofs is led by that 1) during the population update procedure, SMS-EMOA directly preserves two boundary objective vectors which contain the largest objective value for f_1 or f_2, and 2) during the reproduction procedure, it uses uniform parent selection and generates only one offspring solution in each generation. The detailed proof of Theorem 4 is given in the supplementary due to space limitation.

Theorem 3. *For SMS-EMOA solving OneMinMax, if using an archive, and a population size μ such that $\mu \geq 2$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n \log n)$.*

Proof. For the first phase, the maximal f_1 value will not decrease because SMS-EMOA directly keeps the two boundary points in the population update procedure. Now, we consider the increase of the maximal f_1 value. Note that SMS-EMOA in Algorithm 2 selects a parent solution uniformly at random, instead of using binary tournament selection; thus, the probability of selecting any specific solution in the population is $1/\mu$. Eq. (1) changes to

$$\frac{1}{\mu} \cdot 0.1 \cdot \frac{n-|\boldsymbol{x}|_0}{n} \cdot \Big(1-\frac{1}{n}\Big)^{n-1} \geq \frac{n-i}{10e\mu n}.$$

Different from NSGA-II which produces $\mu/2$ pairs of offspring solutions in each generation, SMS-EMOA only reproduces one solution in each generation. Thus, the expected number of generations for increasing the maximal f_1 value to n, i.e., finding 0^n, is at most $\sum_{i=0}^{n-1} 10e\mu n/(n-i) = O(\mu n \log n)$. That is, the expected number of generations of the first phase is $O(\mu n \log n)$.

For the second phase, by considering the difference between uniform parent selection employed by SMS-EMOA and binary tournament selection employed by NSGA-II, Eq. (2) changes to

$$\frac{1}{4} \cdot \frac{1}{\mu} \cdot 0.9 \cdot \frac{n+1-i}{n} \cdot \Big(1-\frac{1}{n}\Big)^n \geq \frac{n+1-i}{8e\mu n},$$

and Eq. (3) changes to

$$\frac{1}{2} \cdot \frac{1}{\mu} \cdot 0.9 \cdot \frac{n+1-|D|}{n} \cdot \Big(1-\frac{1}{n}\Big)^n \geq \frac{n+1-i}{4e\mu n}.$$

Thus, the expected number of generations of the second phase (i.e., for finding the whole Pareto front) is at most $\sum_{i=2}^{n} 8e\mu n/(n+1-i) = O(\mu n \log n)$.

Combining the two phases, the total expected number of generations is $O(\mu n \log n)$. Since SMS-EMOA only generates one solution in generation, the expected number of fitness evaluations is also $O(\mu n \log n)$. □

Theorem 4. *For SMS-EMOA solving LeadingOnesTrailingZeroes, if using an archive, and a population size μ such that $\mu \geq 2$, then the expected number of fitness evaluations for finding the Pareto front is $O(\mu n^2 + \mu^2 n \log n)$.*

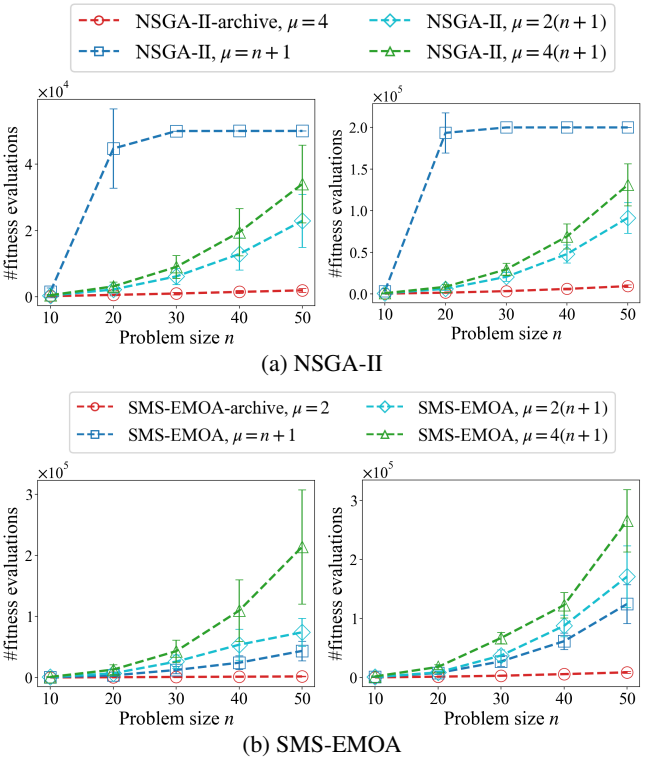
The expected number of fitness evaluations of the original SMS-EMOA (without crossover) solving OneMinMax and LeadingOnesTrailingZeroes has been shown to be $O(\mu n \log n)$ and $O(\mu n^2)$, respectively, where the population size $\mu \geq n+1$ [Zheng and Doerr, 2024]. Though their analysis does not consider crossover, the running time bounds still hold asymptotically here as the crossover operator is not performed with a constant probability 0.1 in Algorithm 2. Note that the size of the Pareto front is $n+1$, thus when $\mu < n+1$, the whole Pareto front cannot be covered. Therefore, our results in Theorems 3 and 4 show that if a constant population size is used for SMS-EMOA having an archive, the expected running time can be accelerated by a factor of $\Theta(n)$. The main reason for the acceleration is similar to that of NSGA-II. That is, introducing the archive enables SMS-EMOA to only preserve essential solutions for locating the Pareto front, and thus makes the exploration more efficient.

5 Experiments

In the previous sections, we have proved that when an archive is used in NSGA-II and SMS-EMOA, the expected number of fitness evaluations for solving OneMinMax and LeadingOnesTrailingZeroes can be reduced by a factor of $\Theta(n)$. However, as only upper bounds on the running time of the original NSGA-II and SMS-EMOA have been derived, we conduct experiments to examine their actual performance to complement the theoretical results.

Specifically, we set the problem size n from 10 to 50, with a step of 10. For NSGA-II and SMS-EMOA using an archive, the population size μ is set to 4 and 2, respectively, as suggested in Theorems 1–4. For the original NSGA-II and SMS-EMOA without an archive, we test three values of μ, i.e., $n+1$, $2(n+1)$, and $4(n+1)$, as suggested by [Bian and Qian, 2022; Zheng and Doerr, 2023a; Zheng and Doerr, 2024]. Note that $n+1$ is the size of the Pareto front of OneMinMax and LeadingOnesTrailingZeroes. If using a population size $\mu < n+1$, the original algorithms without archive obviously cannot find the Pareto front. For each n, we run an algorithm 1000 times independently, and record the average number of fitness evaluations until the Pareto front is found. In case where the Pareto front cannot be found in acceptable running time, we set the maximum number of fitness evaluations to 5×10^4 for OneMinMax and 2×10^5 for LeadingOnesTrailingZeroes. We can observe from Figure 1 that using an archive brings a clear acceleration.

We can also observe that when using a population size of $n+1$, SMS-EMOA performs much better than NSGA-II. The main reason is that for NSGA-II, when two Pareto optimal solutions corresponding to one objective vector (e.g., two solutions corresponding to one boundary point in the Pareto front) have been found, they both can have fairly large crowding distance, and thus occupy two slots in the population. Then, the population size of $n+1$ is not sufficient to ensure the preservation of objective vectors in the Pareto front. However, for SMS-EMOA, these duplicate Pareto optimal solutions have a zero hypervolume contribution, and thus are less preferred, implying that they will not affect the preservation of other objective vectors in the Pareto front. Meanwhile, since SMS-EMOA only removes one solution in each generation, the objective vector corresponding to these duplicate solutions will also be preserved. Thus, a population size of $n+1$ is sufficient for SMS-EMOA to preserve the whole Pareto front.



(a) NSGA-II

(b) SMS-EMOA

Figure 1: Average number of fitness evaluations of NSGA-II and SMS-EMOA with or without an archive for solving the OneMinMax and LeadingOnesTrailingZeroes problems. Left subfigure: OneMinMax; right subfigure: LeadingOnesTrailingZeroes.

6 Conclusion

In this paper, we perform a first theoretical study for MOEAs with an archive, an increasingly popular practice in the design of MOEAs. Through rigorous running time analysis for NSGA-II and SMS-EMOA solving two commonly studied bi-objective problems, OneMinMax and LeadingOnesTrailingZeroes, we prove that using an archive can allow a constant population size, bringing an acceleration of factor $\Theta(n)$. This is also verified by the experiments. Our results provide theoretical confirmation for the benefit of using an archive to store non-dominated solutions generated during the search process of MOEAs, which has frequently been observed empirically [Fieldsend *et al.*, 2003; Bezerra *et al.*, 2019]. In the future, it would be interesting to derive the lower bounds of NSGA-II and SMS-EMOA without using an archive to make the comparison strict. Another interesting direction is studying real-world problems, e.g., multi-objective combinatorial optimization problems.

Acknowledgements

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

References

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

[Bezerra *et al.*, 2019] L. C. T. Bezerra, M. López-Ibáñez, and T. Stützle. Archiver effects on the performance of state-of-the-art multi- and many-objective evolutionary algorithms. In *GECCO*, 2019.

[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, 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, 2018.

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

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

[Brockhoff *et al.*, 2008] D. Brockhoff, T. Friedrich, and F. Neumann. Analyzing hypervolume indicator based algorithms. In *PPSN*, pages 651–660, 2008.

[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, 2023.

[Chen *et al.*, 2019] T. Chen, M. Li, and X. Yao. Standing on the shoulders of giants: Seeding search-based multi-objective optimization with prior knowledge for software service composition. *Information and Software Technology*, 114:155–175, 2019.

[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, 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, 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 (TEVC)*, 6(2):182–197, 2002.

[Deb *et al.*, 2005] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pages 105–145. 2005.

[Dinot *et al.*, 2023] M. Dinot, B. Doerr, U. Hennebelle, and S. Will. Runtime analyses of multi-objective evolutionary algorithms in the presence of noise. In *IJCAI*, pages 5549–5557, 2023.

[Doerr and Qu, 2023a] B. Doerr and Z. Qu. A first runtime analysis of the NSGA-II on a multimodal problem. *TEVC*, 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, 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, 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, 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, 2013.

[Doerr *et al.*, 2016] B. Doerr, W. Gao, and F. Neumann. Runtime analysis of evolutionary diversity maximization for OneMinMax. In *GECCO*, pages 557–564, 2016.

[Dubois-Lacoste *et al.*, 2015] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Anytime Pareto local search. *EJOR*, 243(2):369–385, 2015.

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

[Fieldsend, 2017] J. E. Fieldsend. University staff teaching allocation: formulating and optimising a many-objective problem. In *GECCO*, pages 1097–1104, 2017.

[Fieldsend, 2020] J. E. Fieldsend. Data structures for non-dominated sets: implementations and empirical assessment of two decades of advances. In *GECCO*, pages 489–497, 2020.

[Friedrich *et al.*, 2011] T. Friedrich, C. Horoba, and F. Neumann. Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science (TCS)*, 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 (ECJ)*, 18(3):335–356, 2010.

[Giel, 2003] O. Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *CEC*, pages 1918–1925, 2003.

[Glasmachers, 2017] T. Glasmachers. A fast incremental BSP tree archive for non-dominated points. In *EMO*, pages 252–266, 2017.

[Huang and Zhou, 2020] Z. Huang and Y. Zhou. Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In *AAAI*, pages 2359–2366, 2020.

[Huang *et al.*, 2019] Z. Huang, Y. Zhou, Z. Chen, and X. He. Running time analysis of MOEA/D with crossover on discrete optimization problem. In *AAAI*, pages 2296–2303, 2019.

[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, 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.

[Jaszkiewicz and Lust, 2018] A. Jaszkiewicz and T. Lust. ND-tree-based update: a fast algorithm for the dynamic nondominance problem. *TEVC*, 22(5):778–791, 2018.

[Knowles and Corne, 2004] J. Knowles and D. Corne. Bounded Pareto archiving: Theory and practice. In *Metaheuristics for Multiobjective Optimisation*, pages 39–64. Springer, 2004.

[Laumanns *et al.*, 2002] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *ECJ*, 10(3):263–282, 2002.

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

[Li and Yao, 2019] M. Li and X. Yao. An empirical investigation of the optimality and monotonicity properties of multiobjective archiving methods. In *EMO*, pages 15–26, 2019.

[Li *et al.*, 2021] B. Li, Y. Zhang, P. Yang, X. Yao, and A. Zhou. A two-population algorithm for large-scale multi-objective optimization based on fitness-aware operator and adaptive environmental selection. *TEVC*, 14(8):1–15, 2021.

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

[Li *et al.*, 2023b] M. Li, M. López-Ibáñez, and X. Yao. Multi-objective archiving. *TEVC*, 2023.

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

[Nan *et al.*, 2020] Y. Nan, K. Shang, H. Ishibuchi, and L. He. Reverse strategy for non-dominated archiving. *IEEE Access*, 8:119 458–119 469, 2020.

[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, 2010.

[Neumann, 2007] F. Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *EJOR*, 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. *TCS*, 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 (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, 2016.

[Ren *et al.*, 2024] S. Ren, Z. Qiu, C. Bian, M. Li, and C. Qian. Maintaining diversity provably helps in evolutionary multimodal optimization. In *IJCAI*, page to appear, 2024.

[Tanabe and Oyama, 2017] R. Tanabe and A. Oyama. Benchmarking moeas for multi-and many-objective optimization using an unbounded external archive. In *GECCO*, pages 633–640, 2017.

[Wang *et al.*, 2019] H. Wang, C. Sun, Y. Jin, S. Qin, and H. Yu. A multi-indicator based selection strategy for evolutionary many-objective optimization. In *CEC*, pages 2042–2049, 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, 2023.

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

[Zhang *et al.*, 2023] J. Zhang, L. He, and H. Ishibuchi. Dual fuzzy classifier-based evolutionary algorithm for expensive multiobjective optimization. *TEVC*, 27(6):1575–1589, 2023.

[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, 2022.

[Zheng and Doerr, 2023a] W. Zheng and B. Doerr. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *AIJ*, 325:104016, 2023.

[Zheng and Doerr, 2023b] W. Zheng and B. Doerr. Runtime analysis for the NSGA-II: Proving, quantifying, and explaining the inefficiency for many objectives. *TEVC*, 2023.

[Zheng and Doerr, 2024] W. Zheng and B. Doerr. Runtime analysis of the SMS-EMOA for many-objective optimization. In *AAAI*, 2024.