Speeding Up the NSGA-II via Dynamic Population Sizes

Benjamin Doerr[1] , **Martin S. Krejca**[1] , **Simon Wietheger**[2]
[1]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris
[2]Algorithms and Complexity Group, TU Wien
{firstname.lastname}@polytechnique.edu, swietheger@ac.tuwien.ac.at,

Abstract

Multi-objective evolutionary algorithms (MOEAs) are among the most widely and successfully applied optimizers for multi-objective problems. However, to store many optimal trade-offs (the Pareto optima) simultaneously, MOEAs are typically run with a large population of solution candidates. This slows down the algorithm and renders the choice of the population size a crucial design decision. In this work, we aim to overcome these difficulties by proposing the *dynamic NSGA-II*, a variant of the well-known NSGA-II that starts with a small initial population and doubles it after a user-specified number τ of function evaluations, up to a maximum size of $N_{\max}$. We prove that the dynamic NSGA-II with optimal parameters computes the Pareto front of the OneMinMax benchmark of size n with high probability in $O(n \log^2 n)$ function evaluations, which is considerably faster than the $\Theta(n^2 \log n)$ runtime of the static NSGA-II with optimal parameters. For the OneJumpZeroJump benchmark with gap size k, we show a runtime of $O(n^k \log^2 n)$, improving upon the known runtime of $\Theta(n^{k+1})$. We also propose a variant that uses the initial population size for a longer period and achieves slightly better performance. Finally, we show that a simple concurrent-run strategy turns our dynamic NSGA-II variants into parameter-less algorithms that exceed the above runtimes only by a logarithmic factor and hence still outperform the static NSGA-II by a factor of $\tilde{\Omega}(n)$.

1 Introduction

Real-world problems often require the optimization of conflicting objectives, resulting in several incomparable optimal trade-offs, known as *Pareto optima*. One solution concept for such multi-objective optimization problems is to compute several Pareto optima, ideally all of them (the *Pareto front*), and let a decision maker select the final solution. *Multi-objective evolutionary algorithms (MOEAs)* are a natural choice for this task as they usually work with sets of solutions (*populations*) and, in fact, MOEAs are among the most successful and widely applied approaches in multi-objective optimization [Coello *et al.*, 2007; Zhou *et al.*, 2011].

Recently, the empirical success of popular state-of-the-art MOEAs, such as the NSGA-II [Deb *et al.*, 2002], NSGA-III [Deb and Jain, 2014], SMS-EMOA [Beume *et al.*, 2007], MOEA/D [Zhang and Li, 2007], or SPEA2 [Zitzler *et al.*, 2001], has been supported by the first mathematical analyses of these algorithms [Zheng *et al.*, 2022; Wietheger and Doerr, 2023; Bian *et al.*, 2023; Li *et al.*, 2016; Ren *et al.*, 2024]. This not only gave rigorous performance guarantees for MOEAs but also led to deeper insights into their merits and shortcomings. As a result, theoretically motivated and provably improved MOEAs have been presented, e.g., in [Bian *et al.*, 2023; Doerr *et al.*, 2025a]. We aim at this as well, for the most prominent MOEA, the NSGA-II.

When analyzing the mathematical runtime analyses of the NSGA-II, we observe an ambivalent role of the population size N, the main parameter of this algorithm. On the one hand, the population size must be large enough to represent the Pareto front (and, in fact, a little larger, since the algorithm fails to perfectly efficiently cover the Pareto front) [Zheng and Doerr, 2023]. On the other hand, since the NSGA-II generates and evaluates N solutions per iteration, a large N slows down the algorithm. This effect was made precise in [Doerr and Qu, 2023b]. These two insights, together with the details of the proofs in these works, suggest to design an NSGA-II that gradually increases the population size.

Our contribution. We propose the *dynamic NSGA-II* (Algorithm 2), which is an NSGA-II that starts with a small population of size four and doubles it after a user-specified number τ of function evaluations, but not beyond a maximum population size $N_{\max}$. This second main parameter of the algorithm plays a similar role as the population size N of the classic NSGA-II, in particular, it has to be large enough for the algorithm to eventually cover the Pareto front. Since both empirical [Kukkonen and Deb, 2006] and mathematical [Zheng and Doerr, 2025] results have shown that it is preferable to select the next population of the NSGA-II sequentially, always based on the current crowding distance at that moment, we use this variant of the NSGA-II.

We analyze mathematically the performance of this dynamic NSGA-II on two standard benchmarks from the theory of MOEAs. For the OneMinMax benchmark (denoted here by OMM), we show that our algorithm with opti-

mal parameters with high probability finds the Pareto front within $O(n \log^2 n)$ function evaluations (Theorem 1). This is considerably faster than the $O(n^2 \log n)$ best runtime guarantees known for the standard versions of the NSGA-II, NSGA-III, SMS-EMOA, SPEA2, and MOEA/D with optimal parameters [Zheng and Doerr, 2023; Dang *et al.*, 2023; Zheng and Doerr, 2024; Ren *et al.*, 2024; Li *et al.*, 2016], proven to be asymptotically tight for the NSGA-II [Doerr and Qu, 2023b]. For the OneJumpZeroJump benchmark (OJZJ_k) with difficulty parameter $k \in [2..\lfloor \frac{n}{2} \rfloor]$, we prove that the dynamic NSGA-II with optimal parameters satisfies a runtime guarantee of $O(n^k \log^2 n)$ function evaluations with high probability (Theorem 4). Except for very large values of k, this compares again favorably with the best known runtime guarantee of $O((n-2k)n^k)$ for the NSGA-II, NSGA-III, SMS-EMOA, and SPEA2 [Doerr and Qu, 2023a; Opris, 2025; Bian *et al.*, 2023; Ren *et al.*, 2024], which is again tight for the classic NSGA-II [Doerr and Qu, 2023b]; we are not aware of any runtime analysis of the MOEA/D on OJZJ. Both of our runtime guarantees are independent of the maximum population size $N_{\max}$ as long as this is at least slightly larger than the size of the Pareto front.

Noting that the optimal choice of the doubling parameter τ is mostly determined by the part of the optimization process working with the initial population size, we also propose a variant of the dynamic NSGA-II that allows for a longer initial phase. For this *dynamic NSGA-II with longer initial phase*, we set the length of the initial phase in such a manner that it takes as long as all later phases combined until the maximum population size is reached. This significantly increases the time spent with a small population without increasing the asymptotic order of the runtime. In our runtime analysis, we show that this variant allows for a smaller doubling factor (for all but this initial phase), resulting in slightly superior runtimes (Theorems 5 and 6), e.g., an $O(n \log n)$ runtime on OMM, which is asymptotically optimal in a large class of unbiased black-box algorithms [Lehre and Witt, 2012].

Our results described so far stated the performance for an optimal choice of the doubling parameter τ. They display a linear dependence on τ when τ is at least of a certain size. Below this, the runtime essentially reverts to the one of the classic NSGA-II with population size $N_{\max}$. In that sense, a suboptimal choice of τ is not fatal, but the advantage of the dynamic choice of the population size could decrease or vanish. To relieve the user from manually optimizing τ, we propose a simple concurrent-run strategy (Algorithm 3) that removes the parameter τ from the algorithm at the price of a runtime that is by a logarithmic factor larger than the runtime obtainable with the optimal choice of τ (Theorem 7). This strategy works both for the dynamic NSGA-II and its variant with longer initial phase. Since the runtime of the former was independent of $N_{\max}$, we obtain here a version of the NSGA-II that has no parameters related to the population size.

Note. All proofs are available at [Doerr *et al.*, 2025b].

2 Related Work

Doubling schemes have been considered for single-objective evolutionary optimization, e.g., [Harik and Lobo, 1999; Doerr and Doerr, 2020]. To the best of our knowledge, this work is the first to study a MOEA with dynamic parameters via mathematical means, and our findings heavily rely on the multi-objective nature of the considered problems. In such, there is little directly related work to be discussed here.

What comes closest to our work are the recent works that study how to run a MOEA with a smaller population size with the help of an archive. An archive is a storage for good solution candidates that do not take part in the evolutionary process. This way, good solutions can be stored outside the main population of the MOEA, possibly allowing the MOEA to be run with a smaller population size. We refer to [Li *et al.*, 2024] for a recent general survey on this technique and review the two independent theoretical works on this topic.

[Bian *et al.*, 2024] consider the NSGA-II with an archive. They prove that this algorithm optimizes the OMM and LOTZ benchmark in $O(n \log n)$ and $O(n^2)$ function evaluations, respectively, when a constant population size is chosen. To achieve these runtimes, the analysis relies on an intensive use of one-point crossover. This is natural for a problem like LOTZ, but less natural for a problem like OMM, in which there is no particular order of the bits. We are very skeptical that the NSGA-II with archive and constant population size can optimize the OMM or OJZJ_k benchmarks without this type of crossover. All results proven in this work hold if each parent in each iteration has a constant chance to receive mutation only, allowing arbitrary use of crossover otherwise.

[Doerr *et al.*, 2024a] consider the MOEA/D, which by definition uses an archive. Their main result is that the mutation-only MOEA/D with a population size smaller than the Pareto front size has enormous difficulties finding the Pareto front of OMM when bit-wise mutation is used. However, when a heavy-tailed mutation operator (having a higher sampling variance) is used, smaller population sizes are admissible and lead to considerably better runtimes. More precisely, for $N = O(n^{\beta-1})$ the runtime becomes $O(n^\beta \log n)$, where the constant $\beta > 1$ is the exponent of the power-law distribution used to define the mutation operator. By choosing β small, runtimes of $O(n^{1+\varepsilon})$ can be obtained. Building on this work, [Corus *et al.*, 2025] showed that the MOEA/D with Fast Contiguous Somatic Hypermutation can optimize OMM in a runtime of order $O(n \log n)$. The MOEA/D with constant population size and one-point crossover was shown to optimize OMM in time $O(n \log n)$ in [Huang *et al.*, 2021].

From these results and our understanding of the proofs, we tend to believe that working with a growing population size is slightly preferable to an archive. Since the solutions in the archive can never be used to generate new solutions, there is a considerable risk (as proven in the bit-wise mutation setting regarded in [Doerr *et al.*, 2024b]) that the few solutions in the population are not sufficient to reach all solutions on the Pareto front. That said, a dynamic population size requires choosing the right growth schedule for the population.

3 Preliminaries

Let $\mathbb{N}$ denote the positive natural numbers, and $\mathbb{N}_0 := \{0\} \cup \mathbb{N}$. For $i, j \in \mathbb{N}_0$, let $[i..j] := [i, j] \cap \mathbb{N}_0$, $[i] := [1..i]$, and $[i]_0 := [0..i]$. We abbreviate $\log_2(x)$ to $\log(x)$ and $\log_e(x)$ to $\ln(x)$.

Algorithm	OMM	OJZJ$_k$ ($k \in [2..\lfloor \frac{n}{2} \rfloor]$)
static	$\Theta(Nn \log n)$ $N \geq 4(n+1)$	$\Theta(Nn^k)$ $N \geq 4(n-2k+3)$
dynamic	$O(\tau \log n)$ (Theorem 1) $\tau \geq Cn \ln(n)$	$O(\tau \log n)$ (Theorem 4) $\tau \geq Cn^k \ln(n)$
dynamic with longer initial phase	$O(\tau \log(N_{\max}) + N_{\max} \log n)$ (Theorem 5) $\tau \geq Cn$	$O(\tau \log N_{\max})$ (Theorem 6) $\tau \geq Cn^k$

Table 1: Number of function evaluations for the static NSGA-II (Algorithm 1), the dynamic NSGA-II (Algorithm 2), and the dynamic NSGA-II with longer initial phase until the Pareto front is covered. The results for the static NSGA-II on OMM are due to [Zheng and Doerr, 2023; Doerr and Qu, 2023b], and on OJZJ$_k$ due to [Doerr and Qu, 2023a; Doerr and Qu, 2023b], and they state an expected value. The other results hold with probability $1 - O(\frac{1}{n})$ and assume that $C \in \mathbb{R}_{>0}$ is a sufficiently large constant and that $N_{\max} \geq n + 4$.

We consider pseudo-Boolean *maximization*, that is, for a given $n \in \mathbb{N}$, we study functions $f\colon \{0,1\}^n \to \mathbb{R}^2$. We call each $x \in \{0,1\}^n$ an *individual* and $f(x) =: \big(f_1(x), f_2(x)\big)$ its *objective value*. Furthermore, for all $x \in \{0,1\}^n$, we denote the number of ones of x by $|x|_1$.

We compare objective values $u, v \in \mathbb{R}^2$ via the *domination* partial order $\succeq$. We say u *weakly dominates* v ($u \succeq v$) if $u_1 \geq v_1$ and $u_2 \geq v_2$; and u and v are *incomparable* if neither $u \succeq v$ nor $v \succeq u$. If and only if $u \succeq v$ and $u \neq v$, we say u *strictly dominates* v ($u \succ v$). We use domination also for individuals, referring to domination of their objective values.

For a function f, we call each $x \in \{0,1\}^n$ *Pareto-optimal* if and only if no $y \in \{0,1\}^n$ satisfies $f(y) \succ f(x)$. For each Pareto-optimal $x \in \{0,1\}^n$, we call $f(x)$ a *Pareto optimum*. The *Pareto front* is the set of all Pareto optima.

The OneMinMax Benchmark

We consider the OneMinMax (OMM) benchmark function [Giel and Lehre, 2010], which is the most commonly studied benchmark in the theory of MOEAs. OMM features two conflicting objectives: the first returns the number of zeros in a given individual, the second its number of ones. Formally, we have $\text{OMM}\colon \{0,1\}^n \to [n]_0^2$ with $x \mapsto (n - |x|_1, |x|_1)$.

Due to the conflicting objectives, all individuals are Pareto-optimal, and the Pareto front is $\{(i, n-i) \mid i \in [n]_0\}$.

The OneJumpZeroJump Benchmark

As a second benchmark function, we consider OneJumpZeroJump (OJZJ$_k$) [Doerr and Zheng, 2021], which is based on the commonly studied single-objective benchmark Jump [Droste *et al.*, 2002] benchmark. Similar to OMM, OJZJ$_k$ features two conflicting objectives that try to maximize the number of ones and zeros, respectively. However, given a hardness parameter $k \in [\lfloor \frac{n}{2} \rfloor]$, both objectives have low values for numbers of ones (zeros) which are at a distance of 1 to k to the maximum number n, requiring the population to jump across this valley of small objective values. Formally, we have $\text{OJZJ}_k\colon \{0,1\}^n \to [n+k]_0^2$ with $x \mapsto (\text{Jump}_k(x), \text{ZeroJump}_k(x))$, where

$$\text{Jump}_k(x) = \begin{cases} |x|_1 + k & \text{if } |x|_1 \leq n-k \text{ or } |x|_1 = n; \\ n - |x|_1 & \text{else;} \end{cases}$$

and ZeroJump$_k$ is like Jump$_k$, swapping ones and zeros.

Individuals in the valley of low objective values are not Pareto-optimal, and the Pareto front is $\{(n+k, 0), (0, n+k)\} \cup \{(i+k, n-i+k) \mid i \in \mathbb{N}, k \leq i \leq n-k\}$. We call the objective values $(n+k, 0)$ and $(0, n+k)$ the *outer* Pareto front, and the remaining Pareto front the *inner* Pareto front.

Throughout this work, when we mention OMM or OJZJ$_k$, we assume that the problem size $n \in \mathbb{N}$ is given. Our mathematical statements aim at asymptotics in terms of this n, implying that we may assume n to be sufficiently large.

Runtime

During the run of an MOEA, we assume that the objective value of an individual is evaluated exactly once—at the point of its creation. Hence, each potential copy of an individual is also evaluated exactly once. The *runtime* of an MOEA is the (random) number of function evaluations until the current multi-set of individuals that the MOEA maintains (its *population*) *covers* the Pareto front for the first time, that is, until the objective values of the population contain the Pareto front.

Empty Intervals

Given a population P of individuals not covering the Pareto front of the chosen benchmark, we are interested in the consecutive objective values of the Pareto front that are not in P. We call these intervals *empty*, defining them formally below. The classic NSGA-II that uses the current crowding distance (details in Section 4) has the useful property that empty intervals (for OMM) do not increase over time and even reduce quickly in time, if the population size is large enough and if the algorithm's parent population contains the all-zeros bit string (0^n) as well as the all-ones bit string (1^n). Note that the objective values of 0^n and 1^n are the two extreme points of the Pareto front of both OMM and OJZJ$_k$. Considering such populations is vital for our runtime analysis.

In the interest of clarity, we here only define empty intervals for OMM and defer the definition for OJZJ$_k$ to the respective section. In order to allow the comparison of empty intervals among different populations, we follow the definition of Zheng and Doerr [2025] and define a fixed number of n empty intervals for any population, based on the points $\{i - 0.5 \mid i \in [n]\}$ in the space $[0, n]$ of the first objective of OMM. Formally, let P be a population of individuals containing at least 0^n and 1^n. For all $i \in [n]$, the *i-th empty interval* I_i (of P for OMM) is defined as

$$I_i = [\max\{f_1(x) \mid x \in P \wedge \text{OMM}_1(x) \leq i - 0.5\}.. \\ \min\{f_1(x) \mid x \in P \wedge \text{OMM}_1(x) \geq i - 0.5\}].$$

Algorithm 1: The classic (static) NSGA-II [Deb *et al.*, 2002] with population size $N \in \mathbb{N}$ and selection based on current crowding distance [Zheng and Doerr, 2025], maximizing a given pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}^2$.

```
t := 0;
P_0 := draw N individuals uniformly at random;
while termination criterion not met do
    Q_t := generate offspring population with size N;
    P_{t+1} := select N individuals from P_t ∪ Q_t by
      non-dominated sorting followed by the current
      crowding distance (with uniform tie-breaker);
    t := t + 1;
```

By definition, the objective values not in P are given by the interior points of the empty intervals. For example, the empty interval $[0..2]$ implies that the objective value $(1, n-1)$ is not in the population. Different empty intervals can coincide.

Furthermore, given a population P of individuals containing at least 0^n and 1^n, we define the *maximum empty interval* (MEI) as the maximum length of its empty intervals, that is,

$$\mathrm{MEI}(P) = \max\{|I_i| \mid i \in [n]\},$$

where $|[a..b]| = b - a$ denotes the length of an interval $[a..b]$. Note that if P covers the Pareto front, then $\mathrm{MEI}(P) = 1$.

4 The Classic NSGA-II as a Baseline

The *non-dominated sorting genetic algorithm II* [Deb *et al.*, 2002] (NSGA-II, Algorithm 1) is the most popular and widely applied MOEA. As such, we use it as a baseline for our dynamic versions. However, we note that we use a slight improvement proposed by [Zheng and Doerr, 2025], which applies the *current crowding distance*, which handles ties in a manner that results in a more equidistant spread of solutions.

The NSGA-II maintains a multi-set of promising individuals (the *parent population*) of a given size N. The initial population is generated uniformly at random and updated iteratively. In each iteration, the NSGA-II creates an *offspring population* of size N by user-defined means and then selects from the combined parent and offspring population the N most promising individuals for the next iteration. This selection mechanism ranks individuals hierarchically: first, individuals are selected by their rank assigned via non-dominated sorting. Ties are handled via the (current) crowding distance, and any remaining ties are handled uniformly at random.

Computing the offspring population. The NSGA-II creates in total N offspring by repeatedly picking a certain number of individuals from the parent population and then creating modifications of them. If it picks a single parent, the process of creating an offspring is called *mutation*.

We allow for a variety of ways to select parents and to create offspring. The only restriction we make is that there is a constant $q \in (0, 1]$ such that for each possible parent population P and each parent $x \in P$, the probability to select x at least once during the computation of the offspring population and to apply *bit-wise mutation* to x is at least q.

Given a parent $x \in \{0,1\}^n$, bit-wise mutation creates an offspring $y \in \{0,1\}^n$ by first copying x and then flipping each bit of y independently with probability $\frac{1}{n}$.

Non-dominated sorting. Non-dominated sorting assigns each individual x in a given (combined) population R a *rank* that determines by how many individuals x is strictly dominated. Individuals with the same rank are grouped into the same *non-dominated front*, resulting in a partition $(F_i)_{i\in[0..k-1]}$ of R (of some size k)

The NSGA-II always attempts to add entire fronts in increasing order to its next parent population, as the individuals per front are incomparable. Once at least N individuals are chosen, the algorithm does not add any further fronts. Formally, $i^* = \min\{i \in [0..k-1] \mid \sum_{j\in[0..i]} F_i \geq N\}$ is the *critical rank*. The fronts up to $i^* - 1$ are always part of the next parent population. If adding F_{i^*} results exactly in a population size of N, the selection process for the next parent population terminates. Otherwise, the NSGA-II selects the best $N - |\bigcup_{j\in[0..i^*-1]} F_i|$ individuals from F_{i^*} based on their *crowding distance*, where higher values are preferable.

Crowding distance. The crowding distance of each individual $x \in F_{i^*}$ is the sum of the crowding distance of x *per objective*. For each objective $j \in \{1, 2\}$, let $(y_k)_{k\in[|F_{i^*}|]}$ denote the individuals of F_{i^*} sorted in increasing order with respect to objective j. If x is in the first or last position, its crowding distance for objective j is positive infinity. Otherwise, its crowding distance for objective j is the normalized distance in objective value to its direct neighbors, that is, it is $\big(f_j(y_{k^*+1}) - f_j(y_{k^*-1})\big)/\big(f_j(y_{|F_{i^*}|}) - f_j(y_1)\big)$.

The *current* crowding distance [Zheng and Doerr, 2025] is a modification of the classic crowding distance that aims at decreasing the maximum distances of the resulting population. The current crowding distance achieves this by iteratively removing an individual x with the smallest (classic) crowding distance and then recomputing all (classic) crowding distances[1], until the desired population size is achieved.

5 The Dynamic NSGA-II

Based on the classic, static NSGA-II (Algorithm 1), we propose the *dynamic NSGA-II* (Algorithm 2), which starts with an initial population of size 4 and attempts to double it periodically after a user-defined number $\tau \in \mathbb{N}$ of function evaluations. An attempt is successful if and only if the current population size is less than the user-defined maximum population size $N_{\max} \in \mathbb{N}$. We call the consecutive function evaluations between attempts *phases*, noting that phase 0 starts at the beginning of the algorithm run and ends at the end of the iteration in which the population size is attempted to be doubled for the first time. Moreover, note that the number of phases is unbounded although the number of doublings is not.

It is important that we attempt doublings after τ function evaluations and not iterations. If we double the population size after a fixed number of iterations, then, since the work done per iteration is proportional to the population size, relatively little work is done with small population sizes. This

[1]Note that x has at most four neighbors in its objectives.

Algorithm 2: The dynamic NSGA-II with maximum population size $N_{\max} \in \mathbb{N}$ and doubling parameter $\tau \in \mathbb{N}$, maximizing a given pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}^2$. Our only assumption on the offspring generation is that each parent has a probability of at least a constant $q \in (0,1]$ to be selected for generating an offspring via bit-wise mutation.

$t \coloneqq 0$, $N \coloneqq 4$, $w \coloneqq 0$;
$P_0 \coloneqq$ draw N individuals uniformly at random;
while termination criterion not met **do**
  $Q_t \coloneqq$ generate offspring population with size N;
  $w \coloneqq w + N$;
  if $w \geq \tau$ **then** $w \coloneqq 0$, $N \coloneqq \max\{2N, N_{\max}\}$;
  $P_{t+1} \coloneqq$ select N individuals from $P_t \cup Q_t$ by non-dominated sorting followed by the current crowding distance (with uniform tie-breaker);
  $t \coloneqq t + 1$;

risks losing any advantage from a dynamic value for the population size. By counting function evaluations, we ensure that each phase receives roughly the same computational budget.

Performance Guarantees for OMM We start by proving the following bound for the OMM benchmark. For the optimal value of τ, our runtime guarantee is $O(n \log^2 n)$, which is considerably faster than the $\Theta(n^2 \log n)$ known for the classic NSGA-II with optimal parameters [Zheng and Doerr, 2023; Doerr and Qu, 2023b]. We note that all slower doubling schedules with $\tau = o(n^2)$ still lead to an asymptotic improvement over the static NSGA-II. Likewise, a too small value of τ is not detrimental, but reverts to the $O(N_{\max} n \log n)$ performance of the classic NSGA-II.

Theorem 1. *Consider the dynamic NSGA-II (Algorithm 2) optimizing* OMM *with $N_{\max} \geq n+4$. Let T denote the number of iterations until the population covers the Pareto front, and let F denote the number of function evaluations in the first T iterations. If $\tau \geq 64en\ln(n)/q$, then with probability at least $1-O(1/n)$, we have $T = O(\tau)$ and $F = O(\tau \log n)$.*

For smaller values of τ, we have $F = O(N_{\max} n \log n)$ both in expectation and with probability $1-O(1/n)$, the same for the static NSGA-II with population size $N_{\max}$.

We note that we did not optimize the constants in Theorem 1. Remarkably, for sufficiently large τ, both the bounds on T and F do not depend on $N_{\max}$, as the dynamic NSGA-II covers the entire Pareto front with high probability once its population size is in the order of the size of the Pareto front, that is, order n. This makes the dynamic NSGA-II very robust in terms of choosing $N_{\max}$, in particular, since a population size of $n+1$ is required in order to cover the entire Pareto front, and our lower bound of $n+4$ almost matches this required minimum. Moreover, since we cover all possible cases for τ, we see that the dynamic NSGA-II is also very robust in terms of choosing the doubling parameter, being better or as good as the static NSGA-II for any $\tau = O(n^2)$.

Moreover, we note that the arguments for small τ in Theorem 1 also apply to the failure case with probability $O(1/n)$. Hence if the dynamic NSGA-II fails to display the desired behavior (which happens with probability $O(1/n)$), then the runtime reverts to the classic $O(N_{\max} n \log n)$ number of function evaluations, both in expectation and with probability $1 - O(1/n)$. We note further that by raising the constant in the requirement on τ, all failure probabilities can be reduced to n^{-c} for any desired constant c. We omit the details.

We prove Theorem 1 in two parts. First, we show that, with high probability, the dynamic NSGA-II finds the extreme solutions 0^n and 1^n during phase 0. Afterward, we show that the MEI roughly halves, with high probability, in each phase.

For the first part, we apply a result by [Zheng and Doerr, 2025], which holds in expectation, whereas we require one that holds with high probability. We obtain such a result by following essentially the analysis in [Zheng and Doerr, 2025] but adding a tail bound for harmonic sums of geometric random variables from [Doerr, 2020, Theorem 1.10.35]. Since the proof holds also for the NSGA-II with classic crowding distance and for arbitrary population sizes (of at least four), we formulate the result in this general way.

Lemma 2. *Consider the NSGA-II (Algorithm 1) with arbitrary initialization, with survival based on the classic or current crowding distance, and with arbitrary population size $N \geq 4$. Assume that in each iteration, each parent with probability at least $q \in (0,1]$ is chosen to generate an offspring via bit-wise mutation. Let T denote the number of iterations until in this algorithm, when optimizing* OMM*, the population contains 1^n and 0^n. Then $T \leq 2en\ln(n)/q$ with probability at least $1 - \frac{2}{n}$.*

For the second part, we rely on the following lemma showing that the MEI quickly reduces once 0^n and 1^n are found. This result and its proof are similar to [Zheng and Doerr, 2025, Lemmas 11 and 12], except that we need a more detailed result and a more precise tail bound, since we apply this statement for each phase separately (and thus also for phases of few iterations, where the strong concentration behavior is weaker).

Lemma 3. *Consider a run of the NSGA-II (Algorithm 1), generating the next population in any way that has the property that any parent has a probability of at least $q \in (0,1]$ to be selected a parent of a bit-wise mutation, and using the current crowding distance for the selection of the next population. Consider some iteration $t \in \mathbb{N}_0$ of this algorithm optimizing* OMM*. Let P_t be arbitrary except that $\{0^n, 1^n\} \subseteq P_t$. Let $m, m' \in \mathbb{N}$ such that $\max\{\frac{2n}{N-3}, 1\} \leq m'$ and $\mathrm{MEI}(P_t) \leq m$. Let $\delta \in \mathbb{N}$ such that $\delta \geq 4e(m-m')/q$. Then $\Pr[\mathrm{MEI}(P_{t+\delta}) \leq m'] \geq 1 - n\exp(-\frac{\delta q}{16e})$.*

Performance Guarantees for OJZJ_k

Theorem 4. *Consider the dynamic NSGA-II (Algorithm 2) optimizing OJZJ_k with $k \in [2..\lfloor \frac{n}{2} \rfloor]$ and $N_{\max} \geq n+4$. Let T denote the number of iterations until the population covers the Pareto front, and let F denote the number of function evaluations in the first T iterations. If $\tau \geq 4e(n^k + 8n^2)\ln(n)/q$, then with probability at least $1 - O(1/n)$, we have $T = O(\tau)$ and $F = O(\tau \log n)$.*

For smaller values of τ, we have $F = O(N_{\max} n^k \log n)$ with probability $1 - O(1/n)$ and $E[F] = O(N_{\max} n^k)$, both the same as for the static NSGA-II with population size $N_{\max}$.

For an optimal choice of τ, the resulting runtime guarantee of $O(n^k \log^2 n)$ is considerably faster than the $\Theta(n^{k+1})$ bound for the static NSGA-II [Doerr and Qu, 2023a; Doerr and Qu, 2023b], and it remains faster for any such choice with $\tau = o(n^{k+1}/\log n)$. Moreover, a too small choice of τ just results in the same bound as for the static NSGA-II.

We note that, in comparison to our result for OMM (Theorem 1), our speed-up is slower by a factor of $O(\log n)$, which is a consequence of our results holding with high probability. The final steps in the analysis of OJZJ_k usually wait for a mutation to occur that flips exactly k bits. In expectation, this takes $O(n^k)$ iterations (with each iteration requiring $N_{\max}$ function evaluations). However, since this event follows a geometric distribution, it is not strongly concentrated around its expectation and requires and additional factor of $O(\log n)$ in order to hold with high probability.

The details of our proof for Theorem 4 are in the supplementary material. However, the outline of our proof strategy is very similar to that for OMM. The main difference is that we first wait until we find the optima of the outer Pareto front, which requires in either direction a mutation that flips exactly k bits at once. This happens with high probability for the stated choice of τ. During the next phase, we wait to reach the outermost optima of the *inner* Pareto front. This requires for either direction a mutation that flips at least k *arbitrary* bits, which is easier than the first phase. From then on, the problem essentially boils down to OMM.

6 Using a Longer Initial Phase

As the proofs of the results in the previous section show, the initial phase of the dynamic NSGA-II is crucial as here the extreme points of the Pareto front are found. To accommodate for this, we propose a dynamic NSGA-II with longer initial phase. For all but the initial phase, it works as the dynamic NSGA-II (Algorithm 2), doubling its population size after a fixed budget of $\tau \in \mathbb{N}$ fitness evaluations. For the initial phase, we use a budget that is roughly equal to the budget of all following phases (until the maximum population size $N_{\max}$ is reached), namely $\lceil \log(N_{\max}/4) \rceil \tau$ function evaluations. By this, the dynamic NSGA-II with longer initial phase spends more time in the very efficient phase with population size four, but this without increasing the asymptotic runtime. As we shall see, this allows to use smaller values for τ, leading to better runtimes.

We prove runtime guarantees for the dynamic NSGA-II with longer initial phase on the OMM and OJZJ_k benchmarks in Theorems 5 and 6, respectively. For optimal parameter values, these runtime guarantees are better by a factor of $O(n)$ compared to the best known bounds for static NSGA-II, in particular, they are better by a factor of $O(\log n)$ than the guarantees for optimal parameters for the dynamic NSGA-II. However, the runtime now always depends on $N_{\max}$, due to the choice of the budget for phase 0.

Performance Guarantees for OMM For optimal parameters, we show a runtime of $O(n \log n)$ function evaluations, which is considerably faster than the $\Theta(n^2 \log n)$ runtime guarantee for the static NSGA-II. We note that no unbiased mutation-only black-box algorithm can find the all-ones string faster than in $\Theta(n \log n)$ time [Lehre and Witt, 2012], so it is hard to imagine that a general-purpose heuristic can solve the OMM problem asymptotically faster than this.

Theorem 5. *Consider the dynamic NSGA-II with longer initial phase optimizing* OMM *with $N_{\max} \geq n + 4$. Let T denote the number of iterations until the population covers the Pareto front, and let F denote the number of function evaluations in the first T iterations. If $\tau \geq 16en/q$, then with probability at least $1 - O(1/n)$, we have*

$$T = O(\tau \log N_{\max}) \text{ and } F = O(\tau \log N_{\max} + N_{\max} \log n).$$

For smaller values of τ, we have $F = O(N_{\max} n \log n)$ both in expectation and with probability $1 - O(1/n)$, the same for the static NSGA-II (Algorithm 1) with population size $N_{\max}$.

The proof of Theorem 5 is similar to the one of Theorem 1, noting that the budget for phase 0 is of the same order. Afterward, the budget for the dynamic NSGA-II with longer initial phase is slightly smaller, which is why we apply Lemma 3 only until the MEI is $O(\log n)$. We then wait until the algorithm attains a population size linear in n and show via Lemma 3 that the algorithm finds all remaining Pareto optima within the next $O(\log n)$ phases.

Performance Guarantees for OJZJ_k For optimal parameters, we show a runtime of $O(n^k \log n)$. For small values of k, this is close to the general $\Omega(n^k)$ lower bound when relying solely on bit-wise mutation to flip exactly k bits.

Theorem 6. *Consider the dynamic NSGA-II with longer initial phase optimizing OJZJ_k with $k \in [2..\lfloor \frac{n}{2} \rfloor]$ and $N_{\max} \geq n + 4$. Let T denote the number of iterations until the population covers the Pareto front, and let F denote the number of function evaluations in the first T iterations. If $\tau \geq 4e(2n^k + 8n^2)/q$, then with probability at least $1 - O(1/n)$, we have*

$$T = O(\tau \log N_{\max}) \quad \text{and} \quad F = O(\tau \log N_{\max}).$$

For smaller values of τ, we have $F = O(N_{\max} n^k \log n)$ with probability $1 - O(1/n)$ and $E[F] = O(N_{\max} n^k)$, both the same as for the static NSGA-II (Algorithm 1) with population size $N_{\max}$.

The proof strategy for Theorem 5 is similar to the one for Theorem 4. As the first phase is now longer by a factor of $\Theta(\log N_{\max}) \geq \Theta(\log n)$, a value of $\tau = \Theta(n^k)$ is sufficient to find with high probability the outer Pareto front. Afterward, finding the outermost Pareto optima on the inner front may have a smaller budget than in the case with the dynamic NSGA-II, but since it is easier to go from the outer front to the inner front than vice versa, the slightly reduced budget does not affect the time to find the outermost solutions on the inner front. Last, we deal essentially with an OMM instance.

7 Choosing the Phase Length Automatically

To relieve users from manually select a suiting value for the phase length τ of the dynamic NSGA-II and the dynamic NSGA-II with longer initial phase, we propose a framework (Algorithm 3) that concurrently runs multiple instances of the dynamic variant with distinct values for τ, namely all powers of

Algorithm 3: Concurrent optimization for the dynamic NSGA-II and the dynamic NSGA-II with longer initial phase, called $A^\star$, given parameter $N_{\max} \in \mathbb{N}_{>4}$. Read Algorithm 3 as an initialization for whenever these values are used for the first time, not as infinite loop. Recalling Eqs. (1) and (2), Algorithm 3 searches with increasing $j \in \mathbb{N}$ and stops once 2^j is larger than all preceding values.

```
for i ∈ ℕ_{≥2} do r_i := 0;    φ_i := 0;
while no instance meets termination criterion do
    i := arg min_{j∈ℕ}{r_j + Phase(N_max, j, φ_j)};
    if φ_i = 0 then initialize an A* instance A_i
      with N_max as in the input and with τ = 2^i;
    Run the next phase of A_i;
    φ_i := φ_i + 1;
    r_i := r_i + Phase(N_max, i, φ_i);
```

two. For each such $\tau = 2^i$, we consider an instance A_i of the dynamic variant. The framework repeatedly picks an instance A_i and runs its next phase. If an instance was picked before, we continue with the process as we last left it. Each time, we pick an instance A_i such that the total number of function evaluations spend on A_i *after* the potential next phase is minimal among all instances, breaking ties arbitrarily.

Formally, let $\mathrm{Phase}(N_{\max}, i, \phi)$ denote the number of function evaluations in phase ϕ of the dynamic NSGA-II variant with parameters $N_{\max}$ and $\tau = 2^i$. Then, at the beginning of each iteration of the **while**-loop, r_i equals the number of function evaluations that instance A_i received. Formally, for the dynamic NSGA-II and $N_\phi = \min\{4 \cdot 2^\phi, N_{\max}\}$, we have

$$\mathrm{Phase}(N_{\max}, i, \phi) = \lceil \tfrac{2^i}{N_\phi} \rceil \cdot N_\phi, \tag{1}$$

since we perform exactly N_ϕ function evaluations per iteration of A_i and we only stop once we reach or surpass the total budget of $\tau = 2^i$ function evaluations. For the dynamic NSGA-II with longer initial phase, Phase is the same, except

$$\mathrm{Phase}(N_{\max}, i, 0) = \lceil \log(N_{\max}/4) \rceil \cdot 2^i. \tag{2}$$

Our main runtime result for Algorithm 3 is Theorem 7, which shows essentially that on both OMM and OJZJ_k, both dynamic NSGA-II variants are slower by only a factor of $O(\log n)$ and $O(k \log n)$, respectively, when compared to an optimal value of τ in Theorems 1 and 4 to 6. This small slow-down still results in runtimes that are far faster than the typical $O(n^2 \log n)$ and $O(n^{k+1})$ runtimes of many MOEAs on OMM and OJZJ_k, respectively, as discussed before.

Theorem 7. *Consider Algorithm 3 optimizing* OMM *or* OJZJ_k *with* $k \in [2..\lfloor \frac{n}{2} \rfloor]$ *and* $N_{\max} \geq n + 4$. *Let* F_{OMM} *and* F_{OJZJ_k} *denote the total number of function evaluations until one of the instances covers the Pareto front, respectively.*

If the instances run the dynamic NSGA-II (Algorithm 2),

$$E[F_{\mathrm{OMM}}] = O(n \log^3 n) \text{ and } E[F_{\mathrm{OJZJ}_k}] \quad = O(kn^k \log^3 n).$$

If they run the dynamic NSGA-II with longer initial phase,

$$E[F_{\mathrm{OMM}}] = O(n \log(N_{\max}) \log(n \log N_{\max})),$$
$$E[F_{\mathrm{OJZJ}_k}] = O(kn^k \log(N_{\max}) \log(n \log N_{\max})).$$

These four bounds also hold with probability $1 - O(1/n)$.

In contrast to our results in the previous sections, the bounds in Theorem 7 hold additionally in expectation, as multiple runs start with a small population size and thus result in independent trials of covering the Pareto front once the population is sufficiently large. Moreover, as in the case for the dynamic NSGA-II, Algorithm 3 with the dynamic NSGA-II also effectively eliminates $N_{\max}$ as a parameter, as its runtime is independent of $N_{\max}$, once sufficiently large. Hence, choosing arbitrarily large values for $N_{\max}$ results in an essentially parameterless algorithm, which is a remarkable result, in particular since this parameterless version still exhibits a very fast runtime on OMM and OJZJ_k.

Theoretical analysis. The balancing of the function evaluations across the instances in Algorithm 3 allows us to bound the total number of function evaluations from above by a function of the number of evaluations performed within any given instance A_i, as captured in the following lemma.

Lemma 8. *Let* $i \in \mathbb{N}_{\geq 2}$ *and* $r \in \mathbb{N}$. *Consider some point in the execution of Algorithm 3 on any objective function with* $N_{\max} \in \mathbb{N}_{>4}$ *such that at least phase 0 of* A_i *has been conducted and* A_i *received at most* r *function evaluations. Then the total number of function evaluations in Algorithm 3 so far is at most* $2r \log(2r)$.

The proof of Lemma 8 uses that in order to start a new instance $i \in \mathbb{N}_{\geq 2}$, its run time budget is larger than the cost of running any of the roughly $\log i$ smaller instances.

The proof of Theorem 7 uses bounds from Theorems 1 and 4 to 6 for optimal choices of τ. Then, Lemma 8 bounds the number of function evaluations in all other instances. The expected runtimes in Theorem 7 follow since the runtimes in Theorem 1 hold with high probability for sufficiently large τ.

8 Conclusion

We introduced the *dynamic NSGA-II* (Algorithm 2), which runs the classic NSGA-II with a periodically increasing population size, based on a periodicity τ determined by the user. We proved that the dynamic NSGA-II has a superior runtime performance in comparison to typical MOEAs for the OMM (Theorem 1) and OJZJ_k (Theorem 4) benchmark for a large range of τ. For optimal parameter choices, the dynamic NSGA-II even obtains speed-ups of $\widetilde{O}(n)$.

Moreover, we showed that if the initial phase of the dynamic NSGA-II is slightly longer than the other phases, the optimal performance improves by a factor of $O(\log n)$, both on OMM (Theorem 5) and OJZJ_k (Theorem 6).

Last, in order to remove the need to choose the doubling parameter τ adequately, we introduced a scheme that runs the dynamic NSGA-II variants concurrently (Algorithm 3). We showed that this scheme increases the total runtime performance by only a logarithmic factor of the original runtime (Theorem 7), while eliminating the parameter τ.

For future research, it is interesting to see whether the dynamic NSGA-II maintains its advantage in other popular scenarios, such as LOTZ [Laumanns *et al.*, 2004].

Acknowledgments

Simon Wietheger acknowledges support by the Austrian Science Fund (FWF) [10.55776/Y1329]. Martin S. Krejca is supported by the French National Agency for Research (ANR) via the JCJC grant titled *MultiEDA* (ANR-25-CE23-2418-01). The research of Benjamin Doerr and Martin S. Krejca benefited from the support of the FMJH Program PGMO.

References

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

[Bian *et al.*, 2023] Chao Bian, Yawen Zhou, Miqing Li, and Chao Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5513–5521. ijcai.org, 2023.

[Bian *et al.*, 2024] Chao Bian, Shengjie Ren, Miqing Li, and Chao Qian. An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 6905–6913. ijcai.org, 2024.

[Coello *et al.*, 2007] Carlos Artemio Coello Coello, Gary B. Lamont, and David A. van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edition, 2007.

[Corus *et al.*, 2025] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. Fast contiguous somatic hypermutations for single-objective optimisation and multi-objective optimisation via decomposition. In *Conference on Artificial Intelligence, AAAI 2025*, pages 26922–26930, 2025.

[Dang *et al.*, 2023] Duc-Cuong Dang, Andre Opris, Bahare Salehi, and Dirk Sudholt. Analysing the robustness of NSGA-II under noise. In *Genetic and Evolutionary Computation Conference, GECCO 2023*, pages 642–651. ACM, 2023.

[Deb and Jain, 2014] Kalyanmoy Deb and Himanshu 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*, 18:577–601, 2014.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[Doerr and Doerr, 2020] Benjamin Doerr and Carola Doerr. Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 271–321. Springer, 2020. Also available at https://arxiv.org/abs/1804.05650.

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

[Doerr and Qu, 2023b] Benjamin Doerr and Zhongdi Qu. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Conference on Artificial Intelligence, AAAI 2023*, pages 12408–12416. AAAI Press, 2023.

[Doerr and Zheng, 2021] Benjamin Doerr and Weijie Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, pages 12293–12301. AAAI Press, 2021.

[Doerr *et al.*, 2024a] Benjamin Doerr, Joshua Knowles, Aneta Neumann, and Frank Neumann. A block-coordinate descent EMO algorithm: theoretical and empirical analysis. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, pages 493–501. ACM, 2024.

[Doerr *et al.*, 2024b] Benjamin Doerr, Martin S. Krejca, and Noé Weeks. Proven runtime guarantees for how the MOEA/D computes the Pareto front from the subproblem solutions. In *Parallel Problem Solving from Nature, PPSN 2024, Part III*, pages 197–212. Springer, 2024.

[Doerr *et al.*, 2025a] Benjamin Doerr, Tudor Ivan, and Martin S. Krejca. Speeding up the NSGA-II with a simple tie-breaking rule. In *Conference on Artificial Intelligence, AAAI 2025*, pages 26964–26972. AAAI Press, 2025.

[Doerr *et al.*, 2025b] Benjamin Doerr, Martin S. Krejca, and Simon Wietheger. Speeding up the NSGA-II via dynamic population sizes. *CoRR*, abs/2509.01739, 2025.

[Doerr, 2020] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 1–87. Springer, 2020. Also available at https://arxiv.org/abs/1801.06733.

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

[Giel and Lehre, 2010] Oliver Giel and Per Kristian Lehre. On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation*, 18:335–356, 2010.

[Harik and Lobo, 1999] Georges R. Harik and Fernando G. Lobo. A parameter-less genetic algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 1999*, pages 258–265. Morgan Kaufmann, 1999.

[Huang *et al.*, 2021] Zhengxin Huang, Yuren Zhou, Zefeng Chen, Xiaoyu He, Xinsheng Lai, and Xiaoyun Xia. Running time analysis of MOEA/D on pseudo-Boolean functions. *IEEE Transactions on Cybernetics*, 51:5130–5141, 2021.

[Kukkonen and Deb, 2006] Saku Kukkonen and Kalyanmoy Deb. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *Conference on Evolutionary Computation, CEC 2006*, pages 1179–1186. IEEE, 2006.

[Laumanns *et al.*, 2004] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8:170–182, 2004.

[Lehre and Witt, 2012] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[Li *et al.*, 2016] Yuan-Long Li, Yu-Ren Zhou, Zhi-Hui Zhan, and Jun Zhang. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20:563–576, 2016.

[Li *et al.*, 2024] Miqing Li, Manuel López-Ibáñez, and Xin Yao. Multi-objective archiving. *IEEE Transactions on Evolutionary Computation*, 28:696–717, 2024.

[Opris, 2025] Andre Opris. A first runtime analysis of NSGA-III on a many-objective multimodal problem: provable exponential speedup via stochastic population update. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*, pages 8903–8911. ijcai.org, 2025.

[Ren *et al.*, 2024] Shengjie Ren, Chao Bian, Miqing Li, and Chao Qian. A first running time analysis of the Strength Pareto Evolutionary Algorithm 2 (SPEA2). In *Parallel Problem Solving from Nature, PPSN 2024, Part III*, pages 295–312. Springer, 2024.

[Wietheger and Doerr, 2023] Simon Wietheger and Benjamin Doerr. A mathematical runtime analysis of the non-dominated sorting genetic algorithm III (NSGA-III). In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5657–5665. ijcai.org, 2023.

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

[Zheng and Doerr, 2023] Weijie Zheng and Benjamin Doerr. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *Artificial Intelligence*, 325:104016, 2023.

[Zheng and Doerr, 2024] Weijie Zheng and Benjamin Doerr. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*, pages 20874–20882. AAAI Press, 2024.

[Zheng and Doerr, 2025] Weijie Zheng and Benjamin Doerr. Approximation guarantees for the non-dominated sorting genetic algorithm II (NSGA-II). *IEEE Transactions on Evolutionary Computation*, 29:891–905, 2025.

[Zheng *et al.*, 2022] Weijie Zheng, Yufei Liu, and Benjamin Doerr. A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II). In *Conference on Artificial Intelligence, AAAI 2022*, pages 10408–10416. AAAI Press, 2022.

[Zhou *et al.*, 2011] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32–49, 2011.

[Zitzler *et al.*, 2001] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK report*, 103, 2001.