Speeding Up the NSGA-II with a Simple Tie-Breaking Rule

Benjamin Doerr[1], **Tudor Ivan**[2], **Martin S. Krejca**[1]

[1]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris
[2]École Polytechnique, Institut Polytechnique de Paris
{first-name.last-name}@polytechnique.edu

Abstract

The non-dominated sorting genetic algorithm II (NSGA-II) is the most popular multi-objective optimization heuristic. Recent mathematical runtime analyses have detected two shortcomings in discrete search spaces, namely, that the NSGA-II has difficulties with more than two objectives and that it is very sensitive to the choice of the population size. To overcome these difficulties, we analyze a simple tie-breaking rule in the selection of the next population. Similar rules have been proposed before, but have found only little acceptance. We prove the effectiveness of our tie-breaking rule via mathematical runtime analyses on the classic ONEMINMAX, LEADINGONESTRAILINGZEROS, and ONEJUMPZEROJUMP benchmarks. We prove that this modified NSGA-II can optimize the three benchmarks efficiently also for many objectives, in contrast to the exponential lower runtime bound previously shown for ONEMINMAX with three or more objectives. For the bi-objective problems, we show runtime guarantees that do not increase when moderately increasing the population size over the minimum admissible size. For example, for the ONEJUMPZEROJUMP problem with representation length n and gap parameter k, we show a runtime guarantee of $O(\max\{n^{k+1}, Nn\})$ function evaluations when the population size is at least four times the size of the Pareto front. For population sizes larger than the minimal choice $N = \Theta(n)$, this result improves considerably over the $\Theta(Nn^k)$ runtime of the classic NSGA-II.

Introduction

Many real-world optimization tasks face several, usually conflicting objectives. One of the most successful approaches to such *multi-objective* optimization problems are evolutionary algorithms (EAs) (Coello, Lamont, and van Veldhuizen 2007; Zhou et al. 2011), with the NSGA-II (Deb et al. 2002) standing out as the by far dominant algorithm in practice (over 50 000 citations on Google Scholar).

Recent mathematical works have shown many positive results for the NSGA-II (see *Previous Works*) but have also exhibited two difficulties. (1) For more than two objectives, the crowding distance as selection criterion seems to have some shortcomings. This was formally proven for the optimization of the simple ONEMINMAX benchmark, where an exponential lower bound on the runtime was shown for three and more objectives when the population size is linear in the size of the Pareto front (Zheng and Doerr 2024b). The proof of this result suggests that similar problems exist for many optimization problems with three or more objectives. (2) All runtime guarantees proven for the NSGA-II increase linearly with the population size. For some settings, even matching lower bounds proving this effect were proven (Doerr and Qu 2023b). This behavior is very different from many single-objective EAs, where a moderate increase of the population size increases the cost of an iteration but at the same time reduces the number of iterations in a way that the total runtime (number of function evaluations) is at most little affected.

The reason for this undesired behavior of the NSGA-II, as the proofs by Doerr and Qu (2023b) reveal, is that the population of the NSGA-II is typically not evenly distributed on the known part of the Pareto front. Instead, the distribution is heavily skewed toward the inner region of the Pareto front.

Inspired by this observation, we propose to add a simple tie-breaking criterion to the selection of the next population. In the classic NSGA-II, selection is done according to non-dominated sorting, ties are broken according to the crowding distance, and remaining ties are broken randomly. We add the number of individuals having a certain objective value as the third criterion, and break only the remaining ties randomly. We note that a similar idea was suggested already by Fortin and Parizeau (2013) and supported by empirical results, but has not made it into the typical use of the NSGA-II.

Our tie-breaker solves both problems. We rigorously study the runtime for the three most established benchmarks in the theory of multi-objective EAs (MOEAs): ONEMINMAX, LEADINGONESTRAILINGZEROS, and ONEJUMPZEROJUMP. For all defined versions with at least three objectives and for constant gap parameter k, we show that our NSGA-II with a population size exceeding the Pareto front size only by a lower-order term efficiently solves the problem when the Pareto front size is polynomial (Theorem 3).

For the bi-objective versions of these problems, we prove runtime guarantees showing that for a certain range of the population size, the runtime is (asymptotically) not affected by this parameter. The size of this range depends on the difficulty of the problem. For the difficult ONEJUMPZEROJUMP benchmark with problem size n and gap parameter k, our runtime guarantee is $O(n^{k+1})$ for all population sizes N between $4(n - 2k + 3)$ and $O(n^k)$, and it is $O(Nn)$ for

$N = \Omega(n^k)$. Compared to the runtime of $\Theta(Nn^k)$ proven by Doerr and Qu (2023a), this is a noteworthy speed-up for larger population sizes. From a practical perspective, this result indicates that our tie-breaking rule significantly reduces the need for a careful optimization of the algorithm parameter N. We support this claim empirically, showing that this speed-up is already noticeable when the chosen and optimal population size deviate only by a constant factor.

Overall, this work shows that adding the simple tie-breaker of preferring individuals with rarer objective values can lead to considerable performance gains and greatly reduce the need of determining an optimal population size.

All proofs and some additional details are in the full version (Doerr, Ivan, and Krejca 2024a).

Previous Works

This being a theoretical work, for space reasons, we refer to the surveys by Coello et al. (2007) and by Zhou et al. (2011) for the success of MOEAs in practical applications.

The mathematical analysis of randomized search heuristics has supported for a long time the development of these algorithms (Neumann and Witt 2010; Auger and Doerr 2011; Jansen 2013; Zhou, Yu, and Qian 2019; Doerr and Neumann 2020), including MOEAs. The runtime analysis for MOEAs was started in (Laumanns et al. 2002; Giel 2003; Thierens 2003) with very simplistic algorithms like the *simple evolutionary multi-objective optimizer (SEMO)* or the *global SEMO*. Due to the complex population dynamics, it took many years until more prominent MOEAs could be analyzed, such as the $(\mu + 1)$ SIBEA (Brockhoff, Friedrich, and Neumann 2008), MOEA/D (Li et al. 2016), NSGA-II (Zheng, Liu, and Doerr 2022), NSGA-III (Wietheger and Doerr 2023), or SMS-EMOA (Bian et al. 2023).

The analysis of the NSGA-II, the by far dominant algorithm in practice, had a significant impact on the field and was quickly followed up by other runtime analyses for this algorithm. The vast majority of these prove runtime guarantees for bi-objective problems. Some are comparable to those previously shown for the (G)SEMO (Bian and Qian 2022; Doerr and Qu 2023a; Dang et al. 2023b; Cerf et al. 2023; Deng et al. 2024), others explore new phenomena like approximation properties (Zheng and Doerr 2024a), better robustness to noise (Dang et al. 2023a), or new ways how crossover (Doerr and Qu 2023c) or archives (Bian et al. 2024) can be advantageous. All these works, except for (Bian et al. 2024), require that the population size is at least a constant factor larger than the size of the Pareto front. Increasing the population size further leads to a proportional increase of the runtime guarantee. Lower bounds for the ONEMINMAX and ONEJUMPZEROJUMP benchmarks show that this increase is real, i.e., the runtime is roughly proportional to the population size. This runtime behavior is very different from what is known from single-objective optimization, where results like (Jansen, Jong, and Wegener 2005; Witt 2006; Doerr and Künnemann 2015) show that often there is a regime where an increase of the population size does not lead to an increase of the runtime. Hence, for the NSGA-II, the choice of the population size is much more critical than for many single-objective algorithms (or the (G)SEMO with its flexible population size). The results most relevant for our work are the upper and lower bounds for the bi-objective ONEMINMAX, LEADINGONESTRAILINGZEROS, and ONEJUMPZEROJUMP problems. We describe these in the sections where they are relevant.

So far, there is only one runtime analysis of the NSGA-II on a problem with more than two objectives (Zheng and Doerr 2024b). It shows that the NSGA-II takes at least exponential time to optimize ONEMINMAX in at least three objectives, for any population size at most a constant factor larger than the Pareto front (see also further below). This result was recently extended to LEADINGONESTRAILINGZEROS (Doerr, Korkotashvili, and Krejca 2024).

Preliminaries

The natural numbers $\mathbb{N}$ include 0. For $m, n \in \mathbb{N}$, we define $[m..n] \coloneqq [m, n] \bigcap \mathbb{N}$ as well as $[n] \coloneqq [1..n]$.

Given $m, n \in \mathbb{N}$, an *m-objective function* f is a tuple $(f_j)_{j\in[m]}$ where, for all $j \in [m]$, it holds that $f_j : \{0,1\}^n \to \mathbb{R}$. Given an m-objective function, we implicitly assume that we are also given n. We call each $x \in \{0,1\}^n$ an *individual* and $f(x) \coloneqq (f_j(x))_{j\in[m]}$ the *objective value of* x. For each $i \in [n]$, we denote the i-th component of x by x_i. We denote the number of 1s of x by $|x|_1$, and its number of 0s by $|x|_0$.

We consider the *maximization* of m-objective functions. The objective values of an m-objective function f induce a weak partial order on the individuals, denoted by $\succeq$. For $x, y \in \{0,1\}^n$, we say that x *weakly dominates* y (written $x \succeq y$) if and only if for all $j \in [m]$ holds that $f_j(x) \geq f_j(y)$. If one of these inequalities is strict, we say that x *strictly* dominates y (written $x \succ y$). We say that x is *Pareto-optimal* if and only if x is not strictly dominated by any individual. We call the set of objective values of all Pareto-optimal individuals the *Pareto front* of f, and we call the set of all Pareto-optimal individuals the *Pareto set* of f.

The NSGA-II

The non-dominated sorting genetic algorithm II (NSGA-II, Algorithm 1) is the most popular heuristic for multi-objective optimization. It optimizes a given m-objective function iteratively, maintaining a multi-set (a *population*) of individuals of a given size $N \in \mathbb{N}_{\geq 1}$. This population is initialized with individuals chosen uniformly at random.

In each iteration, the NSGA-II generates an additional, new *offspring* population of N individuals as we detail below. Out of the $2N$ individuals from the combined population of current and offspring population, the algorithm selects N individuals as the new population for the next iteration. To this end, the NSGA-II utilizes two characteristics defined over individuals, which we also both detail below: *non-dominated ranks* and *crowding distance*. The $2N$ individuals are sorted lexicographically by first minimizing the rank and then by maximizing the crowding distance. Ties are broken uniformly at random (u.a.r.). The first N individuals from the sorted population are kept for the next iteration.

Algorithm 1 showcases the NSGA-II in a way suited toward our modification described in the following section.

Runtime. The runtime of an algorithm optimizing a function f is the (random) number of evaluations of f until the objective values of the population cover the Pareto front of f. We assume that the objective value of each individual is evaluated exactly once when it is created. Thus, the NSGA-II uses N function evaluations when initializing the population, and N function evaluations per iteration for the offspring population. As the number of function evaluations is essentially N times the number of iterations, we use the term *runtime* interchangeably for both quantities and always specify which one we mean. We state our runtime results in big-O notation with asymptotics in the problem size n.

Offspring generation. Given a *parent* population P, the NSGA-II creates N *offspring* by repeating the following *standard bit mutation* N times: Choose an individual $x \in P \subseteq \{0,1\}^n$ uniformly at random, and create a copy y of x where one flips each component of x independently with probability $1/n$. That is, for all $i \in [n]$, we have $y_i = 1 - x_i$ with independent probability $1/n$, and $y_i = x_i$ otherwise.

Non-dominated ranks. Given a population R, the *rank* of each $x \in R$ is defined inductively and roughly represents how many layers in R dominate x. Individuals in R that are not strictly dominated by any individual in R receive rank 1. The population of all such individuals is denoted as $F_1 := \{x \in R \mid \forall y \in R\colon x \nprec y\}$. The population of all individuals of rank $j \in \mathbb{N}_{\geq 2}$ is the population of all individuals in R, after removing all individuals of ranks 1 to $j-1$, that are not strictly dominated. That is, $F_j := \{x \in R \setminus \bigcup_{k\in[j-1]} F_k \mid \forall y \in R \setminus \bigcup_{k\in[j-1]}\colon x \nprec y\}$. If there are in total $r \in \mathbb{N}_{\geq 1}$ different ranks of R, then $(F_j)_{j\in[r]}$ is a partition of R.

Given such a partition of R, we say that a rank $j^* \in [r]$ is *critical* if and only if j^* is the minimal index such that the population of all individuals up to rank j^* is at least N. That is, $|\bigcup_{j\in[j^*-1]} F_j| < N$ and $|\bigcup_{j\in[j^*]} F_j| \geq N$. Individuals with a rank strictly smaller than j^* are definitely selected for the next iteration, and individuals with a strictly larger rank than j^* are definitely not selected. Among individuals with a rank of exactly j^*, some individuals may be selected and some not, thus requiring further means to make a choice.

Crowding distance. Given a population F, the *crowding distance* (CD) of each $x \in F$, denoted by $\mathrm{cDis}(x)$, is the sum of the CD of x *per objective*. The CD of x for objective $j \in [m]$ is as follows: Let $a = |F|$, and let $(S_{j.i})_{i\in[a]}$ denote F sorted by increasing value in objective j, breaking ties arbitrarily. The CD of $S_{j.1}$ and of $S_{j.a}$ for objective j is infinity. For all $i \in [2..a-1]$, the CD of $S_{j.i}$ for objective j is $\big(f(S_{j.i+1}) - f(S_{j.i-1})\big)/\big(f(S_{j.N}) - f(S_{j.1})\big)$.

Let $d \in [|F|]$, and let $(C_c)_{c\in[k]}$ be a partition of F such that for all $c \in [k]$, all individuals in C_c have the same CD and that for all $c_1, c_2 \in [k]$ with $c_1 < c_2$, the CD of C_{c_1} is strictly larger than that of C_{c_2}. We say $c^* \in [k]$ is the *critical CD index of* $(C_c)_{c\in[k]}$ *with respect to* d if and only if $|\bigcup_{c\in[c^*-1]} C_c| < d$ and $|\bigcup_{c\in[c^*]} C_c| \geq d$. When selecting d individuals from F, individuals with a CD in a population of index less than c^* are definitely selected, and those with a CD of a population with a strictly larger index are not. From C_{c^*}, some individuals may be selected and some not.

Algorithm 1: The (classic) non-dominated sorting genetic algorithm II (NSGA-II) with population size $N \in \mathbb{N}_{\geq 1}$, optimizing an m-objective function.

$P_0 \leftarrow$ population of N individuals, each u.a.r.;
$t \leftarrow 0$;
while termination criterion not met **do**
  $Q_t \leftarrow$ offspring population of P_t;
  $R_t \leftarrow P_t \cup Q_t$;
  $(F_j)_{j\in[r]} \leftarrow$ partition of R_t w.r.t. non-dom. ranks;
  $j^* \leftarrow$ critical rank of $(F_j)_{j\in[r]}$;
  $(C_c)_{c\in[k]} \leftarrow$ partition of F_{j^*} w.r.t. crowd. dist.;
  $c^* \leftarrow$ critical crowding distance index of $(C_c)_{c\in[k]}$ w.r.t. $N - |\bigcup_{j\in[j^*-1]} F_j|$;
  $s \leftarrow N - |\bigcup_{j\in[j^*-1]} F_j \cup \bigcup_{c\in[c^*-1]} C_c|$;
  $W \leftarrow$ sub-pop. of C_{c^*} of cardinality s, u.a.r.;
  $P_{t+1} \leftarrow \bigcup_{j\in[j^*-1]} F_j \cup \bigcup_{c\in[c^*-1]} C_c \cup W$;
  $t \leftarrow t+1$;

Benchmarks

We consider the three most common functions for the theoretical analysis of multi-objective search heuristics (see *Previous Works*). We define here their bi-objective versions, which are the most common ones, and build on these definitions later when defining the many-objective analogs.

The OneMinMax (OMM) benchmark (Giel and Lehre 2010) returns the number of 0s and 1s of each individual, formally, $\mathrm{OMM}\colon x \mapsto (|x|_0, |x|_1)$. Each individual is Pareto-optimal. The Pareto front is $\{(i, n-i) \mid i \in [0..n]\}$.

The OneJumpZeroJump (OJZJ) benchmark (Doerr and Zheng 2021) extends the classic Jump benchmark (Droste, Jansen, and Wegener 2002) to several objectives. It is defined similarly as OMM but has an additional parameter $k \in [2..n]$. It is effectively identical to OMM for all individuals whose number of 1s is between k and $n-k$ or is 0 or n. The objective values of these individuals constitute the Pareto front of the function. For all other individuals, the objective value is strictly worse. Hence, once an algorithm finds solutions with k or $n-k$ 1s, it needs to change at least k positions at once in a solution in order to expand the Pareto front, which is usually a hard task. Formally, for all $i \in \{0,1\}$ and all $x \in \{0,1\}^n$, let

$$J^{(i)}(x) = \begin{cases} k + |x|_i & \text{if } |x|_i \in [0..n-k] \cup \{n\}, \\ n - |x|_i & \text{else.} \end{cases}$$

Then $\mathrm{OJZJ}(x) = (J^{(1)}(x), J^{(0)}(x))$, with the Pareto front $\{(i, n+2k-i) \mid i \in [2k..n] \cup \{k, n+k\}\}$.

The LeadingOnesTrailingZeros (LOTZ) benchmark (Laumanns, Thiele, and Zitzler 2004), a multi-objective version of LeadingOnes (Rudolph 1997), returns the length of the longest prefix of 1s and suffix of 0s, formally

$$x \mapsto \left(\textstyle\sum_{i\in[n]} \prod_{j\in[i]} x_j, \sum_{i\in[n]} \prod_{j\in[i..n]} (1 - x_j)\right).$$

The Pareto front is the same as that of OMM, but the Pareto set is $\{1^i 0^{n-i} \mid i \in [0..n]\}$.

Improved Tie-Breaking for the NSGA-II

The NSGA-II selects individuals elaborately; first via the non-dominated ranks, second via the crowding distance, and last uniformly at random. While the final tie-breaker seems reasonable, it neglects the structure of the population.

In more detail, since the uniform selection is performed over the subpopulation C_{c^*} from line 11 in Algorithm 1, any imbalances with respect to different objective values are carried over in expectation to the selected population W. That is, if most of the individuals from C_{c^*} have objective value $v_1 \in \mathbb{R}^m$ and only very few have objective value $v_2 \in \mathbb{R}^m$, then it is more likely for individuals with objective value v_1 to be selected although individuals with objective value v_2 might also have interesting properties. To circumvent this problem, we propose to select the individuals from C_{c^*} as evenly as possible from all the different objective values.

Balanced tie-breaking. We replace line 11 in Algorithm 1 with the following procedure, using the same notation as in the pseudo code: Partition C_{c^*} with respect to its objective values into $(C'_c)_{c \in [a]}$, assuming a different objective values in C_{c^*}. That is, for $U := \{f(x) \mid x \in C_{c^*}\}$ (being a set without duplicates) and for each $u \in U$, there is exactly one $c \in [a]$ with $C'_c = \{x \in C_{c^*} \mid f(x) = u\}$ (where C'_c is a multi-set). For each $c \in [a]$, select $\min(|C'_c|, \lfloor s/a \rfloor)$ individuals uniformly at random from C'_c, calling this selected population $\widetilde{C}_c$. That is, $\widetilde{C}_c \subseteq C'_c$ with $|\widetilde{C}_c| = \min(|C'_c|, \lfloor s/a \rfloor)$, chosen uniformly at random among all sub-multi-sets of C'_c of cardinality $\min(|C'_c|, \lfloor s/a \rfloor)$. Add all individuals in $\bigcup_{c \in [a]} \widetilde{C}_c$ to W. If this does not select sufficiently many individuals, that is, if $|\bigcup_{c \in [a]} \widetilde{C}_c| < s$, then select the missing number of individuals uniformly at random from the remaining population, that is, from $C_{c^*} \setminus \bigcup_{c \in [a]} \widetilde{C}_c$.

Since this tie-breaking aims at balancing the amount of individuals per objective value during the third tie-breaker, we call the modified algorithm the *balanced* NSGA-II.

Additional cost. Based on our experiments in the empirical section, we observed that balanced tie-breaking is slower than random tie-breaking in terms of wall clock time by a factor of around 10 on average. However, the total time spent on balanced tie-breaking is still, on average, only 15 % of the total time spent on non-dominated sorting, which is always required during selection. Moreover, as we detail in our empirical evaluation, the overall number of function evaluations (and thus wall clock time) of the balanced NSGA-II is typically far faster than that of the classic NSGA-II.

Properties of the Balanced NSGA-II

For the classic NSGA-II, if the population size N is large enough w.r.t. the Pareto front of the objective function, no value on the Pareto front is lost. We prove that this same useful property also holds for the balanced NSGA-II.

The following lemma proves an upper bound on the number of individuals with positive crowding distance among those with critical rank. The lemma is adapted from an argument in the proof of Lemma 1 by Zheng et al. (2022).

Lemma 1. *Consider the balanced NSGA-II optimizing an m-objective function f. For each iteration $t \in \mathbb{N}$ we have that for each objective value in the critical rank $A \in f(F_{i^*})$ there exist at most $2m$ individuals $x \in F_{i^*} \subseteq R_t$ with $f(x) = A$ such that* $\mathrm{cDis}(x) > 0$.

Lemma 1 yields that in the balanced NSGA-II there is always a fair number of individuals with critical rank.

Lemma 2. *Consider the balanced NSGA-II optimizing an m-objective function. Assume that at some iteration $t \in \mathbb{N}$ we select $C \in \mathbb{N}$ individuals from the critical rank $F_{i^*} \subseteq R_t$ with size of the objective values set $|f(F_{i^*})| = S$. Then, for any $A \in f(F_{i^*})$ we keep at least $\min\big(\max(\lfloor \frac{C}{S} \rfloor - 2m, 0), |\{x \in F_{i^*} \mid f(x) = A\}|\big)$ individuals with $f(x) = A$ in P_{t+1}.*

The Balanced NSGA-II Is Efficient For Three or More Objectives

We analyze the performance of the balanced NSGA-II on OMM, LOTZ, and OJZJ with three or more objectives. We show that the balanced NSGA-II optimizes these benchmarks in polynomial time when the number m of objectives (and the gap parameter of OJZJ) is constant (Theorem 3). This result stands in strong contrast to the performance of the classic NSGA-II. Recently, Zheng and Doerr (2024b) proved that the classic NSGA-II with any population size linear in the Pareto front size cannot optimize OMM with $m \geq 3$ objectives faster than in time $\exp(\Omega(n^{\lceil m/2 \rceil}))$. Their proofs suggest that the classic NSGA-II has similar difficulties on many other many-objective problems (i.e., three or more objectives), including LOTZ and OJZJ.

We briefly state the definitions of the m-objective versions of OMM, LOTZ, and OJZJ from (Zheng and Doerr 2024b; Laumanns, Thiele, and Zitzler 2004; Zheng and Doerr 2024c) (precise definitions in the full version). All three lift the definition of the two-objective problem to an even number m of objectives by splitting the bit string into $m/2$ equal-length segments and then taking as $2i-1$-st and $2i$-th objective the original function applied to the i-th block.

We review the (few) main existing runtime result for these benchmarks. In the first mathematical runtime analysis for a many-objective problem, Laumanns et al. (2004) showed that the SEMO algorithm optimizes the COCZ and LOTZ problems in an expected number of $O(n^{m+1})$ function evaluations. COCZ is similar to OMM, so it is quite clear that the relevant part of their proof also applies to OMM, giving again an $O(n^{m+1})$ bound. Also, it is easy to see that their analysis can be extended to the GSEMO, giving the same runtime guarantees. The bounds for COCZ were improved slightly to $O(n^m)$, and $O(n^3 \log n)$ for $m = 4$, by Bian et al. (2018). Huang et al. (2021) analyzed how the MOEA/D optimizes COCZ and LOTZ. We skip the details since the MOEA/D is very different from all other algorithms discussed in this work. Wietheger and Doerr (2023) proved a runtime guarantee of $O(Nn \log n)$ for the NSGA-III optimizing the 3-objective OMM problem when the population size is at least the size of the Pareto front. The only many-objective results for OJZJ_k are an $O(M^2 n^k)$ bound for the GSEMO and an $O(\mu M n^k)$ bound for the SMS-EMOA with population size $\mu \geq M$, where M is the size of the Pareto front, see (Zheng and Doerr 2024c). Note that as the runtimes of many-objective problems are not too well understood, and in the absence of any reasonable lower bound, there is a high risk that the results above are far from tight.

Before we state our main result of this section, we note that our main goal is to show the drastic difference to the behavior of the classic NSGA-II exhibited by Zheng and Doerr (2024b). We do not optimize our runtime estimates with more elaborate methods but are content with polynomial-time bounds for constant m and k and population sizes linear in the Pareto front size. Near-tight bounds for many-objective evolutionary optimization of our benchmarks were recently proven in (Wietheger and Doerr 2024). For the same reason, we also do not prove bounds that do not show an increase of the runtime with growing population size in certain ranges (as we do for two objectives), though clearly this would be possible with similar arguments.

Theorem 3. *Let $m \in \mathbb{N}$ be even and $m' = m/2$. Assume that $n' = n/m' \in \mathbb{Z}$. Let $k \in [2..n'/2]$. Consider the* OMM*,* LOTZ*, or* OJZJ_k *problem. Denote by M the size of the Pareto front and by S the size of a largest set of pairwise incomparable solutions. Assume that we optimize these problems via the balanced NSGA-II with population size $N \geq S + 2m(n'+1) = S + 4n + 2m$. Then we have the following bounds for the expected runtime:*

1. *For* OMM*, it is at most $2enM$ iterations.*
2. *For* LOTZ*, it is at most $2enM + 2en^2$ iterations.*
3. *For* OJZJ_k*, it is at most $2en^kM + 2ekm'n$ iterations.*

As many runtime analyses, our bounds depend on the size S of the largest incomparable set of solutions the problem admits. For this, the following bounds are known or can easily be found: For OMM, we have $S = M = (n'+1)^{m/2}$, for LOTZ, we have $S \leq (n'+1)^{m-1}$ (Opris et al. 2024), and for OJZJ, we have $S \leq (n'+1)^{m/2}$.

The reason for the drastically different behavior of the classic and the balanced NSGA-II is that the former can lose Pareto optimal solution values with any population size that is linear in the size of the Pareto front. In contrast, for the balanced NSGA-II often a population size exceeding the Pareto front size only by a lower-order term suffices to prevent such a loss of objective values. This follows easily from arguments similar to those used to prove Lemma 2. For the convenience of this and possible future works, we formulate and prove this crucial statement as a separate lemma.

Lemma 4. *Consider the balanced NSGA-II with population size N optimizing some m-objective optimization problem. Assume that S is an upper bound on the size of any set of pair-wise incomparable solutions. Assume that U is an upper bound on the number of individuals with positive crowding distance in a set of solutions such that any two are incomparable or have identical objective values. If $N \geq S + U$, then the following survival property holds.*

Assume that at some time t the combined parent and offspring population R_t contains a solution x that is contained in the first front F_1 of the non-dominated sorting of R_t. Then its objective value survives into the next generation, i.e., surely, P_{t+1} contains an individual y such that $f(y) = f(x)$.

The results above show that the balanced NSGA-II does not have the efficiency problems of the classic NSGA-II for even numbers $m \geq 4$ of objectives. Since Zheng and Doerr (2024b) show that already the case $m = 3$ is problematic for the classic NSGA-II, we show that the balanced NSGA-II optimizes OMM for three objectives also efficiently. The first objective counts the number of zeros in the argument $x \in \{0,1\}^n$; the second and third objectives count the numbers of ones in the first and second half of x, resp.

Theorem 5. *Consider optimizing the 3-objective* OMM *problem via the balanced NSGA-II with population size $N \geq (\frac{n}{2}+1)^2 + 4n + 6$. Then after an expected number of at most $2en(\frac{n}{2}+1)^2$ iterations, the Pareto front is found.*

Runtime Analysis on Bi-Objective OMM

We bound the expected runtime of the balanced NSGA-II on OMM by $O(n + \frac{n^2 \log n}{N})$ iterations, hence $O(Nn + n^2 \log n)$ function evaluations, when the population size at least four times the Pareto front size (Corollary 9). This bound is $O(n^2 \log n)$ function evaluations when $N = O(n \log n)$.

To put this result into perspective, we note that the classic NSGA-II, again for $N \geq 4(n+1)$, satisfies the guarantee of $O(n \log n)$ iterations, that is, $O(Nn \log n)$ function evaluations (Zheng and Doerr 2023). This bound is asymptotically tight for all $N \leq n^{2-\varepsilon}$, $\varepsilon > 0$ any constant (Doerr and Qu 2023b). Hence the classic NSGA-II obtains a $\Theta(n^2 \log n)$ runtime (function evaluations) only with the smallest admissible population size of $\Theta(n)$. Recently, a bound of $O(Nn \log n)$ function evaluations was also proven for the SPEA2 (Ren et al. 2024a). For completeness, we note that the simplistic SEMO algorithm finds the full Pareto front of OMM in $O(n^2 \log n)$ iterations and function evaluations (Giel and Lehre 2010). This result can easily be extended to the GSEMO algorithm. A matching lower bound of $\Omega(n^2 \log n)$ was shown for the SEMO in (Covantes Osuna et al. 2020) and for the GSEMO in (Bossek and Sudholt 2024). An upper bound of $O(\mu n \log n)$ function evaluations was shown for the hypervolume-based $(\mu + 1)$ SIBEA with $\mu \geq n + 1$ (Nguyen, Sutton, and Neumann 2015).

As all individuals are Pareto-optimal for OMM, the runtime follows from how fast the algorithm spreads its population on the Pareto front. We bound this time by considering the extremities of the currently covered Pareto front, i.e., the individuals with the largest number of 1s or of 0s. Those are turned into individuals with one more 1 or 0, respectively, within about n iterations in expectation, requiring only a single bit flip. The *balance* property of the algorithm guarantees that the number of individuals at the extremities stays at about $\frac{N}{n+1}$ (Lemma 6). This number is quickly reached and then used to expand the Pareto front (Lemma 7).

We also prove a general result that bounds the expected time to cover certain parts of the Pareto front (Theorem 8).

We use Lemma 1 from Zheng, Liu, and Doerr (2022), which also applies to the balanced NSGA-II, as it does not impose any restrictions on how to choose from individuals with the same CD. The lemma states that a Pareto-optimal objective value in the population is never lost.

The following lemma is a direct application of Lemma 2. It shows that the population maintains all individuals per objective value it found so far, up to a bound of $\lfloor \frac{N}{n+1} \rfloor - 4$.

Lemma 6. *Consider the balanced NSGA-II with population size $N \geq 4(n+1)$ on* OMM. *Then, for each objective value $(k, n-k)$ with $k \in [0..n]$, from the individuals x in R_t with $f(x) = (k, n-k)$, the population P_{t+1} contains at least $\min(\lfloor \frac{N}{n+1} \rfloor - 4, |\{x \in R_t \mid f(x) = (k, n-k)\}|)$.*

Lemma 6 shows that the population can maintain subpopulations of a size about $\frac{N}{n+1}$. Once such a size is reached, there is a decent chance to extend the Pareto front. The following lemma formalizes how quickly this happens.

Lemma 7. *Consider the balanced NSGA-II with population size $N \geq 4(n+1)$ on* OMM. *For $v \in [1..n]$ and $i \in \{1, 2\}$, let T_v^i denote the number of iterations needed, starting with an individual x_0 in the parent population with $f_i(x_0) = v$, to obtain an individual x_f in the resulting population such that $f_i(x_f) = v - 1$. Then, $E[T_v^i] = O(\log \lceil \frac{n}{v} \rceil + \frac{n^2}{Nv} + 1)$.*

We prove Lemma 7 via the multiplicative up-drift theorem by Doerr and Kötzing (2021, Theorem 3). This theorem provides a bound on the expected number of steps for a random process to grow to a certain number if it increases in every single step by a multiple of its expected value.

Using Lemma 7 lets us prove the following more general result of the bound for expanding the Pareto front. It shows how quickly the population expands on a symmetric portion of the Pareto front, centered around $n/2$.

Theorem 8. *Consider the balanced NSGA-II with population size $N \geq 4(n+1)$ optimizing* OMM. *Let $\alpha \in [0..\lfloor \frac{n}{2} \rfloor]$, and assume that there exists an $x \in P_0$ with $|x|_1 \in [\alpha, n-\alpha]$. Then the expected number of iterations to cover $\{x \in \{0,1\}^n \mid |x|_1 \in [\alpha, n-\alpha]\}$ is $O(n + \frac{n^2 \log n}{N})$.*

For $\alpha = 0$, since the Pareto front covers the entire population, we get the following runtime bound for OMM.

Corollary 9. *The expected runtime of balanced NSGA-II with population size $N \geq 4(n+1)$ optimizing* OMM *is $O(n + \frac{n^2 \log n}{N})$ iterations, i.e., $O\big(nN + n^2 \log n\big)$ expected function evaluations.*

Runtime Analysis on Bi-Objective OJZJ

We analyze the runtime of the balanced NSGA-II on OJZJ_k with $k = [2..\frac{n}{2}]$. We show that this time is $O(n + \frac{n^{k+1}}{N})$ iterations and thus $O(Nn + n^{k+1})$ function evaluations when the population size is at least four times the size of the Pareto front (Theorem 13). This guarantee is $O(n^{k+1})$ function evaluations when $N = O(n^k)$, exhibiting a large parameter range with the asymptotically best runtime guarantee.

Previous results on the runtime of MOEAs on this benchmark include an $O(n^{k+1})$ iterations and function evaluations guarantee for the GSEMO and the result that the SEMO cannot optimize this benchmark (Doerr and Zheng 2021). An upper bound of $O(n^k)$ iterations, hence $O(Nn^k)$ function evaluations, was shown for the classic NSGA-II with population size at least four times the size of the Pareto front (Doerr and Qu 2023a). The latter bound is tight apart from constant factors when $N = o(n^2/k^2)$ (Doerr and Qu 2023b). The same bound $O(Nn^{k+1})$ was recently proven for the SPEA2 (Ren et al. 2024a). For the SMS-EMOA with population size $\mu \geq n - 2k + 3$, the bounds of $O(n^k)$ and $\Omega(n^k/\mu)$ iterations, hence $O(\mu n^k)$ and $\Omega(n^k)$ function evaluations, were shown by Bian et al. (2023). The authors also show that a stochastic population update reduces the runtime by a factor of order $\min\{1, \mu/2^{k/4}\}$. (Ren et al. 2024b) show an expected runtime of $O(N^2 4^k + Nn \log n)$ for a modified version of the NSGA-II that reorders individuals by maximizing the Hamming distance and uses crossover.

In our analysis of the balanced NSGA-II, we follow the approach of Doerr and Qu (2023a), who analyzed how the classic NSGA-II optimizes this benchmark. This means we split the analysis into three stages. The first stage bounds the time to find a solution on the inner Pareto front. The second stage bounds the time to cover the inner Pareto front. The third stage bounds the time to cover the outer Pareto front, which consists of only two objective values. We show in the full version that once the algorithm enters a later stage, it does not return to an earlier one. Thus, we bound the expected number of iterations by separately analyzing each stage.

For stage 1, we find that with very high probability at least one of the initial individuals is on the inner Pareto front. This leads, in expectation, to a constant length of this stage.

Lemma 10. *Regardless of the population size N and of the initial population, for all $k \in [2..n/2]$, stage 1 needs an expected number of at most $\frac{e}{N}k^k + 1$ iterations.*

For stage 2, as OJZJ_k and OMM are similar, we apply Theorem 8 with $\alpha = k$, resulting in the following lemma.

Lemma 11. *Using population size $N \geq 4(n - 2k + 3)$, stage 2 needs in expectation $O(n + \frac{n^2 \log n}{N})$ iterations.*

For stage 3, the arguments follow those for Lemma 7, but now with a *jump* of k bits instead of 1.

Lemma 12. *Using $N \geq 4(n - 2k + 3)$, stage 3 needs in expectation $O(\log \frac{N}{n}) + 2 + \frac{8en^k(n+1)}{N}$ iterations.*

Combining the results for all three stages, we obtain the following runtime guarantee.

Theorem 13. *The expected runtime of the balanced NSGA-II with population size $N \geq 4(n - 2k + 3)$ on OJZJ_k with $k \in [2..\frac{n}{2}]$ is $O(n + n^{k+1}/N)$ iterations, i.e., $O(Nn + n^{k+1})$ expected function evaluations.*

Runtime Analysis on Bi-Objective LOTZ

We bound the expected runtime of the balanced NSGA-II on LOTZ by $O(\frac{n^3}{N} + n \log \frac{N}{n+1})$ iterations, i.e., $O(n^3 + Nn \log \frac{N}{n+1})$ function evaluations, when the population size is at least four times the Pareto front size (Theorem 15). This bound is $O(n^3)$ function evaluations for $N = O(\frac{n^2}{\log n})$.

The following runtime results are known for LOTZ. The SEMO takes $\Theta(n^3)$ iterations and function evaluations (Laumanns, Thiele, and Zitzler 2004). For the GSEMO, Giel (2003) showed an upper bound of $O(n^3)$ iterations and evaluations, a matching lower bound was proven only for unrealistically small mutation rates (Doerr, Kodric, and Voigt 2013). An upper bound of $O(\mu n^2)$ iterations and function evaluations was shown for the $(\mu+1)$ SIBEA with population size $\mu \geq n+1$ (Brockhoff, Friedrich, and Neumann 2008). The classic NSGA-II with population size at least $4(n+1)$ solves the LOTZ problem in $O(n^2)$ iterations and thus $O(Nn^2)$ function evaluations. The same bound was recently proven for the SPEA2 (Ren et al. 2024a). For the SMS-EMOA with population size $\mu \geq n+1$, a runtime guarantee of $O(\mu n^2)$ iterations and function evaluations was given by Zheng and Doerr (2024c).

Our analysis considers two phases. The first phase bounds the time until the current population contains the all-1s bit string, which is Pareto-optimal. The second phase bounds the remaining time until the entire Pareto front is covered. Both phases take about the same time in expectation.

During the first phase, we consider individuals with an increasing prefix of 1s. In the second phase, we consider individuals with an increasing suffix of 0s. Each such improvement denotes a segment. Each segment consists of the following two steps: (1) We bound the time until the population contains about $\frac{N}{n+1}$ individuals that can easily be turned into improving offspring. This step takes $O(\log \frac{N}{n+1})$ in expectation (Lemma 14). (2) We bound the time to create an improving offspring by $O(\frac{n^2}{N})$ in expectation. Theorem 15 then follows since there are at most n segments per phase.

Lemma 14. *Consider the balanced NSGA-II with population size $N \geq 4(n+1)$ optimizing the* LOTZ $=: f$ *function. Let t_0 be any iteration. Furthermore, for all $t \in N$, let $v = \max_{y \in R_{t_0+t}} f_1(y)$ and $Y_t = \{y \in R_{t_0+t} \mid f_1(y) = v\}$. Last, let T denote the first iteration $t \in \mathbb{N}$ such that $|Y_t| \geq \max(1, \lfloor \frac{N}{n+1} \rfloor - 4) =: B$ or such that there is a $z \in R_{t+t_0}$ with $f_1(z) > v$. Then $E[T \mid t_0] = O(\log B)$.*

The same statement holds when exchanging f_1 by f_2.

Lemma 14 is sufficient to prove our main result.

Theorem 15. *The expected runtime of the balanced NSGA-II with $N \geq 4(n+1)$ on* LOTZ *is $O(\frac{n^3}{N} + n \log \frac{N}{n+1})$ iterations, i.e., $O(n^3 + Nn \log \frac{N}{n+1})$ function evaluations.*

Empirical Runtime Analysis

We complement our theoretical results with experiments, aiming to see how much the population size of the NSGA-II actually influences the number of function evaluations. Our code is publicly available (Doerr, Ivan, and Krejca 2024b).

We run the classic and the balanced NSGA-II on OMM, for problem sizes $n \in 10 \cdot [3..12]$ and three population sizes N. For each combination of n and N per algorithm, we start 50 independent runs and log the number of function evaluations until the Pareto front is covered for the first time. Let $M = n + 1$ denote the size of the Pareto front. We consider the choices $N \in \{2M, 4M, 8M, 16M\}$, noting that our theoretical result holds for all $N \geq 4M$ (Corollary 9).

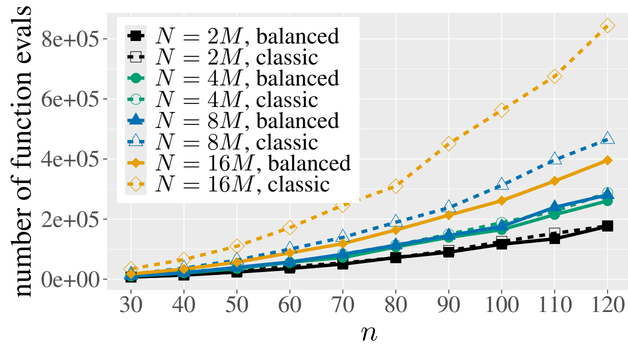

Figure 1: The average number of function evaluations of the classic (dashed lines) and the balanced (solid lines) NSGA-II optimizing ONEMINMAX, for the shown population sizes N and problem sizes n. The value M denotes the size of the Pareto front, i.e., $M = n + 1$. Each point is the average of 50 independent runs.

Figure 1 shows that the runtime increases for both algorithms for increasing N. This additional cost is larger for the classic NSGA-II than for the balanced one, statistically significantly for $N \in \{8M, 16M\}$. Hence, not choosing the optimal population size for the classic NSGA-II is already for constant factors more penalized than for the balanced NSGA-II. For $N \in \{2M, 4M\}$, the runtime of the classic and the balanced NSGA-II is roughly the same. This shows that the classic NSGA-II still can perform very well for a careful choice of N. If the optimal choice for N is unknown, the balanced NSGA-II is a more robust choice.

We note that qualitatively very similar results hold for OJZJ and LOTZ (as shown in the full version). Moreover, we show in the full version that the balanced NSGA-II optimizes the 4-objective OMM problem quickly.

Conclusion

We propose and analyze a simple tie-breaking modification to the classic NSGA-II, aiming to distribute individuals more evenly among different objective values with the same non-dominated rank and crowding distance. Our theoretical results prove two major advantages of this modification: (1) It is capable of efficiently optimizing multi-objective functions with at least three objectives for which the classic NSGA-II has at least an exponential expected runtime. (2) Not choosing an optimal population size is far less important and leads for certain ranges to no asymptotic runtime loss. Our experiments show that this effect can be already very clear when choosing slightly sub-optimal population sizes.

For future work, it would be interesting to prove lower bounds for the bi-objective scenarios and to improve the bounds in the many-objective setting. Furthermore, we are optimistic that similar results could be obtained for other MOEAs that resort to random tie-breaking at some stage, for example, the NSGA-III (Deb and Jain 2014) and the SMS-EMOA (Beume, Naujoks, and Emmerich 2007).

Acknowledgments

This research benefited from the support of the FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

References

Auger, A.; and Doerr, B., eds. 2011. *Theory of Randomized Search Heuristics*. World Scientific Publishing.

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

Bian, C.; and Qian, C. 2022. Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In *Parallel Problem Solving From Nature, PPSN 2022*, 428–441. Springer.

Bian, C.; Qian, C.; and Tang, K. 2018. A general approach to running time analysis of multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2018*, 1405–1411. ijcai.org.

Bian, C.; Ren, S.; Li, M.; and Qian, C. 2024. An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2024*, 6905–6913. ijcai.org.

Bian, C.; Zhou, Y.; Li, M.; and Qian, C. 2023. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5513–5521. ijcai.org.

Bossek, J.; and Sudholt, D. 2024. Runtime analysis of quality diversity algorithms. *Algorithmica*, 86: 3252–3283.

Brockhoff, D.; Friedrich, T.; and Neumann, F. 2008. Analyzing hypervolume indicator based algorithms. In *Parallel Problem Solving from Nature, PPSN 2008*, 651–660. Springer.

Cerf, S.; Doerr, B.; Hebras, B.; Kahane, J.; and Wietheger, S. 2023. The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5522–5530. ijcai.org.

Coello, C. A. C.; Lamont, G. B.; and van Veldhuizen, D. A. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edition.

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

Dang, D.-C.; Opris, A.; Salehi, B.; and Sudholt, D. 2023a. Analysing the robustness of NSGA-II under noise. In *Genetic and Evolutionary Computation Conference, GECCO 2023*, 642–651. ACM.

Dang, D.-C.; Opris, A.; Salehi, B.; and Sudholt, D. 2023b. A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In *Conference on Artificial Intelligence, AAAI 2023*, 12390–12398. AAAI Press.

Deb, K.; and Jain, H. 2014. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18: 577–601.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6: 182–197.

Deng, R.; Zheng, W.; Li, M.; Liu, J.; and Doerr, B. 2024. Runtime analysis for state-of-the-art multi-objective evolutionary algorithms on the subset selection problem. In Affenzeller, M.; Winkler, S. M.; Kononova, A. V.; Trautmann, H.; Tusar, T.; Machado, P.; and Bäck, T., eds., *Parallel Problem Solving from Nature, PPSN 2024, Part III*, 264–279. Springer.

Doerr, B.; Ivan, T.; and Krejca, M. S. 2024a. Speeding Up the NSGA-II With a Simple Tie-Breaking Rule. *CoRR*, abs/2412.11931.

Doerr, B.; Ivan, T.; and Krejca, M. S. 2024b. Speeding Up the NSGA-II With a Simple Tie-Breaking Rule (Code). Zenodo. https://doi.org/10.5281/zenodo.14501034.

Doerr, B.; Kodric, B.; and Voigt, M. 2013. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2013*, 432–439. IEEE.

Doerr, B.; Korkotashvili, D.; and Krejca, M. S. 2024. Difficulties of the NSGA-II with the Many-Objective LeadingOnes Problem. *CoRR*, abs/2411.10017.

Doerr, B.; and Kötzing, T. 2021. Multiplicative up-drift. *Algorithmica*, 83: 3017–3058.

Doerr, B.; and Künnemann, M. 2015. Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—different asymptotic runtimes for different instances. *Theoretical Computer Science*, 561: 3–23.

Doerr, B.; and Neumann, F., eds. 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.

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

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

Doerr, B.; and Qu, Z. 2023c. Runtime analysis for the NSGA-II: Provable speed-ups from crossover. In *Conference on Artificial Intelligence, AAAI 2023*, 12399–12407. AAAI Press.

Doerr, B.; and Zheng, W. 2021. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, 12293–12301. AAAI Press.

Droste, S.; Jansen, T.; and Wegener, I. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276: 51–81.

Fortin, F.; and Parizeau, M. 2013. Revisiting the NSGA-II crowding-distance computation. In *Genetic and Evolutionary Computation Conference, GECCO 2013*, 623–630. ACM.

Giel, O. 2003. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2003*, 1918–1925. IEEE.

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

Huang, Z.; Zhou, Y.; Luo, C.; and Lin, Q. 2021. A runtime analysis of typical decomposition approaches in MOEA/D framework for many-objective optimization problems. In *International Joint Conference on Artificial Intelligence, IJCAI 2021*, 1682–1688.

Jansen, T. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.

Jansen, T.; Jong, K. A. D.; and Wegener, I. 2005. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13: 413–440.

Laumanns, M.; Thiele, L.; and Zitzler, E. 2004. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8: 170–182.

Laumanns, M.; Thiele, L.; Zitzler, E.; Welzl, E.; and Deb, K. 2002. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving from Nature, PPSN 2002*, 44–53. Springer.

Li, Y.-L.; Zhou, Y.-R.; Zhan, Z.-H.; and Zhang, J. 2016. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20: 563–576.

Neumann, F.; and Witt, C. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.

Nguyen, A. Q.; Sutton, A. M.; and Neumann, F. 2015. Population size matters: rigorous runtime results for maximizing the hypervolume indicator. *Theoretical Computer Science*, 561: 24–36.

Opris, A.; Dang, D. C.; Neumann, F.; and Sudholt, D. 2024. Runtime analyses of NSGA-III on many-objective problems. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, 1596–1604. ACM.

Ren, S.; Bian, C.; Li, M.; and Qian, C. 2024a. A first running time analysis of the Strength Pareto Evolutionary Algorithm 2 (SPEA2). In *Parallel Problem Solving from Nature, PPSN 2024, Part III*, 295–312. Springer.

Ren, S.; Qiu, Z.; Bian, C.; Li, M.; and Qian, C. 2024b. Maintaining diversity provably helps in evolutionary multimodal optimization. In *International Joint Conference on Artificial Intelligence, IJCAI 2024*, 7012–7020. ijcai.org.

Rudolph, G. 1997. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovăc.

Thierens, D. 2003. Convergence time analysis for the multi-objective counting ones problem. In *Evolutionary Multi-Criterion Optimization, EMO 2003*, 355–364. Springer.

Wietheger, S.; and Doerr, B. 2023. A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5657–5665. ijcai.org.

Wietheger, S.; and Doerr, B. 2024. Near-tight runtime guarantees for many-objective evolutionary algorithms. In *Parallel Problem Solving from Nature, PPSN 2024, Part IV*, 153–168. Springer.

Witt, C. 2006. Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14: 65–86.

Zheng, W.; and Doerr, B. 2023. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *Artificial Intelligence*, 325: 104016.

Zheng, W.; and Doerr, B. 2024a. Approximation guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). *IEEE Transactions on Evolutionary Computation*. In press, https://doi.org/10.1109/TEVC.2024.3402996.

Zheng, W.; and Doerr, B. 2024b. Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. *IEEE Transactions on Evolutionary Computation*, 28: 1442–1454.

Zheng, W.; and Doerr, B. 2024c. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*, 20874–20882. AAAI Press.

Zheng, W.; Liu, Y.; and Doerr, B. 2022. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Conference on Artificial Intelligence, AAAI 2022*, 10408–10416. AAAI Press.

Zhou, A.; Qu, B.-Y.; Li, H.; Zhao, S.-Z.; Suganthan, P. N.; and Zhang, Q. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1: 32–49.

Zhou, Z.-H.; Yu, Y.; and Qian, C. 2019. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer.