From Understanding the Population Dynamics of the NSGA-II to the First Proven Lower Bounds

Benjamin Doerr, Zhongdi Qu

Laboratoire d'Informatique (LIX), Ecole Polytechnique, CNRS, Institut Polytechnique de Paris, Palaiseau, France
doerr@lix.polytechnique.fr, d.q.1124@gmail.com

Abstract

Due to the more complicated population dynamics of the NSGA-II, none of the existing runtime guarantees for this algorithm is accompanied by a non-trivial lower bound. Via a first mathematical understanding of the population dynamics of the NSGA-II, that is, by estimating the expected number of individuals having a certain objective value, we prove that the NSGA-II with suitable population size needs $\Omega(Nn\log n)$ function evaluations to find the Pareto front of the ONEMINMAX problem and $\Omega(Nn^k)$ evaluations on the ONEJUMPZEROJUMP problem with jump size k. These bounds are asymptotically tight (that is, they match previously shown upper bounds) and show that the NSGA-II here does not even in terms of the parallel runtime (number of iterations) profit from larger population sizes. For the ONEJUMPZEROJUMP problem and when the same sorting is used for the computation of the crowding distance contributions of the two objectives, we even obtain a runtime estimate that is tight including the leading constant.

Introduction

Many real-world problems have several, often conflicting objectives. For such *multi-objective* optimization problems, it is hard to compute a single solution. Instead, one usually computes a set of incomparable, interesting solutions from which a decision maker can select the most preferable one. Due to their population-based nature, evolutionary algorithms (EAs) are well suited for such problems, and in fact, are intensively used in multi-objective optimization.

The most accepted multi-objective evolutionary algorithm (MOEA) in practice (Zhou et al. 2011) is the *non-dominated sorting genetic algorithm II* (NSGA-II) proposed by Deb et al. (2002). It uses a fixed population size N, generates N new solutions per iteration, and selects the next population according to the non-dominated sorting of the combined parent and offspring population and the crowding distance. Due to this complex structure, no mathematical runtime analyses existed for this algorithm until the work (Zheng, Liu, and Doerr 2022), which was soon followed by (Zheng and Doerr 2022; Bian and Qian 2022; Doerr and Qu 2022a).

Interestingly, and different from the previous runtime analyses of other MOEAs, none of these works proved a non-trivial lower bound on the runtime. Such bounds are important since only by comparing upper and lower bounds for different algorithms can one declare one algorithm superior to another. Such bounds are also necessary to have tight runtime estimates, from which information about optimal parameter values can be obtained.

The lack of lower bounds for the NSGA-II, naturally, is caused by the more complicated population dynamics of this algorithm. While for algorithms like the SEMO or global SEMO predominantly analyzed in MOEA theory, it follows right from the definition of the algorithm that there can be at most one individual per objective value, such structural information does not exist for the NSGA-II.

In this work, we gain a first deeper understanding of the population dynamics of the NSGA-II (more precisely, the mutation-based version regarded in almost all previous runtime results for the NSGA-II). For the optimization of the ONEMINMAX and the ONEJUMPZEROJUMP benchmark, we prove that also for relatively large population sizes, only a constant number of individuals exists on the outer positions of the Pareto front. This information allows us to prove upper bounds on the speed with which the Pareto front is explored, and finally yields lower bounds on the runtime of the NSGA-II on these two benchmarks.

More specifically, we prove the following lower bound for the ONEJUMPZEROJUMP benchmark (we do not discuss here in detail the result for ONEMINMAX and refer instead to Theorem 16). Let N denote the population size of the NSGA-II, n the problem size (length of the bit-string encoding), and k the gap parameter of the ONEJUMPZEROJUMP problem. If N is at least 4 times the size $n - 2k + 3$ of the Pareto front and $N = o(n^2/k^2)$, then the time to compute the Pareto front of the ONEJUMPZEROJUMP problem is at least $(\frac{3(e-1)}{8} - o(1))Nn^k$ fitness evaluations or, equivalently, $(\frac{3(e-1)}{8} - o(1))n^k$ iterations. This result shows that the upper bound of $O(Nn^k)$ fitness evaluations or $O(n^k)$ iterations proven by Doerr and Qu (2022a) is asymptotically tight for broad ranges of the parameters. In particular, this shows that there is no advantage in using a population size larger than the smallest admissible one, not even when taking the number of iterations as the performance measure. This is very different from the single-objective world, where, for example, the runtime of the $(1 + \lambda)$ EA on the single-objective JUMP problem with jump size k is easily seen to be $\Omega(n^k/\lambda)$ and $O(n^k/\lambda + n \log n)$ iterations, hence for $\lambda = o(n^{k-1}/\log n)$ the number of iterations reduces with growing value of λ and the number of fitness evaluations does not change (when ignoring lower-order terms). This comparison suggests that the choice of the population size is more critical for the NSGA-II than for single-objective EAs.

For the variant of the NSGA-II which uses the same sorting to compute the crowding distance contribution of both objectives (which is a natural choice for two objectives), we can even determine the runtime precise apart from lower order terms. To this aim, we also exploit our new understanding of the population dynamics to prove a tighter upper bound on the runtime. We shall not exploit this further in this work, but we note that such tight analyses are the prerequisite for optimizing parameters, here for example the mutation rate. To the best of our knowledge, this is only the second runtime analysis of a MOEA that determines the leading constant of the runtime (the other one being the analysis of the synthetic GSEMO algorithm on the ONEJUMPZEROJUMP benchmark).

Overall, this work constitutes a first step toward understanding the population dynamics of the NSGA-II. We exploit this to prove the first asymptotically tight lower bounds. In a non-trivial special case, we even determine a runtime precise apart from lower-order terms. These results already give some information on the optimal parameter values, and we are optimistic that our methods can lead to more insights about the right parameter choices of the NSGA-II.

Previous Works

For a general introduction to multi-objective optimization via evolutionary algorithms, including the most prominent algorithm NSGA-II (47000 citations on Google scholar), we refer to Zhou et al. (2011). This work is concerned with the mathematical runtime analysis of a MOEA, which is a subarea of the broader research area of runtime analyses for randomized search heuristics (Neumann and Witt 2010; Auger and Doerr 2011; Jansen 2013; Doerr and Neumann 2020).

The first runtime analyses of MOEAs date back to the early 2000s (Laumanns et al. 2002; Giel 2003; Thierens 2003) and regarded the artificial SEMO and GSEMO algorithms, which are still the most regarded algorithms in MOEA theory (see, e.g., Bian, Qian, and Tang (2018); Qian et al. (2019); Qian, Liu, and Zhou (2022) for some recent works). Some time later, the first analyses of the more realistic SIBEA (Brockhoff, Friedrich, and Neumann 2008; Nguyen, Sutton, and Neumann 2015; Doerr, Gao, and Neumann 2016) and the MOEA/D (Li et al. 2016; Huang et al. 2019; Huang and Zhou 2020; Huang et al. 2021) followed. Very recently, the first runtime analysis of the NSGA-II appeared (Zheng, Liu, and Doerr 2022), which was quickly followed up by further runtime analyses of this algorithm.

Zheng, Liu, and Doerr (2022) proved that the NSGA-II with population size N can efficiently optimize the classic ONEMINMAX and LOTZ benchmarks if N is at least four times the size of the Pareto front. A population size equal to the size of the Pareto front does not suffice. In this case, the NSGA-II loses desired solutions often enough so that only a constant fraction of the Pareto front is covered for at least exponential time. However, with smaller population sizes the NSGA-II can still compute good approximations to the Pareto front, as proven by Zheng and Doerr (2022) again for the ONEMINMAX benchmark. The first runtime analysis on a benchmark with multimodal objectives (Doerr and Qu 2022a) showed that the NSGA-II, again with population size N at least four times the size of the Pareto front, computes the Pareto front of the ONEJUMPZEROJUMP benchmark with jump parameter $k \in [2..n/4]$ in expected time at most $O(Nn^k)$. These three works regard a version of the NSGA-II without crossover. Bian and Qian (2022) regarded the original NSGA-II with crossover, but did not show better runtime guarantees than what was previously shown without crossover. Via a different selection method, however, they obtained significant speed-ups.

For none of the runtime guarantees proven in these works, a matching (or at least non-trivial) lower bound was shown. The apparent reason, spelled out explicitly in the conclusion of (Doerr and Qu 2022a), is the lack of understanding of the population dynamics for this algorithm. We note that this problem exists even for the simpler SEMO/GSEMO algorithm despite its much more restricted population dynamics (in particular, the strict selection mechanism of these algorithms ensures that for each objective value there is at most one individual in the population). For the SEMO algorithm, which uses one-bit mutation, a lower bound matching the $O(n^2 \log n)$ upper bound on ONEMINMAX of Giel and Lehre (2010) was shown ten years later by Covantes Osuna et al. (2020). For the GSEMO, using bit-wise mutation instead, no lower bound is known for the ONEMINMAX benchmark. Similarly, for the LOTZ benchmark a tight $\Theta(n^3)$ runtime of the SEMO was proven by Laumanns et al. (2002) already. The same upper bound was proven for the GSEMO by Giel (2003), but the only lower bound (Doerr, Kodric, and Voigt 2013) for this problem is valid only for an unrealistically small mutation rate. Only for the ONEJUMP-ZEROJUMP benchmark, a tight bound of $\Theta(n^k)$ was proven also for the GSEMO (Doerr and Zheng 2021), clearly profiting from the fact that the population can contain at most one individual on the local optimum of the objectives.

Preliminaries

The NSGA-II Algorithm

In the interest of brevity, we only give a brief overview of the algorithm here and refer to Deb et al. (2002) for a more detailed description of the general algorithm and to Zheng, Liu, and Doerr (2022) for more details on the particular version of the NSGA-II we regard.

The NSGA-II uses two metrics, rank and crowding distance, to completely order any population. The ranks are defined recursively based on the dominance relation. All non-dominated individuals have rank 1. Then, given that the individuals of ranks $1, \dots, k$ are defined, the individuals of rank $k + 1$ are those not dominated except by individuals of rank k or smaller. This defines a partition of the population into sets $F_1, F_2, \dots$ such that F_i contains all individuals with rank

i. Clearly, individuals with lower ranks are preferred. The crowding distance, denoted by $\mathrm{cDis}(x)$ for an individual x, is used to compare individuals of the same rank. To compute the crowding distances of individuals of rank i with respect to a given objective function f_j, we first sort the individuals in ascending order according to their f_j objective values. The first and last individuals in the sorted list have infinite crowding distance. For the other individuals, their crowding distance is the difference between the objective values of their left and right neighbors in the sorted list, normalized by the difference between the minimum and maximum values. The final crowding distance of an individual is the sum of its crowding distances with respect to each objective function. Among individuals of the same rank, the ones with higher crowding distances are preferred.

The algorithm starts with a random initialization of a parent population of size N. In each iteration, N children are generated from the parent population via a variation operator, and N best individuals among the combined parent and children population survive to the next generation based on their ranks and, as a tie-breaker, the crowding distance. At each iteration, the critical rank i^* is the rank such that if we take all individuals of ranks smaller than i^*, the total number of individuals will be less than or equal to N, but if we also take all individuals of rank i^*, the total number of individuals will be more than N. Thus, all individuals of rank smaller than i^* survive to the next generation, and for individuals of rank i^*, we take the individuals with the highest crowding distance, breaking ties randomly, so that in total exactly N individuals are kept. In practice, the algorithm is run until some stopping criterion is met. In our mathematical analysis, we are interested in how long it takes until the full Pareto front is covered by the population if the algorithm is not stopped earlier. For that reason, we do not specify a termination criterion.

In our analysis, for simplicity we assume that in each iteration every parent produces one child through bit-wise mutation, i.e., mutating each bit independently with probability $\frac{1}{n}$. Our analysis also holds for uniform selection, where N times a parent is selected independently at random, since also here each individual is selected as a parent once in expectation, and our proofs only rely on the expected number of times a parent is selected. When selecting parents via binary tournaments the expected number of times a parent is selected is at most two (which is the expected number of times it participates in a tournament). This estimate would change the population dynamics by constant factors. For that reason, we are optimistic that our methods apply also to this type of selection, but we do not discuss this question in more detail.

For any generation t of a run of the algorithm, we use P_t to denote the parent population and R_t to denote the combined parent and offspring population.

The ONEJUMPZEROJUMP Benchmark

Let $n \in \mathbb{N}$ and $k = [2..n/4]$. The function $\text{ONEJUMPZEROJUMP}_{n,k} = (f_1, f_2) : \{0,1\}^n \to \mathbb{R}^2$, proposed by Doerr and Zheng (2021), is defined by

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

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

where $|x|_1$ denotes the number of bits of x that are 1 and $|x|_0$ denotes the number of bits of x that are 0. The aim is to maximize both f_1 and f_2, two multimodal objectives. The first objective is the classical $\text{JUMP}_{n,k}$ function. It has a valley of low fitness around its optimum, which can be crossed only by flipping the k correct bits if no solutions of lower fitness are accepted. The second objective is isomorphic to the first, with the roles of zeroes and ones exchanged.

According to Theorem 2 of Doerr and Zheng (2021), the Pareto set of this benchmark is $S^* = \{x \in \{0,1\}^n \mid |x|_1 = [k..n-k] \cup \{0, n\}\}$, and the Pareto front $F^* = f(S^*)$ is $\{(a, 2k+n-a) \mid a \in [2k..n] \cup \{k, n+k\}\}$, making the size of the front $n - 2k + 3$. We define the inner part of the Pareto set by $S_I^* = \{x \mid |x|_1 \in [k..n-k]\}$, and the inner part of the Pareto front by $F_I^* = f(S_I^*) = \{(a, 2k+n-a) \mid a \in [2k..n]\}$. Doerr and Qu (2022a) showed that when using a population of size $N \ge 4(n-2k+3)$ to optimize this benchmark, the NSGA-II algorithm never loses a Pareto-optimal solution once found. Moreover, $O(n^k)$ iterations are needed in expectation.

The ONEMINMAX Benchmark

Let $n \in \mathbb{N}$. The function $\text{ONEMINMAX} = (f_1, f_2) : \{0,1\}^n \to \mathbb{R}$, proposed by Giel and Lehre (2010), is defined by

$$f(x) = (f_1(x), f_2(x)) = \left(n - \sum_{i=1}^{n} x_i, \sum_{i=1}^{n} x_i\right).$$

The aim is to maximize both objectives.

For this benchmark, any solution is Pareto-optimal and the Pareto front $F^* = \{(0,n), (1, n-1), \dots, (n, 0)\}$. Hence $|F^*| = n + 1$. Zheng, Liu, and Doerr (2022) showed that when using a population of size $N \ge 4(n+1)$ to optimize the benchmark, the NSGA-II algorithm never loses a Pareto-optimal solution once found. Moreover, in expectation $O(n \log n)$ iterations are needed.

Lower Bound on the Runtime of the NSGA-II on ONEJUMPZEROJUMP

In this section, we prove a lower bound on the runtime of the NSGA-II algorithm on the ONEJUMPZEROJUMP benchmark.[1] We use $X^i_{P_t}$ to denote the number of individuals with $n-k-i$ 1-bits in P_t and $X^i_{R_t}$ to denote that in R_t.

We first show that with probability arbitrarily close to 1, we have that $P_t \subseteq S^*$ for any t so that the analyses that follow do not need to consider gap individuals (those with between 1 and $k-1$ zeroes or ones) as parents.

[1]For reasons of space, some proofs had to be omitted in this extended abstract. They can be found in the preprint (Doerr and Qu 2022b).

Lemma 1. *Consider the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark for* $2 \leq k \leq \frac{n}{4}$ *with population size* $N = c(n-2k+3)$ *for* $c = o(n)$*. The probability that* $P_0 \subseteq S_I^*$ *is* $1 - o(1)$*. Moreover, if* $P_0 \subseteq S_I^*$*, then for any generation* t*, we have* $P_t \subseteq S^*$*.*

Now we give an upper bound on the probability for any individual to obtain more 1-bits through bit-wise mutation.

Lemma 2. *Let* $n, u, v \in \mathbb{N}$ *with* $n \geq 2$*,* $u, v \geq 1$*, and* $u + v \leq n$*. Suppose* $x \in \{0,1\}^n$ *and* $|x|_1 \leq v$*. Denote the result of applying bit-wise mutation to* x *by* x'*. Then*

$$\Pr[|x'|_1 = u + v] \leq \left(\frac{n-v}{n}\right)^u.$$

Corollary 3. *Let* $n \in \mathbb{N}$ *with* $n \geq 2$ *and let* $v \in [1..n]$*. Suppose* $x \in \{0,1\}^n$ *and* $|x|_1 \leq v$*. Denote the result of applying bit-wise mutation to* x *by* x'*. Then* $\Pr[|x'|_1 = v \wedge x' \neq x] \leq \frac{n-v+1}{n}$*.*

Since we already know from Doerr and Qu (2022a) that, for any objective value on the Pareto front, there are at most 4 individuals with that objective value and positive crowding distance, and they all survive to the generation that follows, to further understand the population dynamics on the front, it is crucial to analyze what happens to the individuals with zero crowding distance. In the following lemma, we show that their survival probability is less than $\frac{1}{2} + o(1)$.

Lemma 4. *Consider the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size* $N = c(n-2k+3)$ *for some* $c \geq 4$ *such that* $ck^2 = o(n)$*. Consider a generation* t *of a run of the algorithm where* $P_0 \subseteq S_I^*$*. Suppose* $\mathbb{E}[X_{P_t}^0] = O(ck)$ *and* $\mathbb{E}[X_{P_t}^{n-2k}] = O(ck)$*. For a rank-1 individual* $x \in R_t$ *that has zero crowding distance, the probability that* $x \in P_{t+1}$ *is less than* $\frac{1}{2} + o(1)$*.*

Proof. Let $F_{>1}$ denote the individuals in R_t with ranks greater than 1. Since $P_0 \subseteq S_I^*$, by Lemma 1, $P_t \subseteq S^*$. Then all the individuals in $F_{>1}$ are created through mutation of individuals in P_t. By Lemma 2, for an individual with less than $n-k$ bits of 1 to create an individual with more than $n-k$ bits of 1, the probability is at most $(\frac{k+1}{n})^2$, and for an individual with $n-k$ bits of 1 to create an individual with more than $n-k$ bits of 1, the probability is at most $\frac{k}{n}$. Symmetrically, for an individual with less than $n-k$ bits of 0 to create an individual with more than $n-k$ bits of 0, the probability is at most $(\frac{k+1}{n})^2$, and for an individual with $n-k$ bits of 0 to create an individual with more than $n-k$ bits of 0, the probability is at most $\frac{k}{n}$. Therefore $\mathbb{E}[|F_{>1}|] \leq (\frac{k+1}{n})^2 c(n-2k+3) + \frac{k}{n}\mathbb{E}[X_{P_t}^0] + \frac{k}{n}\mathbb{E}[X_{P_t}^{n-2k}] = o(1)$ for $ck^2 = o(n)$, $\mathbb{E}[X_{P_t}^0] = O(ck)$ and $\mathbb{E}[X_{P_t}^{n-2k}] = O(ck)$. By Markov's inequality, $\Pr(|F_{>1}| \geq 1) \leq \frac{\mathbb{E}[|F_{>1}|]}{1} = o(1)$.

Let F_1^* the rank-1 individuals in R_t with positive crowding distances. So $|R_t| = 2N$ and there are $2N - |F_1^*| - |F_{>1}|$ individuals in R_t with rank 1 and zero crowding distance. Since $N = c(n-2k+3)$, for some $c \geq 4$, by Lemma 1 of Doerr and Qu (2022a), all individuals in F_1^* will survive. So among the rank-1 individuals with zero crowding distance, $N - |F_1^*|$ survive to the next generation if $|F_1^*| \leq N$. Hence the probability that a rank-1 individual x with zero crowding distance survives is

$$\frac{N - |F_1^*|}{2N - |F_1^*|} \Pr[|F_{>1}| = 0] + \Pr[x \text{ survives} \mid |F_{>1}| \geq 1] \Pr[|F_{>1}| \geq 1].$$

Since $\Pr[|F_{>1}| = 0] \leq 1$ and $\Pr[x \text{ survives} \mid |F_{>1}| \geq 1] \leq 1$, we have that the probability that x survives is at most $\frac{1}{2} + o(1)$. □

Corollary 5. *Consider the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size* $N = c(n-2k+3)$ *for some* $c \geq 4$ *such that* $ck^2 = o(n)$*. Consider a generation* t *of a run of the algorithm where* $P_0 \subseteq S_I^*$*. Suppose* $\mathbb{E}[X_{P_t}^0] = O(ck)$ *and* $\mathbb{E}[X_{P_t}^{n-2k}] = O(ck)$*. Then for any* $i \in [0..n-2k]$*, we have* $\mathbb{E}[X_{P_{t+1}}^i] \leq (\frac{1}{2} + o(1))\mathbb{E}[X_{R_t}^i] + 2$*.*

Now, we can start to estimate $\mathbb{E}[X_{P_t}^i]$ for $i \in [0..n-2k]$.

Lemma 6. *Consider the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size* $N = c(n-2k+3)$ *for some* $c \geq 4$ *such that* $ck^2 = o(n)$*. Suppose* $P_0 \subseteq S_I^*$ *and* $i \in [0..n-2k]$*. Then if* $1^n \notin P_t$*, we have* $\mathbb{E}[X_{P_t}^i] \leq c_i$ *for* $c_i = \frac{e}{e-1}(c(k+i+1) + \sum_{j=0}^{i-1} c_j + 4) + o(ck)$*. Similarly, if* $0^n \notin P_t$*, we have* $\mathbb{E}[X_{P_t}^{n-2k-i}] \leq c_{n-2k-i}$ *for* $c_{n-2k-i} = \frac{e}{e-1}(c(k+i+1) + \sum_{j=0}^{i-1} c_{n-2k-j} + 4) + o(ck)$*.*

Proof. We prove the result for the case where the all-ones string has not been found since the other case is symmetrical.

Let Y^i denote the number of individuals with $n-k-i$ 1-bits in P_t for which no bits are flipped during mutation, and let Z^i denote the number of individuals in P_t for which a positive number of bits are flipped and the resulting children have $n-k-i$ 1-bits. We first prove by induction that $\mathbb{E}[X_{P_t}^0] \leq c_0$ for $c_0 = \frac{e}{e-1}(c(k+1)+4) + o(ck) = O(ck)$.

For the base case, consider the random initialization of P_0. The probability that exactly k among n bits are 0 is less than the probability that at most $k < \frac{n}{4}$ bits are 0, which is at most $e^{-\frac{n}{8}}$. Hence, $\mathbb{E}[X_{P_0}^0] < c(n-2k+3)e^{-\frac{n}{8}} < cne^{-\frac{n}{8}} < 3c < c_0$.

For the induction, assume by the induction hypothesis that $\mathbb{E}[X_{P_t}^0] \leq c_0$ and we will show that $\mathbb{E}[X_{P_{t+1}}^0] \leq c_0$. Clearly, $\mathbb{E}[X_{R_t}^0] = \mathbb{E}[X_{P_t}^0] + \mathbb{E}[Y^0] + \mathbb{E}[Z^0]$. By the induction hypothesis $\mathbb{E}[X_{P_t}^0] \leq c_0$. By definition, $\mathbb{E}[Y^0] = \mathbb{E}[(1-\frac{1}{n})^n X_{P_t}^0] \leq \frac{1}{e}\mathbb{E}[X_{P_t}^0] = \frac{c_0}{e}$. Since $1^n \notin P_t$ and $P_0 \subseteq S_I^*$, by Lemma 1, there is no individual with more than $n-k$ 1-bits. Then by Corollary 3, for any individual to have a positive number of bits flipped and produce an individual with $n-k$ 1-bits, the probability is at most $\frac{k+1}{n}$. So $\mathbb{E}[Z^0] \leq c(n-2k+3)\frac{k+1}{n} \leq c(k+1)$. Together, $\mathbb{E}[X_{R_t}^0] \leq (1+\frac{1}{e})c_0 + c(k+1)$. Then by Corollary 5, $\mathbb{E}[X_{P_{t+1}}^0] \leq (\frac{1}{2}+o(1))((1+\frac{1}{e})c_0 + c(k+1)) + 2 = \frac{e}{e-1}(c(k+1)+4) + o(ck) = c_0$.

The same arguments can be applied to estimate the expected number of individuals with $n-k-1$ 1-bits. We have $\mathbb{E}[X^1_{R_t}] = \mathbb{E}[X^1_{P_t}] + \mathbb{E}[Y^1] + \mathbb{E}[Z^1]$. We assume by the induction hypothesis $\mathbb{E}[X^1_{P_t}] \le c_1$ for $c_1 = \frac{e}{e-1}(c(k+2) + c_0 + 4) + o(ck)$. Similarly as before, $\mathbb{E}[Y^1] \le \frac{c_1}{e}$. Moreover, there are no individuals with more than $n-k$ bits of 1 by Lemma 1. So to bound $\mathbb{E}[Z^1]$, consider separately the cases where i) the parent has at most $n-k-1$ 1-bits and ii) the parent has $n-k$ 1-bits. By Corollary 3, for case i), the probability that the child has $n-k-1$ 1-bits is at most $\frac{k+2}{n}$. We trivially bound the probability for case ii) by 1. Therefore, $\mathbb{E}[Z^1] \le c(n-2k+3)\frac{k+2}{n} + c_0 \le c(k+2) + c_0$. Then $\mathbb{E}[X^1_{R_t}] \le (1+\frac{1}{e})c_1 + c(k+2) + c_0$. Hence, by Corollary 5,

$$\begin{aligned}\mathbb{E}[X^1_{P_{t+1}}] &\le (\tfrac{1}{2} + o(1))((1+\tfrac{1}{e})c_1 + c(k+2) + c_0) + 2 \\ &= \tfrac{e}{e-1}(c(k+2) + c_0 + 4) + o(ck) = c_1.\end{aligned}$$

Continuing this way and letting c_i denote the upper bound on the expected number of individuals with $n-k-i$ number of 1-bits for $0 \le i \le n-2k$, we have

$$c_i = \frac{e}{e-1}\left(c(k+i+1) + \sum_{j=0}^{i-1} c_j + 4\right) + o(ck). \quad \square$$

With the bound on $\mathbb{E}[X^1_{P_t}]$ found in Lemma 6, we can now prove a sharper bound on $\mathbb{E}[X^0_{P_t}]$.

Corollary 7. *Consider a generation t of the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size $N = c(n-2k+3)$ for some $c \ge 4$ such that $ck^2 = o(n)$. Suppose $P_0 \subseteq S^*_I$. If $1^n \notin P_t$, then $\mathbb{E}[X^0_t] \le \frac{4e}{e-1} + o(1)$. Similarly, if $0^n \notin P_t$, then $\mathbb{E}[X^{n-2k}_t] \le \frac{4e}{e-1} + o(1)$.*

Now with the upper bounds on $\mathbb{E}[X^0_{P_t}]$, $\mathbb{E}[X^1_{P_t}]$, $\mathbb{E}[X^{n-2k}_{P_t}]$, and $\mathbb{E}[X^{n-2k-1}_{P_t}]$, we can prove a lower bound on the runtime.

Theorem 8. *Consider the NSGA-II algorithm optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size $N = c(n-2k+3)$, for some $c \ge 4$ such that $ck^2 = o(n)$. Then the number of fitness evaluations needed in expectation is at least $\frac{3}{2}(\frac{4}{e-1} + o(1))^{-1}Nn^k$.*

Precise Runtime of the NSGA-II with Fixed Sorting on ONEJUMPZEROJUMP

In the version of the NSGA-II considered in Doerr and Qu (2022a) and the previous section, when the crowding distance is being calculated with respect to each objective, we sort the individuals such that the ones with the same objective value are positioned randomly. A variant of the algorithm, considered in Bian and Qian (2022), is to fix the relative positions of the individuals that have the same objective value. We call this variant of the algorithm the NSGA-II with fixed sorting, and show a precise bound on the runtime of this variant optimizing the ONEJUMPZEROJUMP benchmark.

First we observe in the following lemma that for this variant of the algorithm, after $O(n \log n)$ iterations, for each objective value on F^*_I there are exactly two individuals with positive crowding distances.

Lemma 9. *Consider the NSGA-II algorithm with fixed sorting optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with population size $N = c(n-2k+3)$ for some $c \ge 2$. After $O(n \log n)$ iterations, for any generation t, for every objective value $v \in F^*_I$, there are exactly two individuals $x, y \in R_t$ such that $f(x) = f(y) = v$, $\mathrm{cDis}(x) > 0$ and $\mathrm{cDis}(y) > 0$.*

We call the phase where, for every objective value $v \in F^*_I$, there are exactly two individuals x, y in the combined population such that $f(x) = f(y) = v$, $\mathrm{cDis}(x) > 0$ and $\mathrm{cDis}(y) > 0$, the tightening phase. In the following analyses, we define $s^* = O(n \log n)$ to be the generation where the algorithm first enters the tightening phase. Then by Lemma 9, for any generation $t \ge s^*$, the algorithm stays in the tightening phase. In the following Lemma, we estimate similarly to Lemma 4 the probability that a rank-1 individual with zero crowding distance survives. What is different now is that for the tightening phase, we can calculate the probability precisely (apart from lower-order terms).

Lemma 10. *Consider a generation $t \ge s^*$ of the NSGA-II algorithm with fixed sorting optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with population size $N = c(n-2k+3)$ for some $c \ge 2$ such that $ck^2 = o(n)$. Suppose $P_0 \subseteq S^*_I$, $\mathbb{E}[X^0_{P_t}] = O(ck)$, and $\mathbb{E}[X^{n-2k}_{P_t}] = O(ck)$. For a rank-1 individual $x \in R_t$ that has zero crowding distance, the probability that $x \in P_{t+1}$ is $\frac{c-2}{2c-2} \pm o(1)$.*

Corollary 11. *Consider a generation $t \ge s^*$ of the NSGA-II algorithm with fixed sorting optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with population size $N = c(n-2k+3)$ for some $c \ge 2$ such that $ck^2 = o(n)$. Suppose $P_0 \subseteq S^*_I$, $\mathbb{E}[X^0_{P_t}] = O(ck)$, and $\mathbb{E}[X^{n-2k}_{P_t}] = O(ck)$. Then for any $i \in [0..n-2k]$, we have $\mathbb{E}[X^i_{P_{t+1}}] = (\frac{c-2}{2c-2} + o(1))\mathbb{E}[X^i_{R_t}] + \frac{c}{c-1} - o(1)$.*

Consequently, we can calculate $\mathbb{E}[X^0_{P_t}]$ and $\mathbb{E}[X^{n-2k}_{P_t}]$ precisely apart from lower order terms.

Lemma 12. *Consider the NSGA-II algorithm with fixed sorting optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark with the population size $N = c(n-2k+3)$ for some $c \ge 2$ such that $ck^2 = o(n)$. Suppose $P_0 \subseteq S^*_I$. We have for any $t \ge s^* + \log n$, if $1^n \notin P_t$, then $\mathbb{E}[X^0_{P_t}] = \frac{2ec}{ec-c+2} \pm o(1)$. Similarly, if $0^n \notin P_t$, then $\mathbb{E}[X^{n-2k}_{P_t}] = \frac{2ec}{ec-c+2} \pm o(1)$.*

Theorem 13. *Consider the NSGA-II algorithm with fixed sorting optimizing the* ONEJUMPZEROJUMP$_{n,k}$ *benchmark, for $k \ge 3$, with the population size $N = c(n-2k+3)$, for some $c \ge 2$ such that $ck^2 = o(n)$. Then the number of fitness evaluations needed in expectation is $\frac{3}{2}N(\frac{2c}{ec-c+2} \pm o(1))^{-1}n^k$.*

Lower Bound on the Runtime of the NSGA-II on ONEMINMAX

Zheng, Liu, and Doerr (2022) gave an $O(Nn \log n)$ upper bound on the runtime of the NSGA-II optimizing the ONEMINMAX benchmark. In this section, we prove a matching lower bound using the techniques we have developed so far.

In this section, for any $i \in [0..n]$ and any generation t, we let $X^i_{P_t}$ denote the number of individuals with i 0-bits in P_t and let $X^i_{R_t}$ denote that in R_t.

Suppose in an iteration t the individual with the least number of 0-bits in P_t has i_t 0-bits. Then we can think of the algorithm making progress as it tries to decrease i_t till it becomes 0, at which point the algorithm has found 1^n. In the following lemma, we give upper bounds on $\mathbb{E}[X^{i_t}_{P_t}]$ when i_t is close to 0, which will help us estimate the waiting time needed for the algorithm to make progress there.

Lemma 14. *Consider the NSGA-II algorithm optimizing the* OneMinMax *benchmark for $n > 16$, with $N = c(n+1)$ for $c \geq 4$, where for all $x \in P_0$, $|x|_0 \geq \frac{n}{4}$. Suppose $v \in [0..n]$ such that $cv^2 = o(n)$ and for a generation t there is no individual $x \in P_t$ such that $|x|_0 < v$. Then $\mathbb{E}[X^v_{P_t}] \leq c_0 = \frac{4e}{e-1} + o(1)$ and $\mathbb{E}[X^{v+1}_{P_t}] \leq \frac{e-1}{e}(c(v+2) + c_0 + 4) + o(c)$.*

Then, we show that only with very small probability, $o(n^{-\frac{4}{3}})$, the algorithm can make a progress of a size larger than 1 in an iteration.

Lemma 15. *Consider the NSGA-II algorithm optimizing the* OneMinMax *benchmark for $n > 16$, with $N = c(n+1)$ for $c \geq 4$, where for all $x \in P_0$, $|x|_0 \geq \frac{n}{4}$. Suppose $v \in [0..n]$ such that $cv^3 = o(n)$ and for a generation t, there is no individual $x \in P_t$ such that $|x|_0 < v$. Suppose in R_t the individual with the least number of 0-bits is y. Then $\Pr[|y|_0 \leq v - 2] = o(n^{-\frac{4}{3}})$.*

Finally, we combine everything to obtain a lower bound on the runtime.

Theorem 16. *Consider the NSGA-II algorithm optimizing the* OneMinMax *benchmark for $n > 16$, with $N = c(n+1)$ for $c \geq 4$ and $c = o(n^\mu)$ for $\mu < 1$. Then the number of fitness evaluations needed is at least $N(n(\frac{4e}{e-1} + o(1))^{-1}\frac{1-\mu}{3}\ln n)$.*

Experiments

To complement our theoretical results, we also experimentally evaluate some runs of the NSGA-II on the OneJumpZeroJump benchmark, both with respect to the runtime and the population dynamics. We note that Doerr and Qu (2022a) already presented some results on the runtime (for $k = 3$, $N/(2n - k + 3) = 2, 4, 8$, and $n = 20, 30$). We therefore mostly concentrate on the population dynamics, i.e., the number of individuals in the population for each objective value on the Pareto front, which our theoretical analyses have shown to be crucial for determining the lower bound and the leading coefficient of the runtime.

Settings

We implemented the algorithm as described in the Preliminaries section in Python, and tested the following settings.

- Problem size n: 50 and 100.
- Jump size k: 2. This small number was necessary to admit the problem sizes above. Problem sizes of a certain magnitude are needed to see a behavior not dominated by lower-order effects.
- Population size N: $2(n - 2k + 3)$, $4(n - 2k + 3)$, and $8(n - 2k + 3)$. Doerr and Qu (2022a) suggested that, even though their mathematical analysis applies only for $N \geq 4(n - 2k + 3)$, already for $N = 2(n - 2k + 3)$ the algorithm still succeeds empirically. Therefore, we have also experimented with $N = 2(n-2k+3)$ to confirm that our arguments for the population dynamics still apply to the smaller population size.
- Selection for variation: fair selection.
- Mutation method: bit-wise mutation with rate $\frac{1}{n}$.
- Number of independent repetitions per setting: 30.

	$n = 50$	$n = 100$
$N = 2(n - 2k + 3)$	390,506	3,068,980
$N = 4(n - 2k + 3)$	617,606	4,514,578
$N = 8(n - 2k + 3)$	919,142	5,572,427

Table 1: Average runtime of the NSGA-II with bit-wise mutation on the OneJumpZeroJump benchmark with $k = 2$.

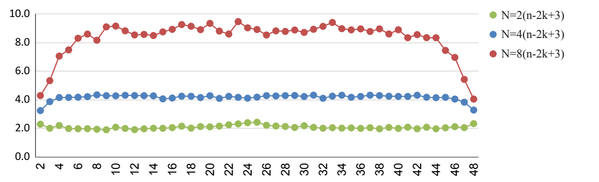

Figure 1: Average number of individuals with $i \in [k..n-k]$ 1-bits for $n = 50$ and $k = 2$.

Results on the Runtime

Table 1 contains the average runtime (number of fitness evaluations done until the full Pareto front is covered) of the algorithm. For all of the settings, we have observed a standard deviation that is between 50% to 80% of the mean, which supports our reasoning that the runtime is dominated by the waiting time needed to find the two extremal points of the front, which is the maximum of two geometric random variables. An obvious observation from the data is that increasing N does not help with the runtime, supporting our theoretical results that the lower bound on the runtime increases when N increases. Moreover, for all the settings that we have experimented with, the average runtime is well above our theoretically proven lower bound made tighter by discarding the lower order terms, namely $\frac{3}{2}(\frac{4}{e-1})^{-1}Nn^k$.

Results on the Population Dynamics

For all of the experiments conducted, we have also recorded the population dynamics throughout the executions of the algorithm. Specifically, for each run, for every $n^k/50$ iterations, we record for each $i \in [k..n-k]$ how many individuals there are in the parent population with i bits of 1. Since as shown in our theoretical analyses and Doerr and Qu (2022a), the greatest contributor to the runtime is the waiting time to find the all-ones and the all-zeroes strings after the inner part

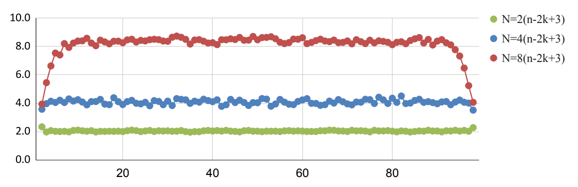

Figure 2: Average number of individuals with $i \in [k..n-k]$ 1-bits for $n = 100$ and $k = 2$.

of the Pareto front has been discovered, we are mostly interested in how the population dynamics develop in that phase. To this end, we discard data points recorded when the inner part of the Pareto front has not been fully covered, and those recorded after one of the extremal points has already been discovered. The final number reported for a run is the average of the data points kept. In the end we report the average of the means across 30 repetitions. In all of the runs, we have never observed an initial population not contained in the inner part of the Pareto front, supporting our theoretical arguments and also making the experiments fall into the scenario that we have studied theoretically.

Figure 1 contains the average number of individuals throughout a run of the algorithm for each point on the inner part of the Pareto front for $n = 50$, averaged by the 10 repetitions, and Figure 2 contains that for $n = 100$. An obvious observation is that for all experiment settings, we have that the average number of individuals with k or $n-k$ 1-bits is less than the proven upper bound $\frac{4e}{e-1} \approx 6.33$. When doubling the population size, the number of individuals with k or $n-k$ 1-bits grows. This does not contradict with our upper bound (which is independent of the population size), but it only suggests that the precise average occupation of these objective values contains a dependence on the population size that is small enough for this number to be bounded by $\frac{4e}{e-1} \approx 6.33$. We note that for the setting with fixed sorting the precise occupation number $\frac{2ec}{2ec-c+2} \pm o(1)$ we proved displayed exactly such a behavior.

Our experimental data also give the occupation numbers for the other objective values. We did not discuss these in much detail in our theoretical analysis since all we needed to know was the occupation number for the outermost points of the inner part of the Pareto front and a relatively generous upper bound for the points one step closer to the middle. A closer look into our mathematical analysis shows that it does give good estimates only for objective values close to the outermost points of the Pareto front. For that reason, it is interesting to observe that our experimental data show that the population is, apart from few positions close to the outermost positions, very evenly distributed on the Pareto front (that is, a typical position is occupied by c individuals, where c is such that the population size is $N = c(n-2k+3)$). Given the mostly random selection of most of the next population (apart from the up to 4 individuals with positive crowding distance per position) and the drift toward the middle in the offspring generation (e.g., a parent with $\frac{3}{4}n$ ones is much more likely to generate an offspring with fewer than more ones), this balanced distribution was a surprise to us. While it has no influence on the time to find the Pareto front of ONEJUMPZEROJUMP, we suspect that such balanced distributions are preferable for many other problems.

Conclusions and Future Works

In this work, we gave the first lower bounds matching previously proven upper bounds for the runtime of the NSGA-II. We proved that the runtime of the NSGA-II with population size at least four times the Pareto front size computes the full Pareto front of the ONEMINMAX problem in expected time (number of function evaluations) $\Omega(Nn\log n)$ and the one of the ONEJUMPZEROJUMP problem with jump size k in expected time $\Omega(Nn^k)$. These bounds match the corresponding $O(Nn\log n)$ and $O(Nn^k)$ upper bounds shown respectively by Zheng, Liu, and Doerr (2022) and by Doerr and Qu (2022a). These asymptotically tight runtimes show that, different from many other population-based search heuristics, the NSGA-II does not profit from larger population sizes, even in an implementation where the expected numbers $\Theta(n\log n)$ and $\Theta(n^k)$ of iterations is the more appropriate performance criterion. Together with the previous result that a population size below a certain value leads to a detrimental performance of the NSGA-II (Zheng, Liu, and Doerr 2022), our results show that the right choice of the population size of the NSGA-II is important for an optimal performance, much more than for many single-objective population-based algorithms, where larger population sizes at least for certain parameter ranges have little influence on the number of fitness evaluations needed.

The main obstacle we had to overcome in our analysis was to understand sufficiently well the population dynamics of the NSGA-II, that is, the expected number of individuals having a particular objective value at a particular time. While we have not completely understood this question, our estimates are strong enough to obtain, for the ONEJUMPZEROJUMP benchmark and the NSGA-II using a fixed sorting to determine the crowding distance, a runtime guarantee that is also tight including the leading constant.

From this work, a number of possible continuations exist. For example, runtime analyses which are tight including the leading constant allow one to distinguish constant-factor performance differences. This can be used to optimize parameters or decide between different operators. For example, we have used the mutation rate $\frac{1}{n}$, which is the most accepted choice for bit-wise mutation. By conducting our analysis for a general mutation rate $\frac{\alpha}{n}$, one would learn how the mutation rate influences the runtime and one would be able to determine an optimal value for this parameter. We note that a different mutation rate not only changes the probability to reach the global optimum from the local one (which is well-understood (Doerr et al. 2017)), but also changes the population dynamics. We are nevertheless optimistic that our methods can be extended in such directions.

Acknowledgments

This work was supported by a public grant as part of the Investissements d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

References

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

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.

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

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.

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.

Doerr, B.; Gao, W.; and Neumann, F. 2016. Runtime analysis of evolutionary diversity maximization for OneMinMax. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, 557–564. ACM.

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.; Le, H. P.; Makhmara, R.; and Nguyen, T. D. 2017. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, 777–784. ACM.

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. 2022a. A first runtime analysis of the NSGA-II on a multimodal problem. In *Parallel Problem Solving From Nature, PPSN 2022*, 399–412. Springer.

Doerr, B.; and Qu, Z. 2022b. From Understanding the Population Dynamics of the NSGA-II to the First Proven Lower Bounds. *CoRR*, abs/2209.13974.

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.

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.; and Zhou, Y. 2020. Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In *Conference on Artificial Intelligence, AAAI 2020*, 2359–2366. AAAI Press.

Huang, Z.; Zhou, Y.; Chen, Z.; and He, X. 2019. Running time analysis of MOEA/D with crossover on discrete optimization problem. In *Conference on Artificial Intelligence, AAAI 2019*, 2296–2303. AAAI Press.

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. ISBN 978-3-642-17338-7.

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.

Qian, C.; Liu, D.; and Zhou, Z. 2022. Result diversification by multi-objective evolutionary algorithms with theoretical guarantees. *Artificial Intelligence*, 309: 103737.

Qian, C.; Yu, Y.; Tang, K.; Yao, X.; and Zhou, Z. 2019. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence*, 275: 279–294.

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

Zheng, W.; and Doerr, B. 2022. Better approximation guarantees for the NSGA-II by using the current crowding distance. In *Genetic and Evolutionary Computation Conference, GECCO 2022*, 611–619. ACM.

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.