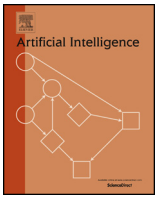
Artificial Intelligence 325 (2023) 104016

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II) ☆

Weijie Zheng [a], Benjamin Doerr [b,*]

[a] School of Computer Science and Technology, International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen, China
[b] Laboratoire d'Informatique (LIX), École Polytechnique, CNRS, Institute Polytechnique de Paris, Palaiseau, France

ARTICLE INFO

Article history:
Received 27 June 2022
Received in revised form 17 September 2023
Accepted 18 September 2023
Available online 22 September 2023

Keywords:
NSGA-II
Runtime analysis
Multi-objective optimization
Theory of computing

ABSTRACT

The non-dominated sorting genetic algorithm II (NSGA-II) is the most intensively used multi-objective evolutionary algorithm (MOEA) in real-world applications. However, in contrast to several simple MOEAs analyzed also via mathematical means, no such study exists for the NSGA-II so far. In this work, we show that mathematical runtime analyses are feasible also for the NSGA-II. As particular results, we prove that with a population size four times larger than the size of the Pareto front, the NSGA-II with two classic mutation operators and four different ways to select the parents satisfies the same asymptotic runtime guarantees as the SEMO and GSEMO algorithms on the basic OneMinMax and LeadingOnesTrailingZeroes benchmarks. However, if the population size is only equal to the size of the Pareto front, then the NSGA-II cannot efficiently compute the full Pareto front: for an exponential number of iterations, the population will always miss a constant fraction of the Pareto front. Our experiments confirm the above findings.

1. Introduction

Many real-world problems need to optimize multiple conflicting objectives simultaneously, see [58] for a discussion of the different areas in which such problems arise. Instead of computing a single good solution, a common approach to such multi-objective optimization problems is to compute a set of interesting solutions so that a decision maker can select the most desirable one from these. Multi-objective evolutionary algorithms (MOEAs) are a natural choice for such problems due to their population-based nature. Such multi-objective evolutionary algorithms (MOEAs) have been successfully used in many real-world applications [58].

Unfortunately, the theoretical understanding of MOEAs falls far behind their success in practice, and this discrepancy is even larger than in single-objective evolutionary computation, where the last twenty years have seen some noteworthy progress on the theory side [44,1,33,16]. After some early theoretical works on convergence properties, e.g., [52], the first

☆ Extended version of a paper that appeared at AAAI 2022 [57], and that was conducted when the first author was with Southern University of Science and Technology. This version contains all proofs, many of them revised and improved. In particular, the runtime result for the NSGA-II with tournament selection on LeadingOnesTrailingZeroes now holds for $N \geq 4(n+1)$ instead of $N \geq 5(n+1)$. In addition, in all upper bounds we now also regard binary tournament selection as in Deb's implementation of the NSGA-II (building the N tournaments from two permutations of the population). We added tail bounds for the runtime guarantees. The experimental section was extended as well.

* Corresponding author.
E-mail addresses: zhengweijie@hit.edu.cn, zhengwj13@tsinghua.org.cn (W. Zheng).

https://doi.org/10.1016/j.artint.2023.104016

mathematical runtime analysis of an MOEA was conducted by Laumanns et al. [35,36]. They analyzed the runtime of the *simple evolutionary multi-objective optimizer (SEMO)*, a multi-objective counterpart of the *randomized local search* heuristic, on the CountingOnesCountingZeroes and LeadingOnesTrailingZeroes benchmarks, which are bi-objective analogues of the classic (single-objective) OneMax and LeadingOnes benchmark. Around the same time, Giel [29] analyzed the *global SEMO (GSEMO)*, the multi-objective counterpart of the $(1+1)$ EA, on the LeadingOnesTrailingZeroes function.

Subsequent theoretical works majorly focused on variants of these algorithms and analyzed their runtime on the CountingOnesCountingZeroes and LeadingOnesTrailingZeroes benchmarks, on variants of them, on new benchmarks, and on combinatorial optimization problems [49,6,51,46,3,25]. We note that the (G)SEMO algorithm keeps all non-dominated solutions in the population and discards all others, which can lead to impractically large population sizes. There are three theory works [2,41,15] on the runtime of a simple hypervolume-based MOEA called *$(\mu+1)$ simple indicator-based evolutionary algorithm* ($(\mu+1)$-SIBEA), regarding both classic benchmarks and problems designed to highlight particular strengths and weaknesses of this algorithm. As the SEMO and GSEMO, the $(\mu+1)$ SIBEA also creates a single offspring per generation; different from the former, it works with a fixed population size μ.

Recently, also decomposition-based multi-objective evolutionary algorithms were analyzed [37,32,31]. These algorithms decompose the multi-objective problem into several related single-objective problems and then solve the single-objective problems in a co-evolutionary manner. This direction is fundamentally different from the above works and our research. Since not primarily focused on multi-objective optimization, we also do not discuss further the successful line of works that solve constrained single-objective problems by turning the constraint violation into a second objective, see, e.g., [43,26,40,27,50,47,48,10,13,11].

Unfortunately, most of the algorithms discussed in these theoretical works are far from the MOEAs used in practice. As pointed out in the survey [58], the majority of the MOEAs used in research and applications builds on the framework of the *non-dominated sorting genetic algorithm II (NSGA-II)* [21]. This algorithm works with a population of fixed size N. It uses a complete order defined by the non-dominated sorting and the crowding distance to compare individuals. In each generation, N offspring are generated from the parent population and the N best individuals (according to the complete order) are selected as the new parent population. This approach is thus substantially different from the (G)SEMO algorithm and hypervolume-based approaches (and naturally completely different from decomposition-based methods), see the features of these algorithms described in the above two paragraphs.

Both the predominance in practice and the fundamentally different working principles ask for a rigorous understanding of the NSGA-II. However, to the best of our knowledge so far no mathematical runtime analysis for the NSGA-II has appeared.[1] We note that the runtime analysis in [9] considers a (G)SEMO algorithm that uses the crowding distance as one of several diversity measures used in the selection of the single parent creating an offspring, but due to the differences of the basic algorithms, none of the arguments used there appear helpful in the analysis of the NSGA-II.

Our Contributions. This paper conducts the first mathematical runtime analysis of the NSGA-II. We regard the NSGA-II with four different parent selection strategies (choosing each individual as a parent once, choosing parents independently and uniformly at random, and two ways of choosing the parents via binary tournaments) and with two classic mutation operators (one-bit mutation and standard bit-wise mutation), but in this first work without crossover (we remark that crossover is very little understood from the runtime perspective in MOEAs, the only works prior to ours we are aware of are [42,49,32]). As previous theoretical works, we analyze how long the NSGA-II takes to cover the full Pareto front, that is, we estimate the number of iterations until the parent population contains an individual for each objective value of the Pareto front.

When trying to determine the runtime of the NSGA-II, we first note that the selection mechanism of the NSGA-II may remove all individuals with some fixed objective value on the front. In other words, the fact that a certain objective value on the Pareto front was found in some iteration does not mean that this is not lost in some later iteration. This is one of the substantial differences to the (G)SEMO algorithm. We prove that if the population size N is at least four times larger than the size of the Pareto front, then both for the OneMinMax and the LeadingOnesTrailingZeroes benchmarks, such a loss of Pareto front points cannot occur. With this insight, we then show that each of these eight variants of the NSGA-II computes the full Pareto front of the OneMinMax benchmark in an expected number of $O(n \log n)$ iterations (Theorems 2 and 6) and the front of the LeadingOnesTrailingZeroes benchmark in $O(n^2)$ iterations (Theorems 8 and 9). When $N = \Theta(n)$, the corresponding runtime guarantees in terms of fitness evaluations, $O(Nn \log n) = O(n^2 \log n)$ and $O(Nn^2) = O(n^3)$, have the same asymptotic order as those proven previously for the SEMO, GSEMO, and $(\mu+1)$ SIBEA (when $\mu \geq n+1$ and when $\mu = \Theta(n)$ for the $(\mu+1)$ SIBEA). We note that the benchmarks OneMinMax and LeadingOnesTrailingZeroes are the two most intensively studied benchmarks in the runtime analysis of MOEAs. In this first runtime analysis work on the NSGA-II, we therefore concentrated on these two benchmarks to allow a good comparison with the known performance of other MOEAs.

Using a population size larger than the size of the Pareto front is necessary. We prove that if the population size is equal to the size of the Pareto front, then the NSGA-II (applying one-bit mutation once to each parent) regularly loses points on

[1] By mathematical runtime analysis, we mean the question of how many function evaluations a black-box algorithm takes to achieve a certain goal. The computational complexity of the operators used by the NSGA-II, in particular, how to most efficiently implement the non-dominated sorting routine, is a different question (and one that is well-understood [21]).

the Pareto front of OneMinMax. This effect is strong enough so that with high probability for an exponential time each generation of the NSGA-II does not cover a constant fraction of the Pareto front of OneMinMax.

Our short experimental analysis confirms these findings and gives some quantitative estimates for which mathematical analyses are not precise enough. For example, we observe that also with population sizes smaller than what is required for our theoretical analysis (four times the size of the Pareto front), the NSGA-II efficiently covered the Pareto front of the OneMinMax and LeadingOnesTrailingZeroes benchmarks. With suitable population sizes, the NSGA-II beats the GSEMO algorithm on these benchmarks. Complementing our negative result, we observe that the fraction of the Pareto front not covered when using a population size equal to the front size is around 20% for OneMinMax and 40% for LeadingOnesTrailingZeroes. Also without covering the full Pareto front, MOEAs can serve their purpose of proposing to a decision maker a set of interesting solutions. With this perspective, we also regard experimentally the sets of solutions evolved by the NSGA-II when the population size is only equal to the size of the Pareto front. For both benchmarks, we observe that after a moderate runtime, the population contains the two extremal solutions and covers in a very evenly manner the rest of the Pareto front.

Overall, this work shows that the NSGA-II despite its higher complexity (parallel generation of offspring, selection based on non-dominated sorting and crowding distance) admits mathematical runtime analyses in a similar fashion as done before for simpler MOEAs, which hopefully will lead to a deeper understanding of the working principles of this important algorithm.

Subsequent works: We note that the conference version [57] of this work has already inspired a substantial amount of subsequent research. We briefly describe these results now. In [54], the performance of the NSGA-II with small population size was analyzed. The main result is that the problem that Pareto front points can be lost can be significantly reduced with a small modification of the selection procedure that was previously analyzed experimentally [34], namely to remove individuals in the selection of the next population sequentially, recomputing the crowding distance after each removal. For this setting, an $O(n/N)$ approximation guarantee was proven. In [5], the first runtime analysis of the NSGA-II with crossover was conducted, however, no speed-ups from crossover could be shown. Also, significant speed-ups were shown when using larger tournaments than binary tournaments. In [22], the performance of the NSGA-II on the multimodal benchmark OneJumpZeroJump benchmark [25] was analyzed. This work shows that the NSGA-II also on this multimodal benchmark has a performance asymptotically at least as good as the GSEMO algorithm (when the population size is at least four times the size of the Pareto front). A matching lower bound for this and our result on OneMinMax was proven in [23]. This work in particular shows that the NSGA-II in these settings does not profit from population sizes larger than the minimum required population size. Two recent works showed significant performance gains from crossover, one on the OneJumpZeroJump benchmark [24] and one on an artificial problem [20]. The first runtime analysis of the NSGA-II on a combinatorial problem, namely the bi-objective minimum spanning tree problem previously regarded in [39,45], was conducted in [8]. The first runtime analysis of the NSGA-II for noisy optimization appeared in [19]. The first runtime analysis of the SMS-EMOA [4] (a variant of the NSGA-II building on the hyper-volume) was conducted in [7]. All these works regard bi-objective problems. For the OneMinMax problem in three or more objectives, it was shown in [55] that the NSGA-II cannot find the full Pareto front in polynomial time and even has difficulties in approximating it. It was shown in [53] that the NSGA-III does not experience these problems, at least in three objectives. With this recent development, we are confident to claim that our first mathematical runtime analysis for the NSGA-II has started a fruitful direction of research.

The remainder of the paper is organized as follows. The NSGA-II framework is briefly introduced in Section 2. Sections 3 and 4 separately show our runtime results of the NSGA-II with large enough population size on the OneMinMax and LeadingOnesTrailingZeroes functions. Section 5 proves the exponential runtime of the NSGA with population size equal to the size of the Pareto front. Our experimental results are discussed in Section 6. Section 7 concludes this work.

2. Preliminaries

In this section, we give a brief introduction to multi-objective optimization and to the NSGA-II framework. For the simplicity of presentation, we shall concentrate on two objectives, both of which have to be maximized. A bi-objective objective function on some search space Ω is a pair $f = (f_1, f_2)$ where $f_i : \Omega \to \mathbb{R}$. We write $f(x) = (f_1(x), f_2(x))$ for all $x \in \Omega$. We shall always assume that we have a bit-string representation, that is, that $S = \{0,1\}^n$ for some $n \in \mathbb{N}$. The challenge in multi-objective optimization is that usually there is no solution that maximizes both f_1 and f_2 and thus is at least as good as all other solutions.

More precisely, in bi-objective maximization, we say *x weakly dominates y*, denoted by $x \succeq y$, if and only if $f_1(x) \geq f_1(y)$ and $f_2(x) \geq f_2(y)$. We say *x strictly dominates y*, denoted by $x \succ y$, if and only if $f_1(x) \geq f_1(y)$ and $f_2(x) \geq f_2(y)$ and at least one of the inequalities is strict. We say that a solution is *Pareto-optimal* if it is not strictly dominated by any other solution. The set of objective values of all Pareto optima is called the *Pareto front* of f. With this language, the aim in multi-objective optimization is to compute a small set P of Pareto optima such that $f(P) = \{f(x) \mid x \in P\}$ is the Pareto front or is at least a diverse subset of it. Consider an algorithm A optimizing a multi-objective problem f with Pareto front M. Let P_t be the population at iteration t and G_t be the number of function evaluations till iteration t, then the *time complexity* or *running time* in this paper is the random variable $T_A(f) = \inf\{G_t \mid f(P_t) \supseteq M\}$. Usually, we discuss the expected runtime or the runtime with some probability.

The NSGA-II

When working with a fixed population size, an MOEA must select the next parent population from the combined parent and offspring population by discarding some of these individuals. For this, a complete order on the combined parent and offspring population could be used so that the next parent population is taken in a greedy manner according to this order. Since dominance is only a partial order, the NSGA-II [21] extends the dominance relation to the following complete order.

In a given population $P \subseteq \{0,1\}^n$, each individual x has both a rank and a crowding distance. The ranks are defined recursively based on the dominance relation. All individuals that are not strictly dominated by another one have rank one. Given that the ranks $1, \dots, k$ are already defined, the individuals of rank $k+1$ are those among the remaining individuals that are not strictly dominated except by individuals of rank k or smaller. This defines a partition of P into sets $F_1, F_2, \dots$ such that F_i contains all individuals with rank i. As shown in [21], this partition can be computed more efficiently than what the above recursive description suggests, namely in quadratic time, see Algorithm 1 for details. It is clear that individuals with lower rank are more interesting, so when comparing two individuals of different ranks, the one with lower rank is preferred.

To compare individuals in the same rank class F_i, the crowding distance of these individuals (in F_i) is computed, and the individual with larger distance is preferred. Ties are broken randomly.

Algorithm 1 Fast-non-dominated-sort(S).

Input: $S = \{S_1, \dots, S_{|S|}\}$: the set of individuals
Output: $F_1, F_2, \dots$
for $i = 1, \dots, |S|$ **do**
  $\mathrm{ND}(S_i) = 0$ % *number of individuals strictly dominating* S_i
  $\mathrm{SD}(S_i) = \emptyset$ % *set of individuals strictly dominated by* S_i
end for
for $i = 1, \dots, |S|$ **do** % *compute* ND *and* SD
  for $j = 1, \dots, |S|$ **do**
    if $S_i \prec S_j$ **then**
      $\mathrm{ND}(S_i) = \mathrm{ND}(S_i) + 1$
      $\mathrm{SD}(S_j) = \mathrm{SD}(S_j) \cup \{S_i\}$
    end if
  end for
end for
$F_1 = \{S_i \mid \mathrm{ND}(S_i) = 0, i = 1, 2, \dots, |S|\}$
$k = 1$
while $F_k \neq \emptyset$ **do**
  $F_{k+1} = \emptyset$
  for any $s \in F_k$ **do** % *discount* F_k *from* ND *and* SD
    for any $s' \in \mathrm{SD}(s)$ **do**
      $\mathrm{ND}(s') = \mathrm{ND}(s') - 1$
      if $\mathrm{ND}(s') = 0$ **then**
        $F_{k+1} = F_{k+1} \cup \{s'\}$
      end if
    end for
  end for
  $k = k + 1$
end while

Algorithm 2 shows how the crowding distance in a given set S is computed. The crowding distance of some $x \in S$ is the sum of the crowding distances x has with respect to each objective function f_i. For a given f_i, the individuals in S are sorted in order of ascending f_i value (for equal values, a tie-breaking mechanism is needed, but we shall not make any assumption on this, that is, our mathematical results are valid regardless of how these ties are broken). The first individual and the last individual in the sorted list have an infinite crowding distance. For other individuals in the sorted list, their crowding distance with respect to f_i is the difference of the objective values of its left and right neighbor in the sorted list, normalized by the difference between the first and the last.

The whole NSGA-II framework is shown in Algorithm 3. After the random initialization of the population of size N, the main loop starts with the generation of N offspring (the precise way how this is done is not part of the NSGA-II framework and is mostly left as a design choice to the algorithm user in [21], although it is suggested to select parents via binary tournaments based the total order described above). Then the total order based on rank and crowding distance is used to remove the worst N individuals in the union of the parent and offspring population. The remaining individuals form the parent population of the next iteration.

3. Runtime of the NSGA-II on OneMinMax

In this section, we analyze the runtime of the NSGA-II on the OneMinMax benchmark proposed first by Giel and Lehre [30] as a bi-objective analogue of the classic OneMax benchmark. It is the function $f : \{0,1\}^n \to \mathbb{N} \times \mathbb{N}$ defined by

Algorithm 2 Crowding-distance(S).

Input: $S = \{S_1, \dots, S_{|S|}\}$: the set of individuals
Output: $\text{cDis}(S) = (\text{cDis}(S_1), \dots, \text{cDis}(S_{|S|}))$, the vector of crowding distances of the individuals in S

```
cDis(S) = (0, ..., 0)
for each objective function f_i do
    Sort S in order of ascending f_i value: S_{i.1}, ..., S_{i.|S|}
    cDis(S_{i.1}) = +∞, cDis(S_{i.|S|}) = +∞
    for j = 2, ..., |S| − 1 do
        cDis(S_{i.j}) = cDis(S_{i.j}) + (f_i(S_{i.j+1}) − f_i(S_{i.j−1})) / (f_i(S_{i.|S|}) − f_i(S_{i.1}))
    end for
end for
```

Algorithm 3 NSGA-II.

```
Uniformly at random generate the initial population P_0 = {x_1, x_2, ..., x_N} with x_i ∈ {0,1}^n, i = 1, 2, ..., N.
for t = 0, 1, 2, ... do
    Generate the offspring population Q_t with size N
    Use Algorithm 1 to divide R_t = P_t ∪ Q_t into F_1, F_2, ...
    Find i* ≥ 1 such that Σ_{i=1}^{i*−1} |F_i| < N and Σ_{i=1}^{i*} |F_i| ≥ N
    Use Algorithm 2 to separately calculate the crowding distance of each individual in F_1, ..., F_{i*}
    Let F̃_{i*} be the N − Σ_{i=0}^{i*−1} |F_i| individuals in F_{i*} with largest crowding distance, chosen at random in case of a tie
    P_{t+1} = (⋃_{i=1}^{i*−1} F_i) ∪ F̃_{i*}
end for
```

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

for all $x = (x_1, \dots, x_n) \in \{0,1\}^n$. The aim is to maximize both objectives in f. We immediately note that for this benchmark problem, any solution lies on the Pareto front. It is hence a good example to study how an MOEA explores the Pareto front when already some Pareto optima were found.

Giel and Lehre [30] showed that the simple SEMO algorithm finds the full Pareto front of OneMinMax in $O(n^2 \log n)$ iterations and fitness evaluations. Their proof can easily be extended to the GSEMO algorithm. For the SEMO, a (matching) lower bound of $\Omega(n^2 \log n)$ was shown in [9]. An upper bound of $O(\mu n \log n)$ was shown for the hypervolume-based $(\mu + 1)$ SIBEA with $\mu \geq n + 1$ [41]. When the SEMO or GSEMO is enriched with a diversity mechanism (strong enough so that solutions that can create a new point on the Pareto front are chosen with constant probability), then the runtime of these algorithms reduces to $O(n \log n)$ [9].

In contrast to the SEMO and GSEMO as well as the $(\mu+1)$ SIBEA with population size $\mu \geq n+1$, the NSGA-II can lose all solutions covering a point of the Pareto front. In the following lemma, central to our runtime analyses on OneMinMax, we show that this cannot happen when the population size is large enough, namely at least four times the size of the Pareto front. Besides, we shall use $[i..j], i \leq j$, to denote the set $\{i, i+1, \dots, j\}$ in this paper.

Lemma 1. *Consider one iteration of the NSGA-II with population size $N \geq 4(n+1)$ optimizing the* OneMinMax *function. Assume that in some iteration t the combined parent and offspring population $R_t = P_t \cup Q_t$ contains a solution x with objective value $(k, n-k)$ for some $k \in [0..n]$. Then also the next parent population P_{t+1} contains an individual y with $f(y) = (k, n-k)$.*

Proof. It is not difficult to see that for any $x, y \in \{0,1\}^n$, we have $x \not\prec y$ and $y \not\prec x$. Hence, all individuals in R_t have rank one in the non-dominated sorting of R_t, that is, after Step 4 in Algorithm 3. Thus, in the notation of the algorithm, $F_1 = R_t$ and $i^* = 1$.

We calculate the crowding distance of each individual of R_t. Let $k \in [0..n]$. Assume that there is at least one individual $x \in R_t$ such that $f(x) = (k, n-k)$. We recall from Algorithm 2 that $S_{1.1}, \dots, S_{1.2N}$ and $S_{2.1}, \dots, S_{2.2N}$ are the sorted populations based on f_1 and f_2 respectively. Since the individuals with the same objective value will continuously appear in the sorted list w.r.t. this objective value, we know that there exist $a \leq b$ and $a' \leq b'$ such that $[a..b] = \{i \mid f_1(S_{1.i}) = k\}$ and $[a'..b'] = \{i \mid f_2(S_{2.i}) = n-k\}$. From the crowding distance calculation in Algorithm 2, we know that $\text{cDis}(S_{1.a}) \geq \frac{f_1(S_{1.a+1}) - f_1(S_{1.a-1})}{f_1(S_{1.|S|}) - f_1(S_{1.1})} \geq \frac{f_1(S_{1.a}) - f_1(S_{1.a-1})}{f_1(S_{1.|S|}) - f_1(S_{1.1})} > 0$ since $f_1(S_{1.a}) - f_1(S_{a-1}) > 0$ by the definition of a. Similarly, we have $\text{cDis}(S_{1.b}) > 0$, $\text{cDis}(S_{2.a'}) > 0$ and $\text{cDis}(S_{2.b'}) > 0$. For all $j \in [a+1..b-1]$ with $S_{1.j} \notin \{S_{2.a'}, S_{2.b'}\}$, we know $f_1(S_{1.j-1}) = f_1(S_{1.j+1}) = k$ and $f_2(S_{1.j-1}) = f_2(S_{1.j+1}) = n-k$ from the definitions of a, b, a' and b'. Hence, we have $\text{cDis}(S_{1.j}) = \frac{f_1(S_{1.j+1}) - f_1(S_{1.j-1})}{f_1(S_{1.|S|}) - f_1(S_{1.1})} + \frac{f_2(S_{2.j'+1}) - f_2(S_{2.j'-1})}{f_2(S_{2.|S|}) - f_2(S_{2.1})} = 0$.

This shows that the individuals with objective value $(k, n-k)$ and positive crowding distance are exactly $S_{1.a}$, $S_{1.b}$, $S_{2.a'}$ and $S_{2.b'}$. Hence, for each $(k, n-k)$, there are at most four solutions x with $f(x) = (k, n-k)$ and $\text{cDis}(x) > 0$. Noting that the Pareto front size for OneMinMax is $n+1$, the number of individuals with positive crowding distance is at most $4(n+1) \leq N$. Since Step 7 in Algorithm 3 keeps N individuals with largest crowding distance, we know that all individuals with positive crowding distance will be kept. Thus, $y = S_{1.a} \in P_{t+1}$, proving our claim. □

Since Lemma 1 ensures that objective values on the Pareto front will not be lost in the future, we can estimate the runtime of the NSGA-II via the sum of the waiting times for finding a new Pareto solution. Apart from the fact that the NSGA-II generates N solutions per iteration (which requires some non-trivial arguments in the case of binary tournament selection), this analysis resembles the known analysis of the simpler SEMO algorithm [30]. For $N = O(n)$, we also obtain the same runtime estimate (in terms of fitness evaluations).

We start with the easier case that parents are chosen uniformly at random or that each parent creates one offspring.

Theorem 2. *Consider optimizing the* OneMinMax *function via the NSGA-II with one of the following four ways to generate the offspring population in Step 3 in Algorithm 3, namely applying one-bit mutation or standard bit-wise mutation once to each parent or N times choosing a parent uniformly at random and applying one-bit mutation or standard bit-wise mutation to it. If the population size N is at least $4(n+1)$, then the expected runtime is at most $\frac{2e^2}{e-1}n(\ln n+1)$ iterations and at most $\frac{2e^2}{e-1}Nn(\ln n+1)$ fitness evaluations. Besides, let T be the number of iterations to reach the full Pareto front, then $\Pr[T \geq \frac{2e^2(1+\delta)}{e-1}n\ln n] \leq 2n^{-\delta}$ holds for any $\delta \geq 0$.*

Proof. Let $x \in P_t$ and $f(x) = (k, n-k)$ for some $k \in [0..n]$. Let p denote the probability that x is chosen as parent to be mutated. Conditional on that, let p_k^+ denote the probability of generating from x an offspring y_+ with $f(y_+) = (k+1, n-k-1)$ (when $k < n$) and p_k^- denote the probability of generating from x an offspring y_- with $f(y_-) = (k-1, n-k+1)$ (when $k > 0$). Consequently, the probability that R_t contains an individual y_+ with objective value $(k+1, n-k-1)$ is at least pp_k^+, and the probability that R_t contains an individual y_- with objective value $(k-1, n-k+1)$ is at least pp_k^-. Since Lemma 1 implies that any existing OneMinMax objective value will be kept in the iterations afterwards, we know that the expected number of iterations to obtain y_+ (resp. y_-) once x is in the population is at most $\frac{1}{pp_k^+}$ (resp. $\frac{1}{pp_k^-}$).

Assume that the initial population of the Algorithm 3 contains an x with $f(x) = (k_0, n-k_0)$. Then the expected number of iterations to obtain individuals containing objective values $(k_0, n-k_0), (k_0+1, n-k_0-1), \ldots, (n, 0)$ is at most $\sum_{i=k_0}^{n-1}\frac{1}{pp_i^+}$. Similarly, the expected number of iterations to obtain individuals containing objective values $(k_0-1, n-k_0+1), (k_0-2, n-k_0-2), \ldots, (0, n)$ is at most $\sum_{i=1}^{k_0}\frac{1}{pp_i^-}$. Consequently, the expected number of iterations to cover the whole Pareto front is at most $\sum_{i=k_0}^{n-1}\frac{1}{pp_i^+} + \sum_{i=1}^{k_0}\frac{1}{pp_i^-}$.

Now we calculate p for the different ways of selecting parents and p_k^+ and p_k^- for the different mutation operations. If we apply mutation once to each parent in P_t, we apparently have $p = 1$. If we choose the parents independently at random from P_t, then $p = 1-(1-\frac{1}{N})^N \geq 1-\frac{1}{e}$. For one-bit mutation, we have $p_k^+ = \frac{n-k}{n}$ and $p_k^- = \frac{k}{n}$. For standard bit-wise mutation, we have $p_k^+ \geq \frac{n-k}{n}(1-\frac{1}{n})^{n-1} \geq \frac{n-k}{en}$ and $p_k^- \geq \frac{k}{n}(1-\frac{1}{n})^{n-1} \geq \frac{k}{en}$.

From these estimates and the fact that the Harmonic number $H_n = \sum_{i=1}^n \frac{1}{i}$ satisfies $H_n < \ln n + 1$, it is not difficult to see that all cases lead to an expected runtime of at most

$$\sum_{i=0}^{n-1}\frac{1}{pp_i^+} + \sum_{i=1}^{n}\frac{1}{pp_i^-} \leq \sum_{i=0}^{n-1}\frac{1}{(1-\frac{1}{e})\frac{n-i}{en}} + \sum_{i=1}^{n}\frac{1}{(1-\frac{1}{e})\frac{i}{en}}$$
$$= 2\sum_{i=1}^{n}\frac{1}{(1-\frac{1}{e})\frac{i}{en}} < \frac{2e^2}{e-1}n(\ln n+1)$$

iterations, hence at most $\frac{2e^2}{e-1}Nn(\ln n+1)$ fitness evaluations.

Now we will prove the concentration result. Let X_k^+ and X_k^- be independent geometric random variables with success probabilities of $(1-\frac{1}{e})\frac{n-k}{en}$ and $(1-\frac{1}{e})\frac{k}{en}$, respectively. Let T be the number of iterations to cover the full Pareto front, and let $Z^+ = \sum_{k=0}^{n-1} X_k^+$ and $Z^- = \sum_{k=1}^{n} X_k^-$. Then from the above discussion, we know that $Z := Z^+ + Z^-$ stochastically dominates T (see [17] for a detailed discussion of how to use stochastic domination arguments in the analysis of evolutionary algorithms). Let the success probabilities of $X_{n-1}^+, X_{n-2}^+, \ldots, X_0^+$ be $q_1^+, \ldots, q_n^+$, and let $q_1^-, \ldots, q_n^-$ denote the success probabilities of $X_1^-, X_2^-, \ldots, X_n^-$. Then we have $q_i^+ \geq (1-\frac{1}{e})\frac{1}{e}\frac{i}{n}$ and $q_i^- \geq (1-\frac{1}{e})\frac{1}{e}\frac{i}{n}$ for all $i \in [1..n]$. From a Chernoff bound for a sum of such geometric random variables ([12, Lemma 4], also found in [18, Theorem 1.10.35]), we have that for any $\delta \geq 0$,

$$\Pr\left[Z^+ \geq (1+\delta)\frac{e^2}{e-1}n\ln n\right] \leq n^{-\delta}$$

and

$$\Pr\left[Z^- \geq (1+\delta)\frac{e^2}{e-1}n\ln n\right] \leq n^{-\delta}.$$

Hence, we have

$$\Pr\left[Z \geq (1+\delta)\frac{2e^2}{e-1} n \ln n\right] \leq 2n^{-\delta}.$$

Since Z stochastically dominates T, we obtain $\Pr[T \geq \frac{2e^2(1+\delta)}{e-1} n \ln n] \leq 2n^{-\delta}$. □

We now analyze the performance of the NSGA-II on OneMinMax when selecting the parents via binary tournaments, which is the selection method suggested in the original NSGA-II paper [21]. We regard two variants of this selection method. The most natural one, discussed for example in [28], is to conduct N *independent tournaments*. Here the offspring population Q_t is generated by N times independently performing the following sequence of actions: (i) Select two different individuals x', x'' uniformly at random from P_t. (ii) Select x as the better of these two, that is, the one with smaller rank in P_t or, in case of equality, the one with larger crowding distance in P_t (breaking ties randomly). (iii) Generate an offspring by mutating x. We note that in some definitions of tournament selection the better individual in a tournament is chosen as winner only with some probability $p > 0.5$, but we do not regard this case any further. We note though that all our mathematical results would remain true in this setting. We also note that sometimes the participants of a tournament are selected "with replacement". Again, this would not change our results, but we do not discuss this case any further.

A closer look in Deb's implementation of the NSGA-II (see Revision 1.1.6 available at [14]), and we are thankful for Maxim Buzdalov (Aberystwyth University) for pointing this out to us, shows that here a different way of selecting the parents is used. In this *two-permutation tournament selection scheme*, two random permutations π_1 and π_2 of P_t are generated and then a binary tournament is conducted between $\pi_j(2i-1)$ and $\pi_j(2i)$ for all $i \in [1..\frac{N}{2}]$ and $j \in \{1,2\}$ (we assume here that N is even). Of course, this is nothing else than saying that twice a random matching on P_t is generated and the end vertices of each matching edge conduct a tournament. Different from independent tournaments, this selection operator cannot be implemented in parallel. On the positive side, it ensures that each individual takes part in exactly two tournaments, so it treats the individuals in a fairer manner. Also, if there is a unique best individual, then this will surely be selected. As above, in our setting where we do not use crossover, each tournament winner is mutated and these N individuals form the offspring population Q_t.

In the case of binary tournament selection, the analysis is slightly more involved since we need to argue that a desired parent is chosen for mutation with constant probability in one iteration. This is easy to see for a parent at the boundary of the front as its crowding distance is infinite, but less obvious for parents not at the boundary. We note that we need to be able to select such parents since we cannot ensure that the population intersects the Pareto front in a contiguous interval (as can be seen, e.g., from the random initial population). We solve this difficulty in the following three lemmas.

We use the following notation. Consider some iteration t. For $i = 1, 2$, let

$$v_i^{\min} = \min\{f_i(x) \mid x \in R_t\},$$
$$v_i^{\max} = \max\{f_i(x) \mid x \in R_t\}$$

denote the extremal objective values in the combined parent and offspring population. Let $V = f(R_t) = \{(f_1(x), f_2(x)) \mid x \in R_t\}$ denote the set of objective values of the solutions in the combined parent and offspring population R_t. We define the set of values such that also the right (left) neighbor on the Pareto front is covered by

$$V_{\text{in}}^+ = \{(v_1, v_2) \in V \mid \exists y \in R_t : (f_1(y), f_2(y)) = (v_1 + 1, v_2 - 1)\},$$
$$V_{\text{in}}^- = \{(v_1, v_2) \in V \mid \exists y \in R_t : (f_1(y), f_2(y)) = (v_1 - 1, v_2 + 1)\}.$$

Lemma 3. *For any $(v_1, v_2) \in V \setminus (V_{\text{in}}^+ \cap V_{\text{in}}^-)$, there is at least one individual $x \in R_t$ with $f(x) = (v_1, v_2)$ and $\text{cDis}(x) \geq \frac{2}{v_1^{\max} - v_1^{\min}}$.*

Proof. Let $(v_1, v_2) \in V \setminus (V_{\text{in}}^+ \cap V_{\text{in}}^-)$, let $S_{1.1}, \dots, S_{1.2N}$ be the sorting of R_t according to f_1, and let $[a..b] = \{i \in [1..2N] \mid f_1(S_{1.i}) = v_1\}$. If $v_1 \in \{v_1^{\max}, v_1^{\min}\}$, then by definition of the crowding distance, one individual in $f^{-1}((v_1, v_2))$ has an infinite crowding distance. Otherwise, if $(v_1, v_2) \in V \setminus V_{\text{in}}^-$, then we have $f_1(S_{1.a+1}) - f_1(S_{1.a-1}) \geq f_1(S_{1.a}) - f_1(S_{1.a-1}) \geq 2$ and thus $\text{cDis}(S_{1.a}) \geq \frac{f_1(S_{1.a+1}) - f_1(S_{1.a-1})}{v_1^{\max} - v_1^{\min}} \geq \frac{2}{v_1^{\max} - v_1^{\min}}$. Similarly, if $(v_1, v_2) \in V \setminus V_{\text{in}}^+$, then $f_1(S_{1.b+1}) - f_1(S_{1.b-1}) \geq f_1(S_{1.b+1}) - f_1(S_{1.b}) \geq 2$ and $\text{cDis}(S_{1.b}) \geq \frac{f_1(S_{1.b+1}) - f_1(S_{1.b-1})}{v_1^{\max} - v_1^{\min}} \geq \frac{2}{v_1^{\max} - v_1^{\min}}$. □

Lemma 4. *For any $(v_1, v_2) \in V_{\text{in}}^+ \cap V_{\text{in}}^-$, there are at most two individuals in R_t with objective value (v_1, v_2) and crowding distance at least $\frac{2}{v_1^{\max} - v_1^{\min}}$.*

Proof. Let $(v_1, v_2) \in V_{\text{in}}^+ \cap V_{\text{in}}^-$, $[a..b] = \{i \in [1..2N] \mid f_1(S_{1.i}) = v_1\}$, and $[a'..b'] = \{j \in [1..2N] \mid f_2(S_{2.j}) = v_2\}$. Let $C = \{S_{1.a}, S_{1.b}\} \cup \{S_{2.a'}, S_{2.b'}\}$. If $(R_t \cap f^{-1}((v_1, v_2))) \setminus C$ is not empty, then for any $x \in (R_t \cap f^{-1}((v_1, v_2))) \setminus C$, there exist

$i \in [a+1..b-1]$ and $j \in [a'+1..b'-1]$ such that $x = S_{1.i} = S_{2.j}$. Hence $\mathrm{cDis}(x) = \frac{f_1(S_{1.i+1})-f_1(S_{1.i-1})}{v_1^{\max}-v_1^{\min}} + \frac{f_2(S_{2.j+1})-f_2(S_{2.j-1})}{v_2^{\max}-v_2^{\min}} = 0$. We thus know that any individual with crowding distance at least $\frac{2}{v_1^{\max}-v_1^{\min}}$ lies in C.

For any $x \in C \setminus (\{S_{1.a}, S_{1.b}\} \cap \{S_{2.a'}, S_{2.b'}\})$, we have $\mathrm{cDis}(x) = \frac{1}{v_1^{\max}-v_1^{\min}}$ or $\mathrm{cDis}(x) = \frac{1}{v_2^{\max}-v_2^{\min}}$. We note that $v_1^{\max} - v_1^{\min} = v_2^{\max} - v_2^{\min}$ since $v_1^{\max} = n - v_2^{\min}$ and $v_1^{\min} = n - v_2^{\max}$. Hence $\mathrm{cDis}(x) < \frac{2}{v_1^{\max}-v_1^{\min}}$. Let now $x \in \{S_{1.a}, S_{1.b}\} \cap \{S_{2.a'}, S_{2.b'}\}$. If $|C| = 1$, then $\mathrm{cDis}(x) = \frac{2}{v_1^{\max}-v_1^{\min}} + \frac{2}{v_2^{\max}-v_2^{\min}} = \frac{4}{v_1^{\max}-v_1^{\min}}$. Otherwise, $\mathrm{cDis}(x) = \frac{1}{v_1^{\max}-v_1^{\min}} + \frac{1}{v_2^{\max}-v_2^{\min}} = \frac{2}{v_1^{\max}-v_1^{\min}}$. Therefore, the number of individuals in $R_t \cap f^{-1}((v_1, v_2))$ with crowding distance at least $\frac{2}{v_1^{\max}-v_1^{\min}}$ is $|\{S_{1.a}, S_{1.b}\} \cap \{S_{2.a'}, S_{2.b'}\}|$, which is at most 2. □

Lemma 5. *Let $N \geq 4(n+1)$. Let P be a parent population in a run of the NSGA-II using independent or two-permutation binary tournament selection optimizing* OneMinMax. *Let $v = (v_1, n - v_1) \notin f(P)$ be a point on the Pareto front that is not covered by P, but a neighbor of v on the front is covered by P, that is, there is a $y \in P$ such that $\|f(y) - v\|_\infty = 1$.*

In the case of independent tournaments, each of the N tournaments with probability at least $\frac{1}{N}(\frac{1}{6} - \frac{3.5}{N-1})$ selects an individual x with $f(x) = f(y)$.

In the case of two-permutation selection, there are two stochastically independent tournaments each of which with probability at least $\frac{1}{6} - \frac{2.5}{N-1}$ selects an individual x with $f(x) = f(y)$.

Proof. By Lemma 3, there is an individual $x' \in P$ with $f(x') = f(y)$ and crowding distance at least $\frac{2}{v_1^{\max}-v_1^{\min}}$. We estimate the probability that x' is the winner of a tournament.

We start with the case of independent tournaments and we regard a particular one of these. With probability $\frac{1}{N}$, the individual x' is chosen as the first participant of the tournament. We condition on this and regard the second individual x'' of the tournament, which is chosen uniformly at random from the remaining $N-1$ individuals. We shall argue that with good probability, it has a smaller crowding distance, and thus loses the tournament.

To this aim, we estimate the number of element $z \in P$ that have a crowding distance of $\frac{2}{v_1^{\max}-v_1^{\min}}$ or more ("large crowding distance"). We treat the individuals differently according to their objective value $w = f(z)$. If $w \in V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-$, then by Lemma 4 at most two individuals with this objective value have a large crowding distance. For other objective values w, we use the general estimate from the proof of Lemma 1 that at most four individuals have this objective value and positive crowding distance. This gives an upper bound of $2|V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-| + 4(|f(P)| - |V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-|) = 2|f(P)| + 2|f(P) \setminus (V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-)|$ individuals with large crowding distance.

We note that out of each consecutive three elements $(w_1, n-w_1), (w_1+1, n-w_1-1), (w_1+2, n-w_1-2)$ of the Pareto front, at most two can be in $f(P) \setminus (V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-)$ – if all three were in $f(P)$, then the middle one would necessarily be in $V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-$. Consequently, $|f(P) \setminus (V_{\mathrm{in}}^+ \cap V_{\mathrm{in}}^-)| \leq 2\lceil \frac{n+1}{3} \rceil$. With this estimate, our upper bound on the number of individuals with large crowding distance becomes at most $2(n+1) + 4\lceil \frac{n+1}{3} \rceil \leq \frac{10}{3}(n+1) + \frac{8}{3}$, and then excluding the first-chosen individual x', we know that the upper bound estimate for the probability that x'' has large crowding distance becomes $\frac{1}{N-1}(\frac{10}{3}(n+1) + \frac{8}{3} - 1) = \frac{1}{N-1}(\frac{10}{3}(n+\frac{3}{4}) + \frac{10}{3}\frac{1}{4} + \frac{5}{3}) \leq \frac{5}{6} + \frac{1}{N-1}(\frac{10}{3}\frac{1}{4} + \frac{5}{3})$. Consequently, the probability that x' is selected as first participant of the tournament and it wins the tournament is at least

$$\frac{1}{N}\left(\frac{1}{6} - \frac{2.5}{N-1}\right).$$

For the case of two-permutation tournament selection, we note that there are two independent tournaments (stemming from different permutations) in which x' participates. In both, the partner x'' of x' is distributed uniformly in $P_t \setminus \{x'\}$. Hence the above arguments can be applied and we see that with probability at least $\frac{1}{6} - \frac{2.5}{N-1}$, the second participant loses against x'. □

With Lemma 5, we can now easily argue that in a given iteration t, we have a constant probability of choosing at least once a parent that is a neighbor of an empty spot on the Pareto front. This allows to re-use the main arguments of the simpler analyses for the cases that the parents were choosing randomly or that each parent creates one offspring. We note that in the following result, as in any other result in this work, we did not try to optimize the leading constant.

Theorem 6. *Let $n \geq 4$. Consider optimizing the* OneMinMax *function via the NSGA-II which creates the offspring population by selecting parents via independent binary tournaments or via the two-permutation approach and applying one-bit or standard bit-wise mutation to these. If the population size N is at least $4(n+1)$, then the expected runtime is at most $\frac{200e}{3}n(\ln n + 1)$ iterations and at most $\frac{200e}{3}Nn(\ln n + 1)$ fitness evaluations. Besides, let T be the number of iterations to reach the full Pareto front. Then we further have that $\Pr[T \geq \frac{200e}{3}(1+\delta)n \ln n] \leq 2n^{-\delta}$ holds for any $\delta \geq 0$.*

Proof. Thanks to Lemma 5, we can essentially follow the arguments of the proof of Theorem 2. Let $y \in P_t$ be such that $f(y)$ is a neighbor of a point on the Pareto front that is not in $f(P_t)$.

For independent tournaments, by Lemma 5 a single tournament will select a parent x with $f(x) = f(y)$, that is, also a neighbor of this uncovered point, with probability at least $\frac{1}{N}(\frac{1}{6} - \frac{2.5}{N-1})$. Hence the probability that at least one such parent is selected in this iteration is

$$p = 1 - \Big(1 - \tfrac{1}{N}(\tfrac{1}{6} - \tfrac{2.5}{N-1})\Big)^N \geq 1 - \exp\Big(-\tfrac{1}{6} + \tfrac{2.5}{N-1}\Big) > 0.03,$$

where the last inequality uses $N \geq 20$ from $n \geq 4$ and $N \geq 4(n+1)$.

For two-permutation tournament selection, again by Lemma 5, with probability at least $p = 1 - (1 - (\frac{1}{6} - \frac{2.5}{N-1}))^2 > 0.03$ (since $N \geq 20$) a parent x with $f(x) = f(y)$ is selected.

With these values of p, the proof of Theorem 2 extends to the two cases of tournament selection, and we know that the expected iterations to cover the full Pareto front is at most

$$\begin{aligned}\sum_{i=0}^{n-1} \frac{1}{pp_i^+} + \sum_{i=1}^{n} \frac{1}{pp_i^-} &\leq \sum_{i=0}^{n-1} \frac{1}{0.03\frac{n-i}{en}} + \sum_{i=1}^{n} \frac{1}{0.03\frac{i}{en}} \\ &= 2\sum_{i=1}^{n} \frac{1}{0.03\frac{i}{en}} < \tfrac{200e}{3} n(\ln n + 1).\end{aligned}$$

We now discuss the concentration result. With the same arguments as in the proof of Theorem 2, but using the success probabilities $0.03\frac{n-k}{en}$ and $0.03\frac{k}{en}$ for X_k^+ and X_k^- respectively and estimating $q_i \geq \frac{0.03}{e}\frac{i}{n}$, we obtain that for any $\delta \geq 0$, we have $\Pr[T \geq \frac{200e}{3}(1+\delta)n\ln n] \leq 2n^{-\delta}$. □

4. Runtime of the NSGA-II on LeadingOnesTrailingZeroes

We proceed with analyzing the runtime of the NSGA-II on the benchmark LeadingOnesTrailingZeroes proposed by Laumanns, Thiele, and Zitzler [36]. This is the function $f : \{0,1\}^n \to \mathbb{N} \times \mathbb{N}$ defined by

$$f(x) = \big(f_1(x), f_2(x)\big) = \Big(\sum_{i=1}^{n}\prod_{j=1}^{i} x_j, \sum_{i=1}^{n}\prod_{j=i}^{n}(1-x_j)\Big)$$

for all $x \in \{0,1\}^n$. Here the first objective is the so-called LeadingOnes function, counting the number of (contiguous) leading ones of the bit string, and the second objective counts in an analogous fashion the number of trailing zeros. Again, the aim is to maximize both objectives. Different from OneMinMax, here many solutions exist that are not Pareto optimal. The known runtimes for this benchmark are $\Theta(n^3)$ for the SEMO [36], $O(n^3)$ for the GSEMO [29], and $O(\mu n^2)$ for the $(\mu+1)$ SIBEA with population size $\mu \geq n+1$ [2].

Similar to OneMinMax, we can show that when the population size is large enough, an objective value on the Pareto front stays in the population from the point on when it is discovered.

Lemma 7. *Consider one iteration of the NSGA-II with population size $N \geq 4(n+1)$ optimizing the* LeadingOnesTrailingZeroes *function. Assume that in some iteration t the combined parent and offspring population $R_t = P_t \cup Q_t$ contains a solution x with rank one. Then also the next parent population P_{t+1} contains an individual y with $f(y) = f(x)$. In particular, once the parent population contains an individual with objective value $(k, n-k)$, it will do so for all future generations.*

Proof. Let F_1 be the set of solutions of rank one, that is, the set of solutions in R_t that are not dominated by any other individual in R_t. By definition of dominance, for each $v_1 \in \{f_1(x) \mid x \in F_1\}$, there exists a unique v_2 such that $(v_1, v_2) \in \{f(x) \mid x \in F_1\}$. Therefore, $|f(F_1)|$ is at most $n+1$. We now reuse the argument from the proof of Lemma 1 for OneMinMax that for each objective value, there are at most 4 individuals with this objective value and positive crowding distance. Thus the number of individuals in F_1 with positive crowding distance is at most $4(n+1) \leq N$. Since the NSGA-II keeps N individuals with smallest rank and largest crowding distance in case of a tie, we know that the individuals with rank one and positive crowding distance will all be kept. This shows the first claim.

For the second claim, let $x \in P_t$ with $f(x) = (k, n-k)$ for some k. Since x lies on the Pareto front of LeadingOnesTrailingZeroes, the rank of x in R_t is necessarily one. Hence by the first claim, a y with $f(y) = f(x)$ will be contained in P_{t+1}. A simple induction extends this finding to all future generations. □

Since not all individuals are on the Pareto front, the runtime analysis for LeadingOnesTrailingZeroes function is slightly more complex than for OneMinMax. We analyze the process in two stages: the first stage lasts until we have found both extremal solutions of the Pareto front. In this phase, we argue that the first (resp. second) objective value increases by

one every (expected) $O(n)$ iterations. Consequently, after an expected number of $O(n^2)$ iterations, we have an individual x in the population with $f_1(x) = n$ (resp. $f_2(x) = n$), which are the desired extremal individuals. The second stage, where we complete the Pareto front from existing Pareto solutions, can be analyzed in a similar manner as for OneMinMax in Theorem 2, noting of course the different probabilities to generate a new solution on the Pareto front. We start with the two easier parent selections and discuss tournament selection separately in Theorem 9.

Theorem 8. *Consider optimizing the* LeadingOnesTrailingZeroes *function via the NSGA-II with one of the following four ways to generate the offspring population in Step 3 in Algorithm 3, namely applying one-bit mutation or standard bit-wise mutation once to each parent or N times choosing a parent uniformly at random and applying one-bit mutation or standard bit-wise mutation to it. If the population size N is at least $4(n+1)$, then the expected runtime is $\frac{2e^2}{e-1}n^2$ iterations and $\frac{2e^2}{e-1}Nn^2$ fitness evaluations. Besides, let T be the number of iterations to reach the full Pareto front. Then*

$$\Pr\left[T \geq \frac{2e^2(1+\delta)}{e-1}n^2\right] \leq \exp\left(-\frac{\delta^2}{2(1+\delta)}(2n-1)\right)$$

holds for any $\delta \geq 0$.

Proof. Consider one iteration t of the first stage, that is, we have $P_t^p = \{x \in P_t \mid f_1(x) + f_2(x) = n\} = \emptyset$. Let $v_t = \max\{f_1(x) \mid x \in P_t\}$ and let $P_t^* = \{x \in P_t \mid f_1(x) = v_t\}$. Note that by Lemma 7, v_t is non-decreasing over time. Let $x \in P_t^*$. Let p denote the probability that x is chosen as a parent to be mutated (note that this probability is independent of x for the two selection schemes regarded here). Conditional on that, let p^* be a lower bound (independent of x) on the probability that x generates a solution with a larger f_1-value. Then the expected number of iterations to obtain a solution with better f_1-value is at most $\frac{1}{pp^*}$. Consequently, the expected number of iterations to obtain a f_1-value of n, thus a solution on the Pareto front, is at most $(n-k_0)\frac{1}{pp^*} \leq n\frac{1}{pp^*}$, where k_0 is the maximum LeadingOnes value in the initial population.

For the second stage, let $x \in P_t^p$ be such that a neighbor of $f(x)$ on the front is not yet covered by P_t. Let p' denote the probability that x is chosen as a parent to be mutated. Conditional on that, let p^{**} denote a lower bound (independent of x) for the probability to generate a particular neighbor of x on the front. Consequently, the probability that R_t covers an extra element of the Pareto front is at least $p'p^{**}$. Since Lemma 7 implies that any existing LeadingOnesTrailingZeroes value on the front will be kept in the following iterations, we know that the expected number of iterations for this progress is at most $\frac{1}{p'p^{**}}$. Since n such progresses are sufficient to cover the full Pareto front, the expected number of iterations to cover the whole Pareto front is at most $n\frac{1}{p'p^{**}}$. Therefore, the expected total runtime is at most $n\frac{1}{pp^*} + n\frac{1}{p'p^{**}}$ iterations.

We recall from Theorem 2 that we have $p = p' = 1$ when selecting each parent once and we have $p = p' = 1 - (1 - \frac{1}{N})^N \geq 1 - \frac{1}{e}$ when choosing parents randomly. To estimate p^* and p^{**}, we note that the desired progress can always be obtained by flipping one particular bit. Hence for one-bit mutation, we have $p^* = p^{**} = \frac{1}{n}$. For standard bit-wise mutation, $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$ is a valid lower bound for p^* and p^{**}.

With these estimates, we obtain in all cases an expected runtime of at most

$$n\frac{1}{pp^*} + n\frac{1}{p'p^{**}} \leq n\frac{1}{(1-\frac{1}{e})\frac{1}{en}} + n\frac{1}{(1-\frac{1}{e})\frac{1}{en}} = \frac{2e^2n^2}{e-1}$$

iterations, hence $\frac{2e^2}{e-1}Nn^2$ fitness evaluations.

Now we will prove the concentration result. The time to cover the full Pareto front is divided into two stages as discussed before. It is not difficult to see that the first stage is to reach a Pareto optimum for the first time, and the corresponding runtime is dominated by the sum of n independent geometric random variables with success probabilities of $(1-\frac{1}{e})\frac{1}{en}$. The second stage is to cover the full Pareto front, and the corresponding runtime is dominated by the sum of another n such independent geometric random variables. Formally, let $X_1, \ldots, X_{2n}$ be independent geometric random variables with success probabilities of $(1-\frac{1}{e})\frac{1}{en}$, and let T be the number of iterations to cover the full Pareto front. Then $Z := \sum_{i=1}^{2n} X_i$ stochastically dominates T, and $E[Z] = \frac{2e^2n^2}{e-1}$. From a Chernoff bound for sums of independent identically distributed geometric random variables [18, (1.10.46) in Theorem 1.10.32], we have that for any $\delta \geq 0$,

$$\Pr\left[Z \geq (1+\delta)\frac{2e^2n^2}{e-1}\right] \leq \exp\left(-\frac{\delta^2}{2}\frac{2n-1}{1+\delta}\right).$$

Since Z dominates T, we have proven this theorem. □

We now study the runtime of the NSGA-II using binary tournament selection. Compared to OneMinMax, we face the additional difficulty that now rank one solutions can exist which are not on the Pareto front. Due to their low rank, they could perform well in the selection, but being possibly far from the front, they are not interesting as parents. We need a sightly different general proof outline to nevertheless argue that sufficiently often a parent on the Pareto front generates

a new neighbor on the front. Also, since not all individuals are on the Pareto front, we do not have anymore the property that the difference between the maximum and minimum value is the same for both objectives. We overcome this by first showing the NSGA-II finds the two extremal points of the Pareto front in reasonable time (then the maximum values are both n and the minimum values are both 0).

Theorem 9. *Consider optimizing the* LeadingOnesTrailingZeroes *function via the NSGA-II. Assume that the parents for variation are chosen either via N independent random tournaments between different individuals or via the two-permutation implementation of binary tournaments. Assume that these parents are mutated via one-bit or standard bit-wise mutation. If the population size N is at least $4(n+1)$, then the expected runtime is at most $15en^2$ iterations and at most $15eNn^2$ fitness evaluations. Besides, let T be the number of iterations to reach the full Pareto front, then*

$$\Pr\left[T \geq \frac{(1+\delta)100e}{3}n^2\right] \leq \exp\left(-\frac{\delta^2}{2(1+\delta)}(3n-1)\right)$$

holds for any $\delta \geq 0$.

Proof. We first argue that, regardless of the initial state of the NSGA-II, it takes $O(n^2)$ iterations until the extremal point $(1,\dots,1)$, which is the unique maximum of f_1, is in P_t. To this aim, let $X_t := \max\{f_1(x) \mid x \in P_t\}$ denote the maximum f_1 value in the parent population. We note that any $x \in P_t$ with $f_1(x) = X_t$ lies on the first front F_1 and that there is a $y \in P_t$ with infinite crowding distance and $f(y) = f(x)$, in particular, $f_1(y) = X_t$.

If parents are chosen via independent tournaments, such a y has a $\frac{2}{N}$ chance of being one of the two individuals of a fixed tournament. It then wins the tournament with at least 50% chance (where the 50% show up only in the rare case that the other individual also lies on the first front and has an infinite crowding distance). Hence the probability that this y is chosen at least once as a parent to be mutated is at least $p = 1 - (1 - \frac{1}{2}\frac{2}{N})^N \geq 1 - \frac{1}{e}$.

When the two-permutation implementation of tournament selection is used, then y appears in both permutations and has a random partner in both. Again, this partner with probability at most $\frac{1}{2}$ wins the tournament. Hence the probability that y is selected as a parent at least once is at least $p = 1 - (\frac{1}{2})^2 = \frac{3}{4}$.

Conditional on y being chosen at least once, let us regard a fixed mutation step in which y was selected as a parent. To mutate y into an individual with higher f_1 value, it suffices to flip a particular single bit (namely the first zero after the initial contiguous segment of ones). The probability for this is $p^* = \frac{1}{n}$ for one-bit mutation and $p^* = \frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$ for standard bit-wise mutation. Denoting by $Y_t := \max\{f_1(x) \mid x \in R_t\}$ the maximum f_1 value in the combined parent and offspring population, we have just shown that $\Pr[Y_t \geq X_t + 1] \geq pp^* = \Omega(1/n)$ whenever $X_t < n$. We note that any $x \in R_t$ with $f_1(x) = Y_t$ lies on the first front F_1 of R_t and that there is a $y \in R_t$ with infinite crowding distance and $f(y) = f(x)$, in particular, $f_1(y) = Y_t$. Consequently, such a y will be kept in the next parent population P_{t+1} (note that there are at most 4 individuals in F_1 with infinite crowding distance – since $N \geq 4$, they will all be included in P_{t+1}). This shows that we also have $\Pr[X_{t+1} \geq X_t + 1] \geq pp^*$ whenever $X_t < n$. By adding the expected waiting times for an increase of the X_t value, we see that the expected time to have $X_t = n$, that is, to have $(1,\dots,1) \in P_t$, is at most

$$\frac{n}{pp^*} \leq \frac{n}{(1-\frac{1}{e})\frac{1}{en}} = \frac{e^2n^2}{e-1}$$

iterations.

By a symmetric argument, we see that after another at most $\frac{e^2}{e-1}n^2$ iterations, also the other extremal point $(0,\dots,0)$ is in the population (and remains there forever by Lemma 7).

With now both extremal points of the Pareto front covered, we analyze the remaining time until the Pareto front is fully covered. We note that by Lemma 7, the number of Pareto front points covered cannot decrease. Hence it suffices to prove a lower bound for the probability that the coverage increases in one iteration. This is what we do now.

Assume that the Pareto front is not yet fully covered. Since we have some Pareto optimal individuals, there also is a Pareto optimal individual $x \in P_t$ such that $f(x)$ is a neighbor of a point v on the Pareto front that is not covered. Since we have both extremal points in the Pareto front, the differences between the maximum and minimum value are the same for both objectives (namely n). Consequently, in the same way as in the proof of Lemma 3, we know that there is also such a y with $f(y) = f(x)$ and with crowding distance at least $\frac{2}{n}$.

We estimate the number of individuals in $P_t \setminus \{y\}$ which could win a tournament against this y. Clearly, these can only be individuals in the first front F_1 of the non-dominated sorting. Assume first that $|f(F_1)| \leq 0.8(n+1)$. We note that, just by the definition of crowding distance and in a similar fashion as in the proof of Lemma 1, for each $v \in f(F_1)$ there are at most four individuals with f value equal to v and positive crowding distance. All other individuals in F_1 have a crowding distance of zero (and thus lose the tournament against y), as do all individuals not in F_1. Consequently, there are at least $N_0 = N - 4 \cdot 0.8(n+1)$ individuals other than y that would lose a tournament against y.

Assume now that $m := |f(F_1)| > 0.8(n+1)$. Since F_1 consists of pair-wise incomparable solutions or solutions with identical objective value (which we may ignore for the following argument), we have $|f_1(F_1)| = |f_2(F_1)| = m$. For any

$v = (v_1, v_2)$, let $v^+ := (v_1 + 1, v_2 - 1)$ and $v^- := (v_1 - 1, v_2 + 1)$. Then we divide $f(F_1)$ into two disjoint sets $U_1 = \{v \in f(F_1) \cap [1..n-1]^2 \mid v^+ \notin f(F_1) \text{ or } v^- \notin f(F_1)\}$ and $U_2 = f(F_1) \setminus U_1$. Since both $f_1(F_1)$ and $f_2(F_1)$ are subsets of $[0..n]$, which has $n+1$ elements, we see that less than $0.2(n+1)$ of the values in $[0..n]$ are missing in $f_1(F_1)$, and analogously in $f_2(F_1)$. Since each value missing in $f_1(F_1)$ or $f_2(F_1)$ leads to at most two values in U_1, we have $|U_1| < 4 \cdot 0.2(n+1) = 0.8(n+1)$. For the values in U_1, we use the blunt estimate from above that at most 4 individuals with this objective value and positive crowding distance exist. For the values $v \in U_2$, we are in the same situation as in Lemma 4, and thus there are at most two individuals $x \in F_1$ with $f(x) = v$ and crowding distance at least $\frac{2}{n}$ (this was not formally proven in Lemma 4 for the case that $v \in \{(0,n),(n,0)\}$ and the unique neighbor of v is in $f(F_1)$, but it is easy to see that in this case only the at most two x with $f(x) = v$ and infinite crowding distance can have a crowding distance of at least $\frac{2}{n}$). Consequently, there are more than

$$\begin{aligned} N - 4|U_1| - 2|U_2| &= N - 4|U_1| - 2(m - |U_1|) = N - 2|U_1| - 2m \\ &> N - 2 \cdot 0.8(n+1) - 2(n+1) = N - 3.6(n+1) \end{aligned}$$

individuals in $P_t \setminus \{y\}$ that would lose against y. Note that this bound is weaker than the one from the first case, so it is valid in both cases.

From this, we now estimate the probability that y is selected as a parent at least once. We first regard the case of independent tournaments. The probability that y is the winner of a fixed tournament is at least the probability that it is chosen as the first contestant times the probability that one of the at least $N - 3.6(n+1)$ sure losers is chosen as the second contestant. This probability is at least $\frac{1}{N} \cdot \frac{N-3.6(n+1)}{N-1} \geq \frac{1}{N} \frac{0.4(n+1)}{4n+3} \geq 0.1\frac{1}{N}$. Hence the probability p that y is chosen at least once as a parent for mutation is at least $p \geq 1 - (1 - 0.1\frac{1}{N})^N \geq 1 - \exp(-0.1) \geq 0.09$. For the two-permutation implementation of tournament selection, y appears in both permutations and has a random partner in each of them. Hence the probability that y wins at least one of these two tournaments is at least $p \geq 1 - (1 - \frac{N-3.6(n+1)}{N-1})^2 \geq 1 - (1-0.1)^2 = 0.19$.

Conditional on y being selected at least once, we regard a mutation step in which y is selected. The probability p^* that the Pareto optimal y is mutated into the unique Pareto optimal bit string z with $f(z) = v$ is $p^* = \frac{1}{n}$ for one-bit mutation and $p^* = \frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq \frac{1}{en}$ for standard bit-wise mutation. Consequently, the probability that one iteration generates the missing Pareto front value v is at least pp^*, the expected waiting time for this is at most $\frac{1}{pp^*}$ iterations, and the expected time to create all missing Pareto front values is at most

$$\frac{n}{pp^*} \leq \frac{n}{0.09\frac{1}{en}} = \frac{100en^2}{9}$$

iterations. Hence, the runtime for the full coverage of the Pareto front starting from the initial population is at most

$$\tfrac{e^2}{e-1}n^2 + \tfrac{e^2}{e-1}n^2 + \tfrac{100e}{9}n^2 < 15en^2$$

iterations, which is at most $15eNn^2$ fitness evaluations.

Now we will prove the concentration result. Note that in this proof, we consider three phases, the first phase to reach the extremal point $(1,\dots,1)$, the second phase to reach $(0,\dots,0)$, and the third phase to cover the full Pareto front. The runtime for each phase is dominated by the sum of n independent geometric random variables with success probabilities of $\frac{0.09}{en}$. Formally, let $X_1,\dots,X_{3n}$ be independent geometric random variables with success probabilities of $\frac{0.09}{en}$, and let T be the number of iterations to cover the full Pareto front. Then we have $Z := \sum_{i=1}^{3n} X_i$ stochastically dominates T, and $E[Z] = \frac{3en^2}{0.09}$. From the Chernoff bound [18, (1.10.46) in Theorem 1.10.32], we have that for any $\delta \geq 0$,

$$\Pr\left[Z \geq (1+\delta)\frac{100e}{3}n^2\right] \leq \exp\left(-\frac{\delta^2}{2}\frac{3n-1}{1+\delta}\right).$$

Since Z dominates T, this shows the theorem. □

5. An exponential lower bound for small population size

In this section, we prove a lower bound for a small population size. Since lower bound proofs can be quite complicated – recall for example that there are matching upper and lower bounds for the runtime of the SEMO (using one-bit mutation) on OneMinMax and LeadingOnesTrailingZeroes, but not for the GSEMO (using bit-wise mutation) – we restrict ourselves to the simplest variant using each parent once to generate one offspring via one-bit mutation. From the proofs, though, we are optimistic that our results, with different implicit constants, can also be shown for all other variants of the NSGA-II regarded in this work. Our experiments support this believe, see Fig. 3 in Section 6.

Our main result is that this NSGA-II takes an exponential time to find the whole Pareto front (of size $n+1$) of OneMinMax when the population size is $n+1$. This is different from the SEMO and GSEMO algorithms (which have no fixed population size, but which will never store a population larger than $n+1$ when optimizing OneMinMax) and the

$(\mu+1)$ SIBEA with population size $\mu = n+1$. Even stronger, we show that there is a constant $\varepsilon > 0$ such that when the current population P_t covers at least $|f(P_t)| \geq (1-\varepsilon)(n+1)$ points on the Pareto front of OneMinMax, then with probability $1-\exp(-\Theta(n))$, the next population P_{t+1} will cover at most $|f(P_{t+1})| \leq (1-\varepsilon)(n+1)$ points on the front. Hence when a population covers a large fraction of the Pareto front, then with very high probability the next population will cover fewer points on the front. When the coverage is smaller, that is, $|f(P_t)| \leq (1-\varepsilon)(n+1)$, then with probability $1-\exp(-\Theta(n))$ the combined parent and offspring population R_t will miss a constant fraction of the Pareto front. From these two statements, it is easy to see that there is a constant δ such that with probability $1-\exp(-\Omega(n))$, in none of the first $\exp(\Omega(n))$ iterations the combined parent and offspring population covers more than $(1-\delta)(n+1)$ points of the Pareto front.

Since it is the technically easier one, we start with proving the latter statement that a constant fraction of the front not covered by P_t implies also a constant fraction not covered by R_t. Before stating the formal result and proof, let us explain the reason behind this result. With a constant fraction of the front not covered by P_t, also a constant fraction that is $\Omega(n)$ away from the boundary points $(0,n)$ and $(n,0)$ is not covered. These values have the property that from an individual corresponding to either of their neighboring positions, an individual with this objective value can only be generated with constant probability via one-bit mutation. Again a constant fraction of these values have only a constant number of individuals on neighboring positions. These values thus have a (small) constant probability of not being generated in this iteration. This shows that in expectation, we are still missing a constant fraction of the Pareto front in R_t. Via the method of bounded differences (exploiting that each mutation operation can change the number of missing elements by at most one), we turn this expectation into a bound that holds with probability $1-\exp(-\Omega(n))$.

Lemma 10. *Let $\varepsilon \in (0,1)$ be a sufficiently small constant. Consider optimizing the* OneMinMax *benchmark via the NSGA-II applying one-bit mutation once to each parent individual. Let the population size be $N = n+1$. Assume that $|f(P_t)| \leq (1-\varepsilon)(n+1)$. Then with probability at least $1-\exp(-\Omega(n))$, we have $|f(R_t)| \leq (1-\frac{1}{10}\varepsilon(\frac{1}{5}\varepsilon-\frac{2}{n})^{5/\varepsilon})(n+1)$.*

Proof. Let $F = \{(v, n-v) \mid v \in [0..n]\}$ be the Pareto front of OneMinMax. For a value $(v, n-v) \in F$, we say that $(v-1, n-v+1)$ and $(v+1, n-v-1)$ are neighbors of $(v, n-v)$ provided that they are in $[0..n]^2$. We write $(a,b) \sim (u,v)$ to denote that (a,b) and (u,v) are neighbors.

Let $\Delta = \lceil \frac{5}{\varepsilon} \rceil - 1$ and let F' be the set of values in F such that more than Δ individuals in P_t have a function value that is a neighbor of this value, that is,

$$F' = \left\{(v, n-v) \in F \,\middle|\, |\{x \in P_t \mid f(x) \sim (v, n-v)\}| \geq \Delta + 1\right\}.$$

Then $|F'| \leq \frac{2}{\Delta+1}(n+1) \leq \frac{2}{5}\varepsilon(n+1)$ as otherwise the number of individuals in our population could be bounded from below by

$$|F'|\tfrac{1}{2}(\Delta+1) > \tfrac{2}{\Delta+1}(n+1) \cdot \tfrac{1}{2}(\Delta+1) = n+1,$$

which contradicts our assumption $N = n+1$ (note that the factor of $\frac{1}{2}$ accounts for the fact that we may count each individual twice).

Let $M = F \setminus f(P_t)$ be the set of Pareto front values not covered by the current population. By assumption, $|M| \geq \varepsilon(n+1)$. Let

$$M_1 = \left\{(v, n-v) \in M \,\middle|\, v \in \left[\lfloor \tfrac{1}{5}\varepsilon(n+1)\rfloor..n - \lfloor \tfrac{1}{5}\varepsilon(n+1)\rfloor\right]\right\} \setminus F'.$$

Then $|M_1| \geq |M| - 2\lfloor \frac{1}{5}\varepsilon(n+1)\rfloor - |F'| \geq \frac{1}{5}\varepsilon(n+1)$.

We now argue that a constant fraction of the values in M_1 is not generated in the current generation. We note that via one-bit mutation, a given $(v, n-v) \in F$ can only be generated from an individual x with $f(x) \sim (v, n-v)$. Let $(v, n-v) \in M_1$. Since $v \in [\lfloor \frac{1}{5}\varepsilon(n+1)\rfloor..n - \lfloor \frac{1}{5}\varepsilon(n+1)\rfloor]$, the probability that a given parent x is mutated to some individual y with $f(y) = (v, n-v)$ is at most

$$\frac{n - \lfloor \frac{1}{5}\varepsilon(n+1)\rfloor + 1}{n} \leq 1 - \frac{1}{5}\varepsilon + \frac{2}{n}$$

since there are at most $n - \lfloor \frac{1}{5}\varepsilon(n+1)\rfloor + 1$ bit positions such that flipping them creates the desired value. Since $v \notin F'$, the probability that Q_t (and thus R_t) contains no individual y with $f(y) = (v, n-v)$, is at least

$$\left(1 - \left(1 - \tfrac{1}{5}\varepsilon + \tfrac{2}{n}\right)\right)^{\Delta} \geq (\tfrac{1}{5}\varepsilon - \tfrac{2}{n})^{5/\varepsilon} := p.$$

Let $X = |F \setminus f(R_t)|$ denote the number of Pareto front values not covered by R_t. We have $E[X] \geq |M_1| p \geq \frac{1}{5}\varepsilon p(n+1)$. The random variable X is functionally dependent on the $N = n+1$ random decisions of the N mutation operations, which are stochastically independent. Changing the outcome of a single mutation operation changes X by at most 1. Consequently, X satisfies the assumptions of the method of bounded differences [38] (also to be found in [18, Theorem 1.10.27]). Hence the classic additive Chernoff bound applies to X as if it was a sum of N independent random variables taking values in an interval of length 1. In particular, the probability that $X \leq \frac{1}{10}\varepsilon p(n+1) \leq \frac{1}{2}E[X]$ is at most $\exp(-\Omega(n))$. □

We now turn to the other main argument, which is that when the current population covers the Pareto front to a large extent, then the selection procedure of the NSGA-II will remove individuals in such a way from R_t that at least some constant fraction of the Pareto front is not covered by P_{t+1}. The key arguments to show this claim are the following. When a large part of the front is covered by P_t, then many points are only covered by a single individual (since the population size equals the size of the front). With some careful counting, we derive from this that close to two thirds of the positions on the front are covered exactly twice in the combined parent and offspring population R_t and that the corresponding individuals have the same crowding distance. Since these are roughly $\frac{4}{3}(n+1)$ individuals appearing equally preferable in the selection, a random set of at least roughly $\frac{1}{3}(n+1)$ of them will be removed in the selection step. In expectation, this will remove both individuals from a constant fraction of the points on the Pareto front. Again, the method of bounded differences turns this expectation into a statement with probability $1-\exp(-\Omega(n))$.

Lemma 11. *Let $\varepsilon > 0$ be a sufficiently small constant. Consider optimizing the* OneMinMax *benchmark via the NSGA-II applying one-bit mutation once to each individual. Let the population size be $N = n+1$. Assume that the current population P_t covers $|f(P_t)| \geq (1-\varepsilon)(n+1)$ elements of the Pareto front. Then with probability at least $1-\exp(-\Omega(n))$, the next population P_{t+1} covers less than $(1-0.01)(n+1)$ elements of the Pareto front.*

Proof. Let U be the set of Pareto front values that have exactly one corresponding individual in P_t, that is, for any $(v, n-v) \in U$, there exists only one $x \in P_t$ with $f(x) = (v, n-v)$. We first note that $|U| \geq (1-2\varepsilon)(n+1)$ as otherwise there would be at least

$$\begin{aligned} &2(|f(P_t)| - |U|) + |U| = 2|f(P_t)| - |U| \\ &\quad > 2(1-\varepsilon)(n+1) - (1-2\varepsilon)(n+1) = n+1 \end{aligned}$$

individuals in P_t, which contradicts our assumption $N = n+1$.

Let U' denote the set of values in U which have all their neighbors also in U. Since each value not in U can prevent at most two values in U from being in U', we have

$$\begin{aligned} |U'| &\geq |U| - 2(n+1-|U|) = 3|U| - 2(n+1) \\ &\geq 3(1-2\varepsilon)(n+1) - 2(n+1) = (1-6\varepsilon)(n+1). \end{aligned}$$

We say that $(v, n-v)$ is double-covered by R_t if there are exactly two individuals in R_t with function value $(v, n-v)$. Noting that via one-bit mutation a certain function value can only be generated from the individuals corresponding to the neighbors of this function value, we see that a given $(v, n-v) \in U'$ with $v \in [1..n-1]$ is double-covered by R_t with probability exactly

$$p_v = \frac{n-(v-1)}{n} + \frac{v+1}{n} - 2\frac{n-(v-1)}{n}\frac{v+1}{n} = 1 - \frac{2v}{n} + 2\frac{v^2-1}{n^2}.$$

Thus the expected number of double-coverages in U' is at least

$$\begin{aligned} \sum_{\substack{v\in[1..n-1]:\\(v,n-v)\in U'}} p_v &= \left(\sum_{v=1}^{n-1} p_v\right) - \left(\sum_{\substack{v\in[1..n-1]:\\(v,n-v)\notin U'}} p_v\right) \\ &\geq \left(\sum_{v=1}^{n-1} 1 - \frac{2v}{n} + 2\frac{v^2-1}{n^2}\right) - \left(\sum_{\substack{v\in[1..n-1]:\\(v,n-v)\notin U'}} 1\right) \\ &\geq (n-1) - \frac{2}{n}\frac{(n-1)n}{2} + \frac{2}{n^2}\left(\frac{(n-1)n(2(n-1)+1)}{6} - (n-1)\right) \\ &\quad - 6\varepsilon(n+1) \\ &= \frac{n-1}{n^2}\frac{2n^2-n-6}{3} - 6\varepsilon(n+1) = (\tfrac{2}{3} - 6\varepsilon)(n+1) - O(1). \end{aligned}$$

Denote by U'' the set of values in U' that are double-covered by R_t and note that we have just shown $E[|U''|] \geq (\frac{2}{3} - 6\varepsilon)(n+1) - O(1)$. The number $m := |U''|$ of double-covered elements is functionally dependent on the random decisions taken (independently) in the N mutation operations. Each mutation operation determines one offspring and thus can change the number of double-covered values by at most 2. Consequently, we can use the method of bounded differences [38] and obtain that $|U''|$ is at least $(\frac{2}{3} - 8\varepsilon)(n+1)$ with probability at least $1-\exp(-\Omega(n))$. We condition on this in the remainder.

Our next argument is that these double-coverages correspond to approximately $\frac{4}{3}(n+1)$ individuals in R_t that have the same crowding distance. Consequently, the selection procedure has to discard at least roughly $\frac{1}{3}(n+1)$ of them, randomly chosen, and this will lead to a decent number of values in U'' that are not covered anymore by P_{t+1}.

To make this precise, let R'' denote the individuals x in R_t such that $f(x) \in U''$. By construction, there are exactly two such individuals for each value in U'', hence $|R''| = 2m$. Further, both neighboring values are also present in $f(R_t)$. Consequently, each $x \in R''$ has crowding distance (in R_t) exactly $d = \frac{1}{v_1^{\max} - v_1^{\min}} + \frac{1}{v_2^{\max} - v_2^{\min}}$. We recall that the selection procedure (since all ranks are equal to one) first discards all individuals with crowding distance less than d since these are at most $|R_t| - |R''| \leq 2(n+1-\tilde{m}) = (2 - \frac{4}{3} + 16\varepsilon)(n+1) + O(1)$, which is less than N for n large and ε small enough. Then, randomly, the selection procedure discards a further number of individuals from all individuals with crowding distance exactly d so that exactly N individuals remain. For N individuals to remain, we need that at least $k := |R''| - N$ individuals from R'' are discarded.

To ease the calculation, we first reduce the problem to the case that $|R''| = 2\tilde{m}$. Indeed, let U''' be any subset of U'' having cardinality exactly $\tilde{m}$ and let R''' be the set of individuals $x \in R_t$ with $f(x) \in U'''$. Then $R''' \subseteq R''$ and $|R'''| = 2\tilde{m}$. With the same argument as in the previous paragraph we see that the selection procedure has to remove at least $\tilde{k} := 2\tilde{m} - N$ elements from R'''. We thus analyze the number of elements of U''' that become uncovered when we remove a random set of $\tilde{k}$ individuals from R''', knowing that this is a lower bound for the number of elements uncovered in U'', both because the number of individuals removed from R''' can be higher than $\tilde{k}$ and because the removal of elements in $R'' \setminus R'''$ can also lead to uncovered elements in U''.

We take a final pessimistic simplification, and this is that we select $\tilde{k}$ elements from R''' with replacement and remove these individuals from R'''. Clearly, this can only lower the number of removed elements, hence our estimate for the number of uncovered elements is also valid for the random experiment without replacement (where we choose exactly $\tilde{k}$ elements to be removed).

For this random experiment the probability for uncovering a position in U''' is at least

$$\begin{aligned}
&1 - 2\left(1 - \frac{1}{2\tilde{m}}\right)^{\tilde{k}} + \left(1 - \frac{1}{2\tilde{m}}\right)^{2\tilde{k}} \\
&= 1 - 2\exp\left(-\frac{\tilde{k}}{2\tilde{m}}\right) + \exp\left(-\frac{\tilde{k}}{\tilde{m}}\right) - O\left(\frac{1}{n}\right) \\
&\geq 1 - 2\exp\left(-1 + \frac{3}{4}\frac{1}{1-12\varepsilon}\right) + \exp\left(-2 + \frac{6}{4}\frac{1}{1-12\varepsilon}\right) - O\left(\frac{1}{n}\right) := p,
\end{aligned}$$

where we used the estimate $\frac{\tilde{k}}{2\tilde{m}} = 1 - \frac{n+1}{2\tilde{m}} = 1 - \frac{3}{4}\frac{1}{1-12\varepsilon}$ and the fact that $\tilde{m} = \Theta(n)$.

Let Y denote the number of elements of U''' uncovered in our random experiment. We note that $1 - 2\exp(-1/4) + \exp(-1/2) \geq 0.04892$. Hence when n is large enough and ε was chosen as a sufficiently small constant, then

$$E[Y] = p\tilde{m} \geq 0.02(n+1).$$

The random variable Y is functionally dependent on the $\tilde{k}$ selected individuals, which are stochastically independent. Changing the outcome of a single selected individual changes Y by at most 1. Consequently, Y satisfies the assumptions of the method of bounded differences [38]. The classic additive Chernoff bound thus applies to Y as if it was a sum of $k = \Omega(n)$ independent random variables taking values in an interval of length 1. In particular, the probability that $Y \leq 0.01(n+1) \leq \frac{1}{2}E[Y]$ is at most $\exp(-\Omega(n))$. □

Combining Lemmas 10 and 11, we have the following exponential runtime result.

Theorem 12. *Consider optimizing* OneMinMax *via the NSGA-II applying one-bit mutation once to each individual. Let the population size be $N = n+1$. There are a positive constant γ and a time $T = \exp(\Omega(n))$ such that with probability $1 - \exp(-\Omega(n))$, in each of the first T iterations at most a fraction of $1-\gamma$ of the Pareto front is covered by P_t.*

Proof. Let ε be a small constant rendering the claims of Lemmas 10 and 11 valid. Assume that n is sufficiently large. Let $\tilde{\varepsilon} = (\frac{1}{10}\varepsilon)^{5/\varepsilon+1}$. By a simple Chernoff bound, we note that a random initial individual x satisfies $\frac{1}{4}n \leq f_1(x) \leq \frac{3}{4}n$ with probability $1 - \exp(\Omega(n))$. Taking a union bound over the $n+1$ initial individuals, we see that the initial population P_0 with probability $1 - \exp(-\Omega(n))$ covers at most half of the Pareto front.

Let t be some iteration. If $|f(P_t)| \geq (1-\varepsilon)(n+1)$, then by Lemma 11 with probability $1 - \exp(-\Omega(n))$ the next population P_{t+1} covers less than $(1 - 0.01)(n+1)$ values of the Pareto front. If $|f(P_t)| \leq (1-\varepsilon)(n+1)$, then by Lemma 10 with probability $1 - \exp(-\Omega(n))$ we have $n+1-|f(P_{t+1})| \geq \frac{1}{10}\varepsilon(\frac{1}{5}\varepsilon - \frac{2}{n})^{5/\varepsilon}(n+1) \geq \tilde{\varepsilon}(n+1)$, where the last estimate holds when n is sufficiently large. Consequently, for each generation t, the probability that P_t covers more than $(1 - \min\{\tilde{\varepsilon}, 0.01\})(n+1)$ values of the Pareto front, is only $\exp(-\Omega(n))$. In particular, a union bound shows that for $T = \exp(\Theta(n))$ suitably chosen, with probability $1 - \exp(-\Omega(n))$ in all of the first T iterations, the population covers at most $(1 - \min\{\tilde{\varepsilon}, 0.01\})(n+1)$ values of the Pareto front. □

6. Experiments

To complement our asymptotic results with runtime data for concrete problem sizes, we conducted the following experiments.

6.1. Settings

We use, in principle, the version of the NSGA-II given by Deb (Revision 1.1.6), available at [14], except that, as in our theoretical analysis, we do not use crossover. We re-implemented the algorithm in Matlab (R2016b). When a sorting procedure is used, we use the one provided by Matlab (and not randomized Quicksort as in Deb's implementation). The code is available at [56].

Our theoretical analysis above covers four parent selection strategies and two mutation operators. In the interest of brevity, with the exception of the data presented in Fig. 3 we concentrate in our experiments on one variant of the algorithm, namely we use two-permutation binary tournament selection (as proposed in [21]) and standard bit-wise mutation with mutation rate $\frac{1}{n}$ (which is the most common mutation operator in evolutionary computation). We use the following experimental settings.

- Problem size n: 100, 200, 300, and 400 for OneMinMax, and 30, 60, 90, and 120 for LeadingOnesTrailingZeroes.
- Population size N: Our theoretical analyses (Theorems 6 and 9) showed that the NSGA-II find the optima of OneMinMax and LeadingOnesTrailingZeroes efficiently for population sizes of at least $N^* = 4(n+1)$. We use this value also in the experiments. We also use the value $N = 2N^*$, for which our theory results apply, but our runtime guarantees are twice as large as for N^* (when making the implicit constants in the results visible). We also use the smaller population sizes $2(n+1)$ and $1.5(n+1)$ for OneMinMax and $2(n+1)$ for LeadingOnesTrailingZeroes. For these values, we have no proven result, but it is not uncommon that mathematical runtime analyses cannot cover all efficient parameter setting, and in fact, we shall observe a good performance in these experiments as well (the reason why we do not display results for $N = 1.5(n+1)$ for LeadingOnesTrailingZeroes is that here indeed the algorithm was not effective anymore). Finally, we conduct experiments with the population size $N = n+1$, which is large enough to represent the full Pareto front, but for which we have proven the NSGA-II to be ineffective (on OneMinMax and when letting each parent create an offspring via one-bit mutation).
- Number of independent runs: 50 for the efficient population sizes in Section 6.2 and 20 for more time-consuming experiments with inefficient population sizes in Sections 6.3 to 6.4. These numbers of independent runs have already shown good concentrations.

6.2. Efficient population sizes

Fig. 1 displays the runtime (that is, the number of fitness evaluations until the full Pareto front is covered) of the NSGA-II with population sizes large enough to allow an efficient optimization, together with the runtime of the (parameter-less) GSEMO.

This data confirms that the NSGA-II can efficiently cover the Pareto front of OneMinMax and LeadingOnesTrailingZeroes when using a population size of at least N^*. The runtimes for $N = 2N^*$ are clearly larger than for N^*, but by a factor slightly less than 2 for both problems. The data for the population sizes smaller than N^* indicates that also for these parameter settings the NSGA-II performs very well.

Comparing the NSGA-II to the GSEMO, we observe that the NSGA-II with a proper choice of the population size shows a better performance. This is interesting and somewhat unexpected, in particular, for simple problems like OneMinMax and LeadingOnesTrailingZeroes. It is clear that the NSGA-II using tournament selection chooses extremal parents with higher rate. More precisely, each individual appears twice in a tournament. For an extremal value on the Pareto front, at least one individual has an infinite crowding distance, making it the tournament winner almost surely (except in the rare case that the tournament partner has infinite crowding distance as well). Consequently, for each extremal objective value, the NSGA-II mutates at least $2 - o(1)$ individuals per iteration. This is twice the average rate. In contrast, the GSEMO treats all individuals equally. This advantage of the NSGA-II comes at the price of a larger population, hence a larger cost per iteration. We note that the NSGA-II throughout the run works with a population of size N, whereas the GSEMO only keeps non-dominated individuals in its population. Consequently, in particular in the early stages of the optimization process, each iteration takes significantly fewer fitness evaluations.

6.3. Inefficient population sizes

When the population size is small, we do not have the result that points on the front cannot be lost (Lemmas 1 and 7) and the proof of Theorem 12 shows that indeed we can easily lose points on the front, leading to a runtime at least exponential in n when $N = n+1$. In this subsection, we analyze this phenomenon experimentally. As discussed earlier, we first concentrate on the NSGA-II with two-permutation tournament selection and standard bit-wise mutation.

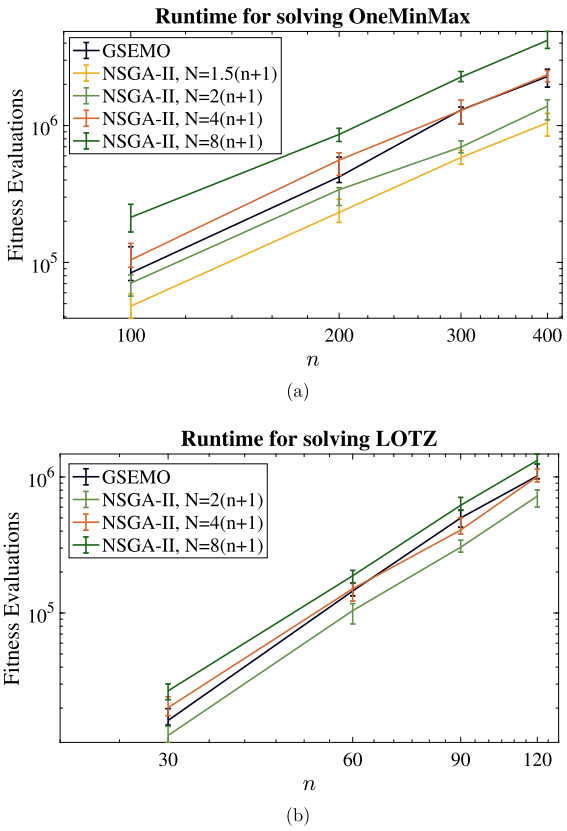


Fig. 1. The number of function evaluations for the NSGA-II (binary tournament selection, standard bit-wise mutation) with different population sizes and for the GSEMO optimizing OneMinMax (1a) and LeadingOnesTrailingZeroes (1b). Displayed are the median (with 1st and 3rd quartiles) in 50 independent runs. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Since it is hard to show an exponential runtime experimentally, we do not run the algorithm until it found the full Pareto front (this would be possible only for very small problem sizes), but we conduct a slightly different experiment for reasonable problem sizes which also strongly indicates that the NSGA-II has enormous difficulties in finding the full front. We ran the NSGA-II for 3000 generations for OneMinMax and 5000 generations for LeadingOnesTrailingZeroes and measured for each generation the ratio by which the Pareto front is covered. This data is displayed in Fig. 2. We see clearly that the coverage of the Pareto front steeply increases at first, but then stagnates at a constant fraction clearly below one (around 80% for OneMinMax and between 50% and 60% for LeadingOnesTrailingZeroes) and this in a very concentrated manner. From this data, there is no indication that the Pareto front will be covered anytime soon.

We said in Section 5 that we were optimistic that our negative result for small population size would also hold for all other variants of the NSGA-II. To experimentally support this claim, we now run all variants of the NSGA-II discussed in this work on OneMinMax with problem size $n = 200$, 20 times for 3000 iterations. In Fig. 3, we see the ratios of the coverage of the Pareto front by the populations in the 20 runs and in iterations [2001..3000] (that is, we regard together $20 * 1000$ populations). We see that all variants fail to cover a constant fraction of the Pareto. The precise constant is different for each variant. Most notable, we observe that the variants using standard bit-wise mutation cover the Pareto front to a lesser extent than those building on one-bit mutation. We do not have a clear explanation for this phenomenon, but we speculate that standard bit-wise mutation is harmed by its constant fraction of mutations that just create a copy of the parent. We would, however, not interpret the results in this figure as a suggestion to prefer one-bit mutation. As shown in [22], with high probability the NSGA-II using one-bit mutation fails to find the Pareto front of the OneJumpZeroJump benchmark, regardless of the runtime allowed.

6.4. Optimization with small population sizes

In the previous subsection, we showed that the NSGA-II with population size equal to the size of the Pareto front cannot cover the full Pareto front in a reasonable time. On the positive side, however, still a large fraction of the Pareto front was covered, e.g., around 80% for the OneMinMax problem. This could indicate that the NSGA-II also with smaller population sizes is an interesting algorithm. This is what we briefly discuss now. We shall not explore this question in full detail, but

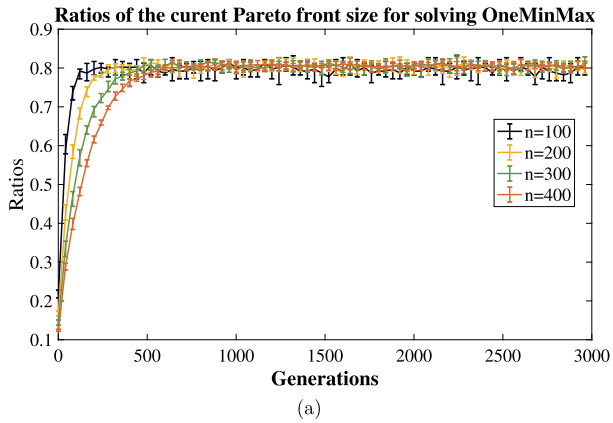


Fig. 2. Ratio of the coverage of the Pareto front by the current population of the NSGA-II (binary tournament selection, standard bit-wise mutation) with population size $N = n + 1$ for solving OneMinMax (2a) and LeadingOnesTrailingZeroes (2b). Displayed are the median (with 1st and 3rd quartiles) in 20 independent runs.

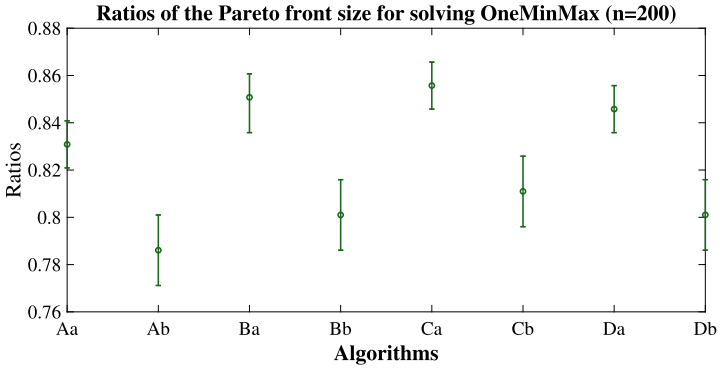


Fig. 3. Ratios of the coverage of the Pareto front by the population of the different NSGA-II variants (using A (selecting each individual as a parent once), B (N times choosing a parent uniformly at random), C (independent binary tournaments), or D (two-permutation binary tournaments) as the mating selection strategy, and using a (one-bit mutation) or b (standard bit-wise mutation) as the mutation strategy) with population size $N = n + 1$ on the OneMinMax with problem size $n = 200$. Displayed are the median (with 1st and 3rd quartiles) in 20 independent runs and [2001..3000] generations.

only to the extent that we observe a good indication that the NSGA-II performs well also with small population sizes. We note that the subsequent work [54] took up this research question and discussed it in detail.

To understand how well the NSGA-II performs with small population size $n + 1$, we first regard how fast its population spreads out on the Pareto front. From the data in Fig. 4, we see that also with this small population size, the NSGA-II quickly finds the two extremal points $(0, n)$ and $(n, 0)$ of the Pareto front. This fits our understanding of the algorithms. Since the two outer-most individuals in the population have infinite crowding distance and since there are at most four individuals with infinite crowding distance, these individuals will never be lost, even if the population size is relatively small.

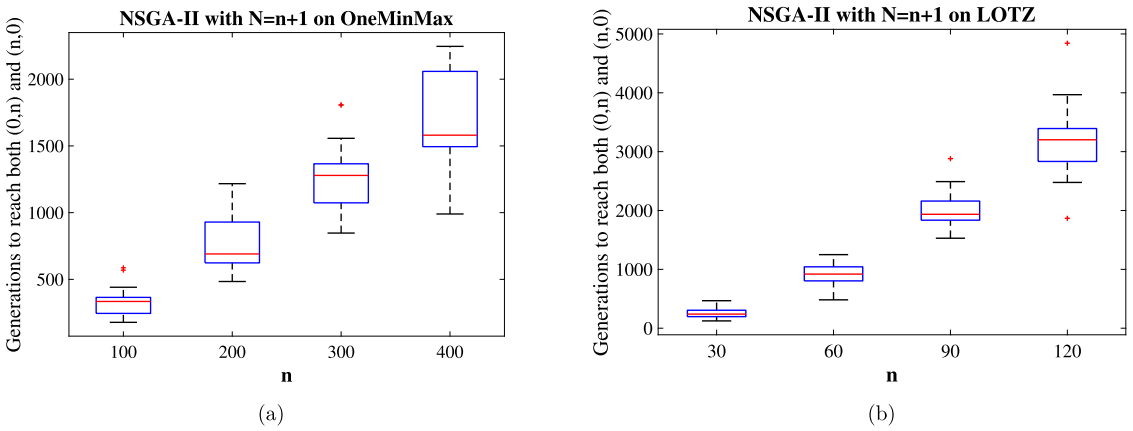


Fig. 4. First generation when both extreme function values $(0, n)$ and $(n, 0)$ were contained in the population of the NSGA-II (binary tournament selection, standard bit-wise mutation, population size $N = n + 1$) for OneMinMax (4a) and LeadingOnesTrailingZeroes (4b).

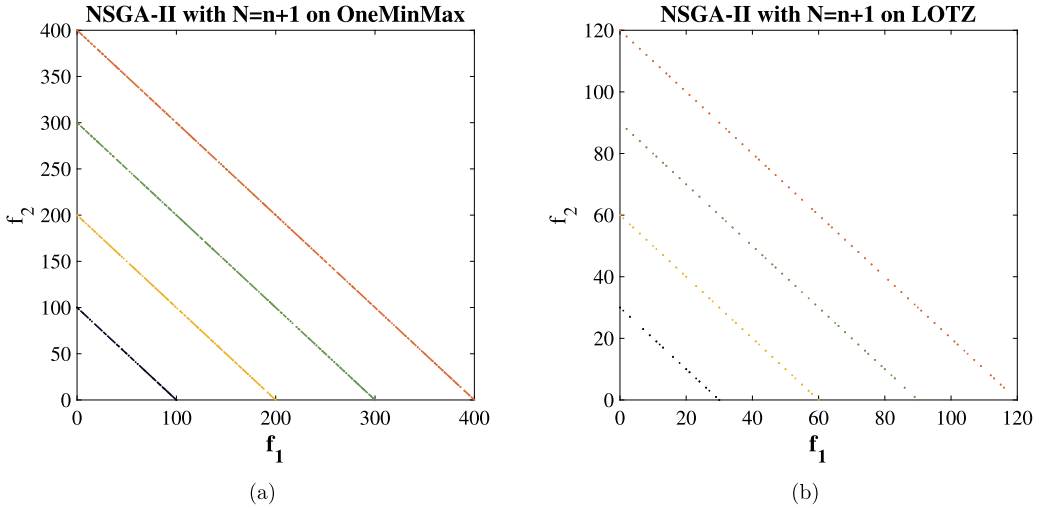


Fig. 5. The function values of the population P_t for $t = 3000$ when optimizing OneMinMax (5a) and for $t = 5000$ when optimizing LeadingOnesTrailingZeroes (5b) via the NSGA-II (binary tournament selection, standard bit-wise mutation, population size $N = n + 1$) in one typical run. Both plots show that this population size is not sufficient to completely cover the Pareto front, but it suffices to approximate very well the front. Different colors are for different problem sizes n, and $n = \{100, 200, 300, 400\}$ for OneMinMax and $n = \{30, 60, 90, 120\}$ for LeadingOnesTrailingZeroes. Also note that the Pareto front is $\{(i, n - i) \mid i \in [0..n]\}$.

More interesting is the question how evenly the population is distributed on the Pareto front once the two extremal points are found. To this aim, we display in Fig. 5 the function values of the populations after a moderate runtime in a run of the NSGA-II. In all eight datasets, the complete Pareto front was not found (as expected). However, the plots also show that in all cases, the front is well approximated by the population. Also, we note that the population contains only individuals on the Pareto front (which is trivially satisfied for OneMinMax, but not so for LeadingOnesTrailingZeroes). We note that the data from two individual runs displayed in the figure is representative. In all runs we never encountered an interval of uncovered points of length longer than 6 and 4 respectively.

7. Conclusion

In this work, we conducted the first mathematical runtime analysis of the NSGA-II, which is the predominant framework in real-world multi-objective optimization. We proved that with a suitable population size, all variants of the NSGA-II regarded in this work satisfy the same asymptotic runtime guarantees as the previously regarded much simpler SEMO, GSEMO, and $(\mu + 1)$ SIBEA when optimizing the two benchmarks OneMinMax and LeadingOnesTrailingZeroes. The choice of the population size is important. We proved an exponential runtime when the population size equals the size of the Pareto front.

On the technical side, this paper shows that mathematical runtime analyses are feasible also for the NSGA-II. We provided a number of arguments to cope with the challenges imposed by this algorithm, in particular, the fact that points in the

Pareto front can be lost and the parent selection via binary tournaments based on the rank and crowding distance. We are optimistic that these tools will aid future analyses of the NSGA-II (and in fact, they have already been used several times in subsequent work, see the discussion in the introduction).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 62306086), Science, Technology and Innovation Commission of Shenzhen Municipality (Grant No. GXWD20220818191018001), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177).

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

References

[1] Anne Auger, Benjamin Doerr (Eds.), Theory of Randomized Search Heuristics, World Scientific Publishing, 2011.
[2] Dimo Brockhoff, Tobias Friedrich, Frank Neumann, Analyzing hypervolume indicator based algorithms, in: Parallel Problem Solving from Nature, PPSN 2008, Springer, 2008, pp. 651–660.
[3] Chao Bian, Chao Feng, Chao Qian, Yang Yu, An efficient evolutionary algorithm for subset selection with general cost constraints, in: Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 3267–3274.
[4] Nicola Beume, Boris Naujoks, Michael Emmerich, SMS-EMOA: multiobjective selection based on dominated hypervolume, Eur. J. Oper. Res. 181 (2007) 1653–1669.
[5] Chao Bian, Chao Qian, 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, Springer, 2022, pp. 428–441.
[6] Chao Bian, Chao Qian, Ke Tang, A general approach to running time analysis of multi-objective evolutionary algorithms, in: International Joint Conference on Artificial Intelligence, IJCAI 2018, IJCAI, 2018, pp. 1405–1411.
[7] Chao Bian, Yawen Zhou, Miqing Li, Chao Qian, Stochastic population update can provably be helpful in multi-objective evolutionary algorithms, in: International Joint Conference on Artificial Intelligence, IJCAI 2023, ijcai.org, 2023, pp. 5513–5521.
[8] Sacha Cerf, Benjamin Doerr, Benjamin Hebras, Jakob Kahane, Simon Wietheger, 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, ijcai.org, 2023, pp. 5522–5530.
[9] Edgar Covantes Osuna, Wanru Gao, Frank Neumann, Dirk Sudholt, Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation, Theor. Comput. Sci. 832 (2020) 123–142.
[10] Victoria G. Crawford, An efficient evolutionary algorithm for minimum cost submodular cover, in: International Joint Conference on Artificial Intelligence, IJCAI 2019, ijcai.org, 2019, pp. 1227–1233.
[11] Victoria G. Crawford, Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions, in: International Joint Conference on Artificial Intelligence, IJCAI 2021, ijcai.org, 2021, pp. 1661–1667.
[12] Benjamin Doerr, Carola Doerr, Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm, Algorithmica 80 (2018) 1658–1709.
[13] Benjamin Doerr, Carola Doerr, Aneta Neumann, Frank Neumann, Andrew M. Sutton, Optimization of chance-constrained submodular functions, in: Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 1460–1467.
[14] Kalyanmoy Deb's implementation of the NSGA-II, https://www.egr.msu.edu/~kdeb/codes.shtml.
[15] Benjamin Doerr, Wanru Gao, Frank Neumann, Runtime analysis of evolutionary diversity maximization for OneMinMax, in: Genetic and Evolutionary Computation Conference, GECCO 2016, ACM, 2016, pp. 557–564.
[16] Benjamin Doerr, Frank Neumann (Eds.), Theory of Evolutionary Computation—Recent Developments in Discrete Optimization, Springer, 2020. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.
[17] Benjamin Doerr, Analyzing randomized search heuristics via stochastic domination, Theor. Comput. Sci. 773 (2019) 115–137.
[18] Benjamin Doerr, Probabilistic tools for the analysis of randomized optimization heuristics, in: Benjamin Doerr, Frank Neumann (Eds.), Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 1–87. Also available at https://arxiv.org/abs/1801.06733.
[19] Duc-Cuong Dang, Andre Opris, Bahare Salehi, Dirk Sudholt, Analysing the robustness of NSGA-II under noise, in: Genetic and Evolutionary Computation Conference, GECCO 2023, ACM, 2023, pp. 642–651.
[20] Duc-Cuong Dang, Andre Opris, Bahare Salehi, Dirk Sudholt, A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation, in: Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 12390–12398.
[21] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.
[22] Benjamin Doerr, Zhongdi Qu, A first runtime analysis of the NSGA-II on a multimodal problem, IEEE Trans. Evol. Comput. (2023), https://doi.org/10.1109/TEVC.2023.3250552.
[23] Benjamin Doerr, Zhongdi Qu, From understanding the population dynamics of the NSGA-II to the first proven lower bounds, in: Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 12408–12416.
[24] Benjamin Doerr, Zhongdi Qu, Runtime analysis for the NSGA-II: provable speed-ups from crossover, in: Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 12399–12407.

[25] Benjamin Doerr, Weijie Zheng, Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives, in: Conference on Artificial Intelligence, AAAI 2021, AAAI Press, 2021, pp. 12293–12301.
[26] Tobias Friedrich, Jun He, Nils Hebbinghaus, Frank Neumann, Carsten Witt, Approximating covering problems by randomized search heuristics using multi-objective models, Evol. Comput. 18 (2010) 617–633.
[27] Tobias Friedrich, Frank Neumann, Maximizing submodular functions under matroid constraints by evolutionary algorithms, Evol. Comput. 23 (2015) 543–558.
[28] David E. Goldberg, Kalyanmoy Deb, A comparative analysis of selection schemes used in genetic algorithms, in: Foundations of Genetic Algorithms, FOGA 1990, Morgan Kaufmann, 1990, pp. 69–93.
[29] Oliver Giel, Expected runtimes of a simple multi-objective evolutionary algorithm, in: Congress on Evolutionary Computation, CEC 2003, IEEE, 2003, pp. 1918–1925.
[30] Oliver Giel, Per Kristian Lehre, On the effect of populations in evolutionary multi-objective optimisation, Evol. Comput. 18 (2010) 335–356.
[31] Zhengxin Huang, Yuren Zhou, Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework, in: Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 2359–2366.
[32] Zhengxin Huang, Yuren Zhou, Zefeng Chen, Xiaoyu He, Running time analysis of MOEA/D with crossover on discrete optimization problem, in: Conference on Artificial Intelligence, AAAI 2019, AAAI Press, 2019, pp. 2296–2303.
[33] Thomas Jansen, Analyzing Evolutionary Algorithms – The Computer Science Perspective, Springer, 2013.
[34] Saku Kukkonen, Kalyanmoy Deb, Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems, in: Conference on Evolutionary Computation, CEC 2006, IEEE, 2006, pp. 1179–1186.
[35] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, Kalyanmoy Deb, Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem, in: Parallel Problem Solving from Nature, PPSN 2002, Springer, 2002, pp. 44–53.
[36] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions, IEEE Trans. Evol. Comput. 8 (2004) 170–182.
[37] Yuan-Long Li, Yu-Ren Zhou, Zhi-Hui Zhan, Jun Zhang, A primary theoretical study on decomposition-based multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 20 (2016) 563–576.
[38] Colin McDiarmid, On the method of bounded differences, in: Surveys in Combinatorics, Cambridge Univ. Press, 1989, pp. 48–118.
[39] Frank Neumann, Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem, Eur. J. Oper. Res. 181 (2007) 1620–1629.
[40] Frank Neumann, Joachim Reichel, Martin Skutella, Computing minimum cuts by randomized search heuristics, Algorithmica 59 (2011) 323–342.
[41] Anh Quang Nguyen, Andrew M. Sutton, Frank Neumann, Population size matters: rigorous runtime results for maximizing the hypervolume indicator, Theor. Comput. Sci. 561 (2015) 24–36.
[42] Frank Neumann, Madeleine Theile, How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem, in: Parallel Problem Solving from Nature, PPSN 2010, Part I, Springer, 2010, pp. 667–676.
[43] Frank Neumann, Carsten Witt, Runtime analysis of a simple ant colony optimization algorithm, in: Algorithms and Computation, ISAAC 2006, Springer, 2006, pp. 618–627.
[44] Frank Neumann, Carsten Witt, Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity, Springer, 2010.
[45] Frank Neumann, Carsten Witt, Runtime analysis of single- and multi-objective evolutionary algorithms for chance constrained optimization problems with normally distributed random variables, in: International Joint Conference on Artificial Intelligence, IJCAI 2022, ijcai.org, 2022, pp. 4800–4806.
[46] Chao Qian, Chao Bian, Chao Feng, Subset selection by Pareto optimization with recombination, in: Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 2408–2415.
[47] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, On subset selection with general cost constraints, in: International Joint Conference on Artificial Intelligence, IJCAI 2017, ijcai.org, 2017, pp. 2613–2619.
[48] Chao Qian, Yang Yu, Ke Tang, Xin Yao, Zhi-Hua Zhou, Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms, Artif. Intell. 275 (2019) 279–294.
[49] Chao Qian, Yang Yu, Zhi-Hua Zhou, An analysis on recombination in multi-objective evolutionary optimization, Artif. Intell. 204 (2013) 99–119.
[50] Chao Qian, Yang Yu, Zhi-Hua Zhou, On constrained Boolean Pareto optimization, in: International Joint Conference on Artificial Intelligence, IJCAI 2015, AAAI Press, 2015, pp. 389–395.
[51] Vahid Roostapour, Aneta Neumann, Frank Neumann, Tobias Friedrich, Pareto optimization for subset selection with dynamic cost constraints, in: Conference on Artificial Intelligence, AAAI 2019, AAAI Press, 2019, pp. 2354–2361.
[52] Günter Rudolph, Evolutionary search for minimal elements in partially ordered finite sets, in: Evolutionary Programming, EP 1998, Springer, 1998, pp. 345–353.
[53] Simon Wietheger, Benjamin Doerr, A mathematical runtime analysis of the non-dominated sorting genetic algorithm III (NSGA-III), in: International Joint Conference on Artificial Intelligence, IJCAI 2023, ijcai.org, 2023, pp. 5657–5665.
[54] Weijie Zheng, Benjamin Doerr, Better approximation guarantees for the NSGA-II by using the current crowding distance, in: Genetic and Evolutionary Computation Conference, GECCO 2022, ACM, 2022, pp. 611–619.
[55] Weijie Zheng, Benjamin Doerr, Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for three or more objectives, IEEE Trans. Evol. Comput. (2023), https://doi.org/10.1109/TEVC.2023.3320278, also available at https://arxiv.org/2211.13084.
[56] Implementation of the NSGA-II in this paper, https://github.com/zhengwj13/NSGA_II_Clean.
[57] Weijie Zheng, Yufei Liu, Benjamin Doerr, A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II), in: Conference on Artificial Intelligence, AAAI 2022, AAAI Press, 2022, pp. 10408–10416.
[58] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, Qingfu Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (2011) 32–49.