Tight Runtime Guarantees From Understanding the Population Dynamics of the GSEMO Multi-Objective Evolutionary Algorithm

Benjamin Doerr[1] , **Martin S. Krejca**[1] and **Andre Opris**[2]

[1]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris
[2]University of Passau

{firstname.lastname}@polytechnique.edu, andre.opris@uni-passau.de

Abstract

The global simple evolutionary multi-objective optimizer (GSEMO) is a simple, yet often effective multi-objective evolutionary algorithm (MOEA). By only maintaining non-dominated solutions, it has a variable population size that automatically adjusts to the needs of the optimization process. The downside of the dynamic population size is that the population dynamics of this algorithm are harder to understand, resulting, e.g., in the fact that only sporadic tight runtime analyses exist. In this work, we significantly enhance our understanding of the dynamics of the GSEMO, in particular, for the classic CountingOnesCountingZeros (COCZ) benchmark. From this, we prove a lower bound of order $\Omega(n^2 \log n)$, for the first time matching the seminal upper bounds known for over twenty years. We also show that the GSEMO finds any constant fraction of the Pareto front in time $O(n^2)$, improving over the previous estimate of $O(n^2 \log n)$ for the time to find the first Pareto optimum. Our methods extend to other classic benchmarks and yield, e.g., the first $\Omega(n^{k+1})$ lower bound for the OJZJ benchmark in the case that the gap parameter is $k \in \{2, 3\}$. We are therefore optimistic that our new methods will be useful in future mathematical analyses of MOEAs.

1 Introduction

Many real-world optimization problems are characterized by several, often conflicting objectives. A common solution concept for such *multi-objective optimization problems* is to compute a diverse set of *Pareto optima* (solutions which cannot be improved in one objective without compromising in another objective) and let a human decision maker select one of these. Due to their population-based nature, *multi-objective evolutionary algorithms (MOEAs)* are among the most prominent approaches to such problems and have found applications in numerous subfields of multi-objective optimization [Coello *et al.*, 2007; Zhou *et al.*, 2011].

The mathematical runtime analysis of MOEAs was started around 20 years ago [Laumanns *et al.*, 2002; Giel, 2003; Thierens, 2003]. It has gained considerable momentum in the last years, among others, with analyses of classic algorithms such as the NSGA-II, NSGA-III, SMS-EMOA, and SPEA2 [Zheng *et al.*, 2022; Bian and Qian, 2022; Wietheger and Doerr, 2023; Bian *et al.*, 2023; Ren *et al.*, 2024; Opris *et al.*, 2024; Wietheger and Doerr, 2024; Alghouass *et al.*, 2025; Doerr *et al.*, 2025a; Deng *et al.*, 2025; Li *et al.*, 2025; Opris, 2025] or works discussing how MOEAs can solve submodular problems [Qian *et al.*, 2019; Qian *et al.*, 2020; Crawford, 2021; Do *et al.*, 2023].

The by far dominant algorithm in the rigorous analysis of MOEAs is the *global simple evolutionary multi-objective optimizer* [Giel, 2003] (GSEMO). Due to its apparent simplicity, it was the first MOEA for which mathematical runtime analyses were conducted, and it is still often the first algorithm for which new phenomena are discovered, see, e.g., [Dinot *et al.*, 2023; Dang *et al.*, 2024] for recent examples. At the same time, it is a central algorithm, and many other algorithms, in particular in the area of submodular optimization, build on it. For example, algorithms such as POSS (Pareto Optimization for Subset Selection) [Qian *et al.*, 2015], POMC (Pareto Optimization for Monotone Constraints) [Qian *et al.*, 2017], and POMS (Pareto Optimization for Multiset Selection) [Qian *et al.*, 2018] are all variants of the GSEMO applied to a suitable bi-objective formulation of the submodular problem of interest.

Despite this impressive body of theoretical works on the GSEMO, a real understanding of the working principles of this algorithm is still missing. This is most visible from the fact that there are almost no lower bounds matching the existing runtime guarantees (see Section 2 for a detailed discussion of this point). The reason is that for matching bounds, a deeper understanding of the population dynamics is necessary. This is particularly crucial for the GSEMO with its dynamic population size (note that the probability to choose a particular individual as parent is the reciprocal of the population size).

Our contribution: In this work, we greatly expand our understanding of the population dynamics of the GSEMO. To this end, we study how this algorithm optimizes the classic CountingOnesCountingZeros (COCZ) benchmark. For readers less familiar with the area of runtime analyses, we note that it is the established approach of this field to study how a specific randomized search heuristics solves a well-understood benchmark problem, and from this derive insights

into the working principles of the heuristic. All runtime analysis works cited above follow this approach.

We give more details on our new understanding of the population dynamics later (Section 4) when we have made precise the GSEMO algorithm and the COCZ benchmark, and now only describe two implications. First, we indeed succeed in proving a lower bound of $\Omega(n^2 \log n)$ (Theorem 4), which matches the classic upper bound of [Laumanns *et al.*, 2002; Giel, 2003]. This is the first tight lower bound for a benchmark problem in which reaching the Pareto front is non-trivial (as opposed to, e.g., the OneMinMax benchmark, where all solutions are on the Pareto front). Second, we also gain a deeper understanding on how difficult it is to reach the Pareto front. Whereas previously the time to find the first solution on the Pareto front was estimated by $O(n^2 \log n)$, we prove that $O(n^2)$ iterations suffice with high probability to reach the Pareto front and to compute any linear fraction of it (Corollary 8).

Our results are made possible by a number of new arguments. The most interesting one is that we add dummy individuals to the population to reach a population size equal to the maximum possible size. If such a dummy individual is chosen as parent, this iteration has no effect (but is counted as iteration). This argument helps overcome the changing population size of the original GSEMO. What is interesting is that this argument, which slows down the original algorithm, can be used to prove lower bounds on the runtime. The reason is that we exploit this argument not to estimate times directly (which is not possibly due to the unclear deceleration from the dummy individuals) but only to understand the shape of the population in the objective space. We are optimistic that this argument, and our other new proof ideas, will be useful in future runtime analyses of MOEAs as well. As a first support for this claim, we show that our methods also give a tight lower bound for the runtime of the GSEMO on the OJZJ_k benchmark for all k (where previous works could not analyze the cases $k = 2$ and $k = 3$) and that all our results extend to the SEMO algorithm.

2 Previous Works

In the interest of space, we concentrate on the previous works most relevant for ours. For a general introduction to MOEAs and their success in applications, we refer to [Coello *et al.*, 2007; Zhou *et al.*, 2011].

We refer to [Neumann and Witt, 2010; Auger and Doerr, 2011; Jansen, 2013; Zhou *et al.*, 2019; Doerr and Neumann, 2020] for introductions to mathematical runtime analyses of randomized search heuristics. We note here that the typical approach in this area is to analyze, with mathematical means, how a specific heuristic solves a particular, often artificial, problem, and to derive from this analysis a deeper understanding of the working principles of the algorithm. Such works have successfully detected strengths or weaknesses of algorithms (e.g., the NSGA-II has intrinsic difficulties with three or more objective [Zheng and Doerr, 2024]), have proposed suitable settings for parameters (e.g., the cutoff time of automated algorithm configurators [Hall *et al.*, 2022]), or have led to the design of novel algorithms (e.g., a variant of the SMS-EMOA with stochastic selection of the next parent population [Bian *et al.*, 2023]).

Already the first mathematical runtime analysis of a MOEA proved an $O(n^2 \log n)$ runtime guarantee for the *simple evolutionary multi-objective optimizer (SEMO)*, a predecessor of the GSEMO, on the COCZ benchmark [Laumanns *et al.*, 2002], see [Laumanns *et al.*, 2004] for the journal version. The same bound for the GSEMO followed a year later [Giel, 2003]. As we will see in this work, both bounds are tight. When looking at the proofs, both results estimate both the time to find the first Pareto optimum and the subsequent time to compute the full Pareto front by $O(n^2 \log n)$, whereas we shall show that the Pareto front is reached, and in fact any constant fraction of it is computed, in time $O(n^2)$. Since then, many more upper bounds on runtimes of the (G)SEMO were shown, and later also for more complex algorithms like the NSGA-II, only very few lower bounds exist, and these only apply to very specific situations.

The first lower bound, matching their own upper bound, is that the SEMO optimizes the LOTZ benchmark in time $\Omega(n^3)$ [Laumanns *et al.*, 2002]. While clearly non-trivial, this result heavily exploits that the SEMO with its one-bit mutation operator cannot generate incomparable solutions until a solution on the Pareto front is found, and from that point on, the population always forms a contiguous interval of the Pareto front. With these restricted population dynamics, proving lower bounds was possible already in the first runtime analysis paper on MOEAs. For the GSEMO, the population dynamics are more complex. In particular, at any time, solutions not comparable with the parent can be generated. Consequently, despite attempts in [Doerr *et al.*, 2013], no interesting lower bounds exist for the GSEMO on LOTZ.

The first tight lower bound for the GSEMO on a classic benchmark was given by [Doerr and Zheng, 2021], who showed that the GSEMO optimizes the OJZJ_k benchmark in time $\frac{3}{2}en^{k+1} \pm o(n^{k+1})$ when the gap parameter satisfies $4 \leq k = o(n)$. That such a tight bound is possible builds on particular properties of this benchmark. The Pareto front of the OJZJ_k benchmark consists of an easy-to-explore inner part, from which two solutions are separated by difficult-to-cross gap of size k. When assuming $k \geq 4$ as in this result, it is easy to argue that the inner part is computed before the gaps are traversed, and hence the traversal of the gaps is slowed down by the then linear-size population. This argument breaks down for smaller values of k, and this is why no tight lower bounds existed in this case prior to this current work. The only other tight lower bound for the GSEMO on a classic benchmark we are aware of is the $\Omega(n^2 \log n)$ bound for the OneMinMax benchmark [Bossek and Sudholt, 2024]. This benchmark has the particularity that all solutions are Pareto optimal, hence the optimization process lacks the phase of advancing towards the Pareto front, which was the most demanding one in our work. Recently, lower bounds where proven for the runtime of the NSGA-II [Doerr and Qu, 2023], but again, these only regard the OneMinMax and OJZJ_k benchmarks; also, clearly, the population dynamics of the NSGA-II with its fixed population size are very different from the GSEMO. In summary, it is safe to say that there are very few interesting lower bounds for the GSEMO, and that

this is caused by the difficulty of understanding the population dynamics of this algorithm.

3 Preliminaries

We now provide some general notation and definitions for multi-objective optimization, define the algorithms (Section 3.1) and benchmark functions (Section 3.2) that used in this study, and state the mathematical tools needed in our analysis (Section 3.3).

Let $\mathbb{Z}$ denote the integers, $\mathbb{N} \coloneqq \mathbb{Z}_{\geq 0}$ the natural numbers (including 0), and $\mathbb{R}$ the reals. For all $a, b \in \mathbb{R}$, let $[a..b] \coloneqq [a, b] \cap \mathbb{Z}$ and $[a] \coloneqq [1..a]$.

We study pseudo-Boolean bi-objective maximization, that is, the maximization of *objective functions* $f\colon \{0,1\}^n \to \mathbb{R}^2$ of *problem size* $n \in \mathbb{N}_{\geq 2}$. We call each $x \in \{0,1\}^n$ an *individual*, and $f(x)$ the *objective value of* x. For all $i \in [n]$, we denote the value of x at position i by x_i.

We compare objective values via the weak and strong *dominance* relationships, which are (strict) partial orders. For all objective values $u, v \in \mathbb{R}^2$, we say that u *weakly dominates* v (written as $u \succeq v$) if and only if $u_1 \geq v_1$ and $u_2 \geq v_2$. If in addition $u \neq v$, then we say that u *strictly dominates* v (written as $u \succ v$). We say that u and v are *incomparable* if neither weakly dominates the other. We extend this terminology to individuals, where it then refers to the individuals' objective values.

Given an objective function f, we call the set of maximal objective values (with respect to dominance) the *Pareto front* of f, that is, the set $\{f(x) \mid x \in \{0,1\}^n \wedge \nexists y \in \{0,1\}^n\colon f(y) \succ f(x)\}$.

3.1 The Algorithms SEMO and GSEMO

We study both the *simple evolutionary multi-objective optimizer* [Laumanns *et al.*, 2002] (SEMO) and the *global SEMO* [Giel, 2003] (GSEMO), which only differ in how they create new solutions (Algorithm 1).

The (G)SEMO maintains a *population* of individuals, which will contain a maximum subset of non-dominated solutions among all solutions seen so far. This population is initialized with a single individual drawn uniformly at random from the search space. In each iteration, one individual is selected uniformly at random (the *parent*) and used to create a new individual (the *offspring*) via *mutation*, that is, a small random perturbation of the parent. Afterward, the algorithm removes all individuals from the population that are weakly dominated by the offspring, and the offspring is added to the population if it is not strictly dominated by a member of the population. This main loop is repeated until a user-defined termination criterion is satisfied.

The difference between the SEMO and the GSEMO is how they create the offspring y from the parent $x \in \{0,1\}^n$. The SEMO uses *1-bit mutation*, which chooses a position $i \in [n]$ uniformly at random and copies x except for position i, which is flipped to the other value. That is, for all $j \in [n] \setminus \{i\}$, we have $y_j = x_j$, for the i-th position we have $y_i = 1 - x_i$. The GSEMO uses *standard bit mutation*, which decides independently for each position whether to flip the bit (with probability $\frac{1}{n}$) or not. That is, for all $i \in [n]$ independently, we have $\Pr[y_i = x_i] = 1 - \frac{1}{n}$ and $\Pr[y_i = 1 - x_i] = \frac{1}{n}$.

Algorithm 1: The (G)SEMO algorithm [Laumanns *et al.*, 2002; Giel, 2003] for maximization of a given bi-objective function $f\colon \{0,1\}^n \to \mathbb{R}^2$. The SEMO uses 1-bit mutation, the GSEMO standard bit mutation (see also Section 3.1).

```
x^(0) ← an individual from {0,1}^n chosen uniformly at random;
P^(0) = {x^(0)};
t ← 0;
while termination criterion not met do
    choose x^(t) from P^(t) uniformly at random;
    y^(t) ← mutation(x^(t));
    Q^(t) ← P^(t) \ {z ∈ P^(t) : f(y^(t)) ⪰ f(z)};
    if ∄z ∈ Q^(t) : f(z) ≻ f(y^(t)) then
        P^(t+1) ← Q^(t) ∪ {y^(t)};
    else P^(t+1) ← Q^(t);
    t ← t + 1;
```

Runtime. As common in the runtime analysis of MOEAs, we define the *runtime* of the (G)SEMO maximizing f as the (random) number of evaluations of f until the objective values of the population contain the Pareto front of f for the first time (we say that the population *covers* the Pareto front). To this end, we assume that the objective value of an individual is evaluated once, namely when it is created. For our definition of runtime to make sense, we assume that the algorithm is never stopped. Since the (G)SEMO creates exactly one individual in each iteration and creates a single individual initially, the runtime is one plus the number of iterations until the population covers the Pareto front of f for the first time.

3.2 The COCZ Benchmark

The function CountingOnesCountingZeros (COCZ) [Laumanns *et al.*, 2002; Laumanns *et al.*, 2004] is defined for even problem sizes $n \in \mathbb{N}_{\geq 2}$. For all $x \in \{0,1\}^n$, we have

$$\mathrm{COCZ}(x) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{[n/2]} x_i + \sum_{i=n/2+1}^{n} (1 - x_i) \right). \quad (1)$$

This popular benchmark models common goals (maximizing the number of ones in the first half of the bit-string) and conflicting goals (maximizing the number of ones resp. zeros in the second half). Formally, let $g_1, g_2\colon \{0,1\}^n \to \mathbb{R}$ denote the number of ones in the first and in the second half of the bit string, respectively. Then, for all $x \in \{0,1\}^n$,

$$\mathrm{COCZ}(x) = \big(g_1(x) + g_2(x), g_1(x) + n/2 - g_2(x)\big).$$

With this notation, it is immediate that the objective space of the COCZ problem is $\mathrm{COCZ}(\{0,1\}^n) = \{(i + j, i + n/2 - j) \mid i, j \in [0..n/2]\}$. Only the objective values with $g_1(x) = n/2$ are Pareto optimal, that is, the Pareto front is $\{(n/2 + j, n - j) \mid j \in [0..n/2]\}$ and has size $n/2 + 1$.

Most objective values, namely all with $g_1(x) \in [0..n/2-1]$, and hence most individuals, are not Pareto-optimal. This is a notable difference to benchmarks such as OneMinMax and OJZJ, where all or most individuals are Pareto-optimal and, in particular, random individuals with high probability lie on the Pareto front.

For COCZ, we finally note that for all $i \in [0..n/2]$, individuals with exactly i ones in their first half are either incomparable or have the same objective value. We use this property in our proofs in Section 4.

3.3 Mathematical Tools

In our analysis, we are mostly concerned with bounding the tails of stopping times. To this end, we decompose a stopping time into smaller parts, each of which denotes a certain phase of the entire process. Theorem 1 provides us with strong guarantees when understanding the separate phases well.

Theorem 1 ([Witt, 2014]). *Let $k \in \mathbb{N}_{\geq 1}$, and let $\{D_i\}_{i\in[k]}$ be independent geometric random variables with respective positive success probabilities $(p_i)_{i\in[k]}$. Let $T^\star \coloneqq \sum_{i\in[k]} D_i$, $s \coloneqq \sum_{i\in[k]} \frac{1}{p_i^2}$, and $p_{\min} \coloneqq \min\{p_i \mid i \in [k]\}$. Then for all $\lambda \in \mathbb{R}_{\geq 0}$, we have*

$$\Pr\big[T^\star \geq \mathbb{E}[T^\star] + \lambda\big] \leq \exp\big(-\tfrac{1}{4}\min\big\{\tfrac{\lambda^2}{s}, \lambda p_{\min}\big\}\big) \text{ and}$$
$$\Pr\big[T^\star \leq \mathbb{E}[T^\star] - \lambda\big] \leq \exp\big(-\tfrac{\lambda^2}{2s}\big).$$

In order to conveniently estimate the sums appearing in applications of Theorem 1, we use the following well-known estimates.

Theorem 2 ([Cormen *et al.*, 2001, Inequality (A.12)]). *Let $g\colon \mathbb{R} \to \mathbb{R}$ be a monotonically non-increasing function, and let $\alpha, \beta \in \mathbb{R}$ with $\alpha \leq \beta$. Then*

$$\int_\alpha^{\beta+1} g(x)\mathrm{d}x \leq \sum_{x=\alpha}^{\beta} g(x) \leq \int_{\alpha-1}^{\beta} g(x)\mathrm{d}x.$$

Last, the following classic Chernoff bound is used to estimate the objective values of initial solutions.

Theorem 3 ([Chernoff, 1952]). *Let $k \in \mathbb{N}_{\geq 1}$, and let $\{X_i\}_{i\in[k]}$ be independent random variables taking values in $[0,1]$. Let $X^\star = \sum_{i\in[k]} X_i$ and $\delta \in [0,1]$. Then*

$$\Pr\big[X^\star \leq (1-\delta)\mathbb{E}[X^\star]\big] \leq \exp\left(-\frac{\delta^2\mathbb{E}[X]}{2}\right).$$

4 Runtime Analysis on COCZ

Our main result is Theorem 4 below, which proves that the (G)SEMO (Algorithm 1) optimizes the COCZ benchmark (equation (1)) with high probability and thus also in expectation in $\Omega(n^2 \log n)$ objective-function evaluations. This matches the $O(n^2 \log n)$ upper bound by [Laumanns *et al.*, 2004], resulting overall in a tight runtime bound of $\Theta(n^2 \log n)$ expected objective-function evaluations.

Theorem 4. *With probability $1 - \Theta(n^{-1})$, the (G)SEMO maximizes* COCZ *in at least $\Omega(n^2 \log n)$ objective-function evaluations.*

Another interesting result of our analysis detailed in the following is that the (G)SEMO achieves a linear population size on the Pareto front of COCZ with high probability after only $O(n^2)$ iterations (Corollary 8). Previously, this time was estimated pessimistically only as $O(n^2 \log n)$.

In order to prove Theorem 4, we need to closely follow the population size of the (G)SEMO during the run. Although the (G)SEMO only creates a single offspring each iteration (and thus only evaluates the objective function a single time), the population size and its composition affect the algorithm's runtime crucially. If the population size is large, progress is only made quickly if the probability is high to select a parent that can be likely turned into a useful offspring. This probability, in turn, relies on where the current population is. For COCZ, assume that the entire population is already on the Pareto front, that is, for each individual x in the population, we have $g_1(x) = n/2$. If the g_2-values of the population consist of a contiguous interval, that is, there is an $i \in [n/2]$ such that for each $j \in [i, n/2-i]$, there is an x in the population such that $g_2(x) = j$, then new solutions are only created likely if individuals with a g_2-value close to the interval borders are chosen.[1] The situation is different if we assume that the individuals have more spread-out g_2-values, that is, if there are some g_2-values in the interval $[i, n/2-i]$ that are not covered by the current population. Then, each individual that is close to the border of some g_2 interval can be useful for finding novel objective values.

A central question to proving our main result (Theorem 4) is which composition the population of the (G)SEMO has once it reaches the Pareto front (and a short time thereafter). In order to answer this question satisfactorily, we view the progress of the algorithm covering the Pareto front of COCZ in two dimensions, namely, with respect to the maximum g_1-value in the population and with respect to the extremal g_2-values in the population. The g_1 quantity translates to how close the population is to reaching the Pareto front, as each individual with a g_1-value of $n/2$ is Pareto-optimal. The g_2 quantity translates to how close the population is to reaching the values in the second objective that are hardest to achieve, that is, $n/2$ (having only zeros in the second half) and n (having only ones in the second half).

In our analysis, we optimistically assume, roughly, that once an individual reaches the Pareto front, all other individuals are also placed there immediately by setting their number of ones in the first half to the maximum value of $n/2$.[2] Thus, tracking the extremal g_2-values tells us from this moment how close the algorithm is to covering the entire Pareto front. In a nutshell, we show that once the algorithm reaches the Pareto front, the extremal g_2-values are still at least order $\sqrt{n}$ away from the borders of the g_2 interval (Lemma 9). From there on, based on a coupon collector argument, the algorithm still requires order $n \log n$ iterations *with useful parents* in order to cover the entire Pareto front. Since we also

[1] For the SEMO, even only the two extreme individuals with a g_2-value of i or $n/2-i$ can create a novel objective values.

[2] Actually, we place all individuals on the Pareto front after a time that is a bit longer than it takes the algorithm to reach the Pareto front, but the main idea remains the same.

prove that the (G)SEMO has a population size of at least $\frac{n}{4}$ once it reaches the Pareto front (Corollary 8), the probability to choose a useful parent is in the order $\frac{1}{n}$. Thus, it still takes $\Omega(n^2 \log n)$ iterations until the algorithm covers the entire front (Lemma 10). Reaching the Pareto front is done faster than that (Lemma 7), and thus Theorem 4 follows.

A modified (G)SEMO algorithm. A main challenge in our proof strategy is to track the exact population size of the algorithm while there are individuals not on the Pareto front. This is due to such individuals being potentially dominated by better solutions and then removed. Once the entire population is on the Pareto front, this problem vanishes, as solutions either have the same objective value or are incomparable. In order to estimate the population size more easily until the Pareto front is reached, we make the following important observation: We only aim to show that the extremal g_2-values are sufficiently far from $n/2$ and n *before* the (G)SEMO reaches the Pareto front. This is a *relative* statement, essentially comparing the progress made with the maximum g_1-value in the population to the progress made with the extremal g_2-values. Thus, we can arbitrarily modify the (G)SEMO as long as we make sure that this relative order is not harmed. We call the resulting algorithm the *modified (G)SEMO*.

The modified (G)SEMO is identical to the (G)SEMO (Algorithm 1) except for line 5, which is replaced by the following procedure, using the notation of the pseudocode. Choose a value $i \in [0..\frac{n}{2}]$ uniformly at random. Check if $\{z \in P^{(t)} \mid g_2(z) = i\}$ is empty. If it is, continue with the next iteration. Otherwise, note that the set contains exactly one individual $x^{(t)}$, as all individuals with equal g_2-value are comparable and $P^{(t)}$ thus contains at most one such individual. Continue with line 6 exactly seen in Algorithm 1, using $x^{(t)}$. Note that the resulting modified (G)SEMO resembles a version of the original (G)SEMO that may add some pointless iterations not modifying the algorithm's state. Thus, in particular, each upper bound on the runtime of the modified (G)SEMO is also an upper bound on the runtime of the original (G)SEMO.

A key observation is that if we consider a run of the modified (G)SEMO and remove all iterations in which an index i with no corresponding individuals is chosen, the algorithm is identical of the original (G)SEMO. Thus, any statements about the states of either algorithm based on stopping times defined only on algorithm states are identical. This allows us to translate results from the modified (G)SEMO to the original one, and it addresses the challenge above of estimating the population size of the original (G)SEMO very closely. Once the modified (G)SEMO reaches the Pareto front and has a linear population size, we switch back to the original (G)SEMO in order to derive a runtime bound for this exact algorithm.

As outlined above, we compare the time it takes the modified (G)SEMO to reach the Pareto front (measured via the maximum g_1-value in the population) and the time it takes to reach extremal g_2-values in the order of $\sqrt{n}$. More specifically, we show that the modified (G)SEMO reaches the Pareto front with high probability in $O(n^2)$ iterations (Lemma 6), whereas it takes $\Omega(n^2 \log n)$ iterations until the g_2-values progressed sufficiently far (Lemma 9). In addition, we show that once the modified (G)SEMO reaches the Pareto front, it reaches a population size linear in n within the same order of time (Lemma 7). Combining these statements, we get with high probability that the modified (G)SEMO (and thus also the original (G)SEMO) has a linear population size before the extremal g_2-values are close to covering the entire interval.

We recall that all upper bounds on the iterations for the modified (G)SEMO algorithm also hold for the original (G)SEMO algorithm.

Progress of the modified (G)SEMO on the g_1-values. We begin by showing that the modified (G)SEMO quickly reaches the Pareto front of COCZ and expands its population to a linear size (where we recall that for the (G)SEMO, different Pareto optima in the population necessarily have different objective values). To this end, we determine the probability to cover a fitness vector of the current best cooperative level if a linear fraction of these vectors is still uncovered.[3]

Lemma 5. *Let $0 < \delta < 1$. Consider one iteration of the modified (G)SEMO maximizing $f \coloneqq$ COCZ, and denote by Z_t the number of individuals with a maximum g_1-value ℓ in $P^{(t)}$. Suppose that $1 \leq Z_t < \delta n/2$. Then the probability to increase Z_t by one is at least $\frac{1}{n/2+1} \cdot \frac{1-\delta}{4e}$.*

With Lemma 5, we bound the expected time to find a Pareto optimal individual in the modified (G)SEMO from above.

Lemma 6. *Consider the modified (G)SEMO maximizing $f \coloneqq$ COCZ. With probability $1 - \exp(-\Omega(\sqrt{n}))$, after at most $O(n^2)$ iterations, for every initialization of $x^{(0)}$ the population of the modified (G)SEMO reaches the Pareto front, i.e. $P^{(t)}$ contains a Pareto optimal individual x.*

Once the modified (G)SEMO reaches the Pareto front, we show that it achieves a population size linear in the problem size in the same amount of time.

Lemma 7. *Consider the modified (G)SEMO maximizing* COCZ *and suppose there is an individual on the Pareto front. Then with probability $1 - \exp(-\Omega(n))$, after at most $O(n^2)$ iterations the population of the modified (G)SEMO contains at least $\frac{n}{4}$ individuals on the Pareto front.*

Combining Lemmas 6 and 7, we obtain that the modified (G)SEMO has a linear population size and is on the Pareto front in at most $O(n^2)$ iterations, with high probability.

Corollary 8. *Consider the modified (G)SEMO maximizing* COCZ*. Then with probability $1 - \exp(-\Omega(\sqrt{n}))$, after at most $O(n^2)$ iterations the population of the modified (G)SEMO contains at least $\frac{n}{4}$ individuals on the Pareto front.*

Progress of the modified (G)SEMO on the g_2-values. We show that the modified (G)SEMO takes some time in order to find solutions that are close to the extremal solutions of the Pareto front. We call this value the *distance to the Pareto borders*. Formally, for all $z \in \{0,1\}^n$, let the *distance of z to the Pareto borders* be $d_{\mathrm{PF}}(z) \coloneqq \min\{g_2(z), \frac{n}{2} - g_2(z)\}$. Using the notation of Algorithm 1, for all $t \in \mathbb{N}$, we define the *distance of the algorithm to the Pareto borders in iteration t* as $d_{\mathrm{PF}}(P^{(t)}) \coloneqq \min_{z \in P^{(t)}} d_{\mathrm{PF}}(z)$.

[3] For reasons of space, most proofs had to be omitted in this extended abstract. They can be found in the long version [Doerr *et al.*, 2025b].

Lemma 9. *Consider the modified (G)SEMO maximizing* COCZ*. Let $c \in \mathbb{R}_{>0}$ be a sufficiently small constant. With probability at least $1 - \Theta(n^{-2/5})$, for all iterations $t \in [0..cn^2 \ln n]$, the distance of the algorithm to the Pareto borders in iteration t is at least $\sqrt{n}$.*

How the original (G)SEMO computes the full Pareto front. We show that if the original (G)SEMO is started in a state that the modified (G)SEMO reaches with high probability in $O(n^2)$ iterations, the original (G)SEMO still requires at least order $n^2 \ln n$ iterations in order to cover the Pareto front. This statement relies on the linear lower bound on the population size from Corollary 8 as well as on the distance of at least $\sqrt{n}$ to the Pareto borders from Lemma 9.

Lemma 10. *Consider the (G)SEMO maximizing* COCZ*, starting with a population size of $\Theta(n)$ on the Pareto front and a distance to the Pareto borders of at least $\sqrt{n}$. Then with probability $1 - \Theta(n^{-1})$, the algorithm covers the Pareto front after $\Omega(n^2 \log n)$ objective-function evaluations.*

Last, we prove our main result (Theorem 4) by showing that the starting state assumed in Lemma 10 is reached with high probability, as sketched before the lemma.

Proof of Theorem 4. We only start counting function evaluations once the (G)SEMO has at least $\Theta(n)$ individuals on the Pareto front. Let T denote the first iteration in which this is the case. We proceed to argue why it has with probability $1 - \Theta(n^{-1})$ a distance of at least $\sqrt{n}$ to the Pareto borders in iteration T. By applying Lemma 10, the statement follows then and the proof is concluded.

In order to show that the (G)SEMO is in the desired state in iteration T, we consider the modified (G)SEMO instead. Recall that the original (G)SEMO changes states if and only if the modified (G)SEMO does so, albeit in potentially different iterations, and they transition into identical states. Let S denote the first iteration in which the modified (G)SEMO has at least $\Theta(n)$ individuals on the Pareto front. By Lemma 6, with probability $1 - \exp\big(-\Omega(\sqrt{n})\big)$, we have that $S = O(n^2)$. Moreover, by Lemma 9, we have with probability $1 - \Theta(n^{-2/5})$ that the distance of the modified (G)SEMO to the Pareto borders is at least $\sqrt{n}$. Hence, with probability $1 - \Theta(n^{-2/5})$, the modified (G)SEMO is in the desired state in iteration S.

Since S and T refer to the same state of the respective algorithm, it follows that the original (G)SEMO is also in the desired state in iteration T with probability $1 - \Theta(n^{-1})$, concluding the proof. □

5 Runtime Analysis on OMM and OJZJ

We show that our insights from Section 4 about the population dynamics on COCZ also translate to the popular bi-objective benchmarks OMM [Giel and Lehre, 2010] and OJZJ [Doerr and Zheng, 2021].

The OMM benchmark aims at maximizing and minimizing the number of ones in a bit string, resulting in *all* individuals being Pareto-optimal. This function resembles COCZ without the cooperative part. Formally, for all $x \in \{0,1\}^n$,

$$\text{OMM}(x) = \Big(\textstyle\sum_{i\in[n]} x_i, \sum_{i\in[n]}(1-x_i)\Big). \tag{2}$$

The Pareto front of OMM is $\{(i, n-i) \mid i \in [0..n]\}$. In particular, each individual is Pareto-optimal, different from COCZ.

The OJZJ benchmark requires a *gap size* $k \in [2..\lfloor n/2 \rfloor]$ and is structurally identical to OMM for all individuals whose number of ones is at least k at most $n-k$. Those individuals are all Pareto-optimal. In addition, the all-ones and the all-zeros bit string are Pareto-optimal as well. All other individuals are strictly worse. This usually requires algorithms to flip at least k bits in order to find the extremal Pareto optima. This is formally defined as for all $x \in \{0,1\}^n$, letting $|x|_1$ and $|x|_0$ denote respectively the number of ones and the number of zeros in x, let $\text{OJZJ}_k(x) = \big(f_1(x), f_2(x)\big)$ with

$$f_1(x) = \begin{cases} k + |x|_1, & \text{if } |x|_1 \leq n-k \text{ or } x = 1^n, \\ n - |x|_1, & \text{else, and} \end{cases} \tag{3}$$

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

The first objective is the single-objective JUMP_k benchmark, which features a local optimum at $n-k$. The second objective is structurally identical to the first but with the roles of ones and zeros reversed. [Doerr and Zheng, 2021] showed that the Pareto front F^* of OJZJ_k is $\{(a, 2k+n-a) \mid a \in [2k \dots n] \cup \{k, n-k\}\}$. Note that each individual x with $f(x) \in F^*$ strictly dominates each individual y with $f(y) \notin F^*$ since, for all $j \in [2]$, we have $f_j(x) \geq k$ but $f_j(y) \leq n - (n - k + 1) = k - 1$. For OMM, tight bounds are already known (see Section 5.1). Hence, our results just provide a different angle of proving them. For OJZJ, tight bounds were known for all $k \in \mathbb{N}_{\geq 4}$ (see Section 5.2), as a pessimistic bound of n for the population size is sufficient. Our result shows that this bound also holds for the cases $k \in \{2, 3\}$, where our insights into the population dynamics are important.

For both benchmarks, we follow a similar strategy as in Section 4. Especially, we rely again on the modified (G)SEMO. This modification needs to be slightly adjusted as follows, using the notation from its original definition: We choose a value $i \in [0..n]$ uniformly at random (instead of from $[0..\frac{n}{2}]$) and check whether the set $\{z \in P^{(t)} \mid g_1(z) + g_2(z) = i\}$ is empty. That is, instead of focusing only on the number of ones in the first half, we now consider the number of ones in the entire bit string. The rest remains identical.

Progress of the modified (G)SEMO. Lemma 11 below essentially translates Corollary 8 to OMM and OJZJ and shows that the modified (G)SEMO reaches a population size of $\frac{n}{2}$ in $O(n^2)$ iterations.

Lemma 11. *Consider the modified (G)SEMO maximizing* OMM *or* OJZJ_k *for $1 < k \leq n/4$. With probability $1 - \exp(-\Omega(\sqrt{n}))$, after at most $O(n^2)$ iterations, for every initialization of $x^{(0)}$ in case of* OMM *or for an initialization on the Pareto front distinct from 0^n and 1^n in case of* OJZJ_k*, the population of the modified (G)SEMO contains at least $n/2$ individuals.*

Lemma 12 below translates Lemma 9 to this setting and also uses its terminology. It shows that the extremal solutions

in the population are still at least $\sqrt{n}$ away from the borders. We need to re-define though what these terms exactly mean in the setting of OMM and OJZJ.

The extremal solutions of the Pareto front are 1^n and 0^n instead of 1^n and $1^{n/2}0^{n/2}$ for COCZ. Furthermore, for all $z \in \{0,1\}^n$, let the *distance of z to the Pareto borders* be $d_{\mathrm{PF}}(z) \coloneqq \min\{|z|_1, n-|z|_1\}$ which is also slightly different to the case of COCZ above. However, the *distance of the algorithm to the Pareto borders in iteration t* is defined as $d_{\mathrm{PF}}(P^{(t)}) \coloneqq \min_{z\in P^{(t)}} d_{\mathrm{PF}}(z)$ for all $t \in \mathbb{N}$ in the same way as in Section 4.

Lemma 12. *Consider the modified (G)SEMO maximizing* OMM *or* OJZJ_k *for $1 < k \leq n/4$. Let $c \in \mathbb{R}_{>0}$ be a sufficiently small constant. With probability at least $1-\Theta(n^{-2/5})$, for all iterations $t \in [0..cn^2 \ln n]$, the distance of the algorithm to the Pareto borders in iteration t is at least $\sqrt{n}$ for* OMM *and at least $\max\{\sqrt{n}, k\}$ for* OJZJ_k.

5.1 OMM

For OMM we prove a bound of $\Omega(n^2 \log n)$ objective-function evaluations, with high probability (Theorem 13). This matches the $O(n^2 \log n)$ bound by [Giel and Lehre, 2010]. The tight $\Theta(n^2 \log n)$ runtime for the GSEMO was already proven by [Bossek and Sudholt, 2024] as a special case of the single-objective problem of quality diversity on the OneMax benchmark. The bound $\Theta(n^2 \log n)$ for the SEMO was shown by [Covantes Osuna *et al.*, 2020].

Theorem 13. *With probability $1 - \Theta(n^{-1})$, the (G)SEMO maximizes* OMM *in $\Omega(n^2 \log n)$ objective-function evaluations.*

The proof of Theorem 13 is very similar to that of Theorem 4. We use Lemmas 11 and 12 for OMM from above.

Lemma 14. *Consider the (G)SEMO maximizing* OMM*, starting with a population size of $\Theta(n)$ on the Pareto front and the distance to the Pareto borders is at least $\sqrt{n}$. Then with probability $1 - \Theta(n^{-1})$, the algorithm covers the Pareto front after $\Omega(n^2 \log n)$ objective-function evaluations.*

By combining Lemmas 11, 12, and 14, we obtain the proof for Theorem 13 in a similar way as the proof of Theorem 4.

5.2 OJZJ

For OJZJ, we only consider the GSEMO, as the SEMO does not cover the Pareto front with high probability ([Doerr and Zheng, 2021]), due to the deceptive nature of the benchmark and the 1-bit mutation being incapable of creating the extremal Pareto optima from non-extremal Pareto optima. For gap size $k \in [2..\frac{n}{4}]$, we prove a bound of $\Omega(n^{k+1})$ objective-function evaluations, with high probability (Theorem 15). This matches the bound $O(n^{k+1})$ for all of these values of k by [Doerr and Zheng, 2021]. Moreover, for $k \in [4..\frac{n}{2}-1]$, [Doerr and Zheng, 2021] proved already a matching bound of $\Omega((n-2k)n^k)$. However, for $k \in \{2,3\}$, our results are new.

Theorem 15. *In expectation, the GSEMO maximizes* OJZJ_k *for $k \in [2..\frac{n}{4}]$ in $\Omega(n^{k+1})$ objective-function evaluations.*

The proof of Theorem 15 makes use of Lemma 16 below, which shows that it takes the GSEMO a lot of time to find the all-ones and all-zeros bit string.

Lemma 16. *Consider the GSEMO maximizing* OJZJ_k *for $1 < k \leq n/4$, starting with a population size of $\Theta(n)$ on the Pareto front, but neither 0^n nor 1^n are in the population. Then the algorithm covers the Pareto front in expectation after $\Omega(n^{k+1})$ objective-function evaluations.*

Theorem 15 can be proven in a similar way to Theorems 4 and 13 by using Lemmas 11 and 12 for OJZJ_k when $1 < k \leq n/4$ and then following Lemma 16.

6 Conclusion

We studied the population dynamics of the (G)SEMO, that is, the size and shape of its population over time. For the COCZ benchmark, we proved that the algorithm has a population size linear in the size of the Pareto front (Corollary 8) while it is still sufficiently far away from covering the entire Pareto front (Lemma 9). Covering these remaining solutions takes at least $\Omega(n^2 \log n)$ iterations (Theorem 4). Since a matching upper bound exists, this result is tight.

Our proof strategy relies on defining a modified process that allows an easier estimate of the probability to select a useful parent for making progress. This modification affects the absolute number of iterations but not the relative order of state changes. Thus, insights into state behavior for the modification also translate directly back to the original (G)SEMO.

We show that our insights for COCZ also transfer to OMM and OJZJ, where we derive lower bounds that match known upper bounds. Although most of these lower bounds were already known, our proof strategy provides a different angle for deriving them. Moreover, our lower bounds for gap sizes $k \in \{2,3\}$ for OJZJ_k are new. Since we prove a general lower bound for a large range of k, all of which are tight, our method captures the true nature of the (G)SEMO well.

Our analysis shows that the GSEMO is hampered by exploiting easy parts of the search space too quickly. This lets the population grow quickly, slowing down the selection required for exploring harder-to-reach parts of the search space.

For future work, it would be interesting to derive lower bounds for the population size while the (G)SEMO is approaching the Pareto front. In this article, we only derive upper bounds for this time (Corollary 8). Lower bounds would give us a clearer picture how wasteful the algorithm is in terms of function evaluations before reaching the Pareto front.

Another interesting open problem is to derive tight lower bounds of the GSEMO for the LeadingOnesTrailingZeros (LOTZ) benchmark. For this setting, so far, only the bound $O(n^3)$ [Giel, 2003] exists (but a $\Theta(n^3)$ bound for the SEMO). In contrast to the setting in this article, for LOTZ, it is far more likely for the GSEMO to get closer to the Pareto front than to create an incomparable offspring. Only once the Pareto front is reached, these two probabilities are in the same order. Hence, the dynamics seem to be different from here.

Our result can also serve as a stepping stone towards the deeper understanding of the population dynamics of other MOEAs, like the NSGA-II. However, since the NSGA-II has a fixed population size, parts of the analysis need to focus on how duplicate entries in the population are treated, which is a separate topic and thus left for future work.

Acknowledgments

This research benefited from the support of the FMJH Program PGMO.

This paper significantly profited from discussions at the Dagstuhl Seminars 23361 *Multiobjective Optimization on a Budget* and 24271 *Theory of Randomized Optimization Heuristics*.

References

[Alghouass *et al.*, 2025] Yasser Alghouass, Benjamin Doerr, Martin S. Krejca, and Mohammed Lagmah. Proven approximation guarantees in multi-objective optimization: SPEA2 beats NSGA-II. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*. ijcai.org, 2025. Accepted for publication.

[Auger and Doerr, 2011] Anne Auger and Benjamin Doerr, editors. *Theory of Randomized Search Heuristics*. World Scientific Publishing, 2011.

[Bian and Qian, 2022] Chao Bian and 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*, pages 428–441. Springer, 2022.

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

[Bossek and Sudholt, 2024] Jakob Bossek and Dirk Sudholt. Runtime analysis of quality diversity algorithms. *Algorithmica*, 86:3252–3283, 2024.

[Chernoff, 1952] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

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

[Cormen *et al.*, 2001] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2 edition, 2001.

[Covantes Osuna *et al.*, 2020] Edgar Covantes Osuna, Wanru Gao, Frank Neumann, and Dirk Sudholt. Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*, 832:123–142, 2020.

[Crawford, 2021] Victoria G. Crawford. Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In *International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1661–1667. ijcai.org, 2021.

[Dang *et al.*, 2024] Duc-Cuong Dang, Andre Opris, and Dirk Sudholt. Crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. *Artificial Intelligence*, 330:104098, 2024.

[Deng *et al.*, 2025] Renzhong Deng, Weijie Zheng, and Benjamin Doerr. The first theoretical approximation guarantees for the non-dominated sorting genetic algorithm III (NSGA-III). In *International Joint Conference on Artificial Intelligence, IJCAI 2025*. ijcai.org, 2025. Accepted for publication.

[Dinot *et al.*, 2023] Matthieu Dinot, Benjamin Doerr, Ulysse Hennebelle, and Sebastian Will. Runtime analyses of multi-objective evolutionary algorithms in the presence of noise. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5549–5557. ijcai.org, 2023.

[Do *et al.*, 2023] Anh Viet Do, Aneta Neumann, Frank Neumann, and Andrew M. Sutton. Rigorous runtime analysis of MOEA/D for solving multi-objective minimum weight base problems. In *Advances in Neural Information Processing Systems, NeurIPS 2023*, pages 36434–36448, 2023.

[Doerr and Neumann, 2020] Benjamin Doerr and Frank Neumann, editors. *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.

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

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

[Doerr *et al.*, 2013] Benjamin Doerr, Bojana Kodric, and Marco Voigt. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2013*, pages 432–439. IEEE, 2013.

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

[Doerr *et al.*, 2025b] Benjamin Doerr, Martin S. Krejca, and Andre Opris. Tight runtime guarantees from understanding the population dynamics of the GSEMO multi-objective evolutionary algorithm. *CoRR*, abs/2505.01266, 2025.

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

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

[Hall *et al.*, 2022] George T. Hall, Pietro S. Oliveto, and Dirk Sudholt. On the impact of the performance metric on efficient algorithm configuration. *Artificial Intelligence*, 303:103629, 2022.

[Jansen, 2013] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer, 2013.

[Laumanns *et al.*, 2002] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, and Kalyanmoy Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving from Nature, PPSN 2002*, pages 44–53. Springer, 2002.

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

[Li *et al.*, 2025] Mingfeng Li, Weijie Zheng, and Benjamin Doerr. Scalable speed-ups for the SMS-EMOA from a simple aging strategy. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*. ijcai.org, 2025. Accepted for publication.

[Neumann and Witt, 2010] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[Opris *et al.*, 2024] Andre Opris, Duc Cuong Dang, Frank Neumann, and Dirk Sudholt. Runtime analyses of NSGA-III on many-objective problems. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, pages 1596–1604. ACM, 2024.

[Opris, 2025] Andre Opris. A many objective problem where crossover is provably indispensable. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*. ijcai.org, 2025. Accepted for publication.

[Qian *et al.*, 2015] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset selection by pareto optimization. In *Neural Information Processing Systems, NIPS 2015*, pages 1774–1782, 2015.

[Qian *et al.*, 2017] Chao Qian, Jing-Cheng Shi, Yang Yu, Ke Tang, and Zhi-Hua Zhou. Optimizing ratio of monotone set functions. In *International Joint Conference on Artificial Intelligence, IJCAI 2017*, pages 2606–2612. ijcai.org, 2017.

[Qian *et al.*, 2018] Chao Qian, Yibo Zhang, Ke Tang, and Xin Yao. On multiset selection with size constraints. In *Conference on Artificial Intelligence, AAAI 2018*, pages 1395–1402. AAAI Press, 2018.

[Qian *et al.*, 2019] Chao Qian, Yang Yu, Ke Tang, Xin Yao, and Zhi-Hua Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *Artificial Intelligence*, 275:279–294, 2019.

[Qian *et al.*, 2020] Chao Qian, Chao Bian, and Chao Feng. Subset selection by Pareto optimization with recombination. In *Conference on Artificial Intelligence, AAAI 2020*, pages 2408–2415. AAAI Press, 2020.

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

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

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

[Wietheger and Doerr, 2024] Simon Wietheger and Benjamin Doerr. Near-tight runtime guarantees for many-objective evolutionary algorithms. In *Parallel Problem Solving from Nature, PPSN 2024, Part IV*, pages 153–168. Springer, 2024.

[Witt, 2014] Carsten Witt. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters*, 114:38–41, 2014.

[Zheng and Doerr, 2024] Weijie Zheng and Benjamin Doerr. Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. *IEEE Transactions on Evolutionary Computation*, 28:1442–1454, 2024.

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

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

[Zhou *et al.*, 2019] Zhi-Hua Zhou, Yang Yu, and Chao Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.