Runtime Analyses of Multi-Objective Evolutionary Algorithms in the Presence of Noise

Matthieu Dinot, Benjamin Doerr, Ulysse Hennebelle, Sebastian Will

Runtime Analyses of Multi-Objective Evolutionary Algorithms in the Presence of Noise*

Matthieu Dinot[1] , **Benjamin Doerr**[2] , **Ulysse Hennebelle**[1] , **Sebastian Will**[2]

[1]École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

[2]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

firstname.lastname@polytechnique.edu

Abstract

In single-objective optimization, it is well known that evolutionary algorithms also without further adjustments can stand a certain amount of noise in the evaluation of the objective function. In contrast, this question is not at all understood for multi-objective optimization.

In this work, we conduct the first mathematical runtime analysis of a simple multi-objective evolutionary algorithm (MOEA) on a classic benchmark in the presence of noise in the objective function. We prove that when bit-wise prior noise with rate $p \leq \alpha/n$, α a suitable constant, is present, the *simple evolutionary multi-objective optimizer* (SEMO) without any adjustments to cope with noise finds the Pareto front of the OneMinMax benchmark in time $O(n^2 \log n)$, just as in the case without noise. Given that the problem here is to arrive at a population consisting of $n + 1$ individuals witnessing the Pareto front, this is a surprisingly strong robustness to noise (comparably simple evolutionary algorithms cannot optimize the single-objective OneMax problem in polynomial time when $p = \omega(\log(n)/n)$). Our proofs suggest that the strong robustness of the MOEA stems from its implicit diversity mechanism designed to enable it to compute a population covering the whole Pareto front.

Interestingly this result only holds when the objective value of a solution is determined only once and the algorithm from that point on works with this, possibly noisy, objective value. We prove that when all solutions are reevaluated in each iteration, then any noise rate $p = \omega(\log(n)/n^2)$ leads to a super-polynomial runtime. This is very different from single-objective optimization, where it is generally preferred to reevaluate solutions whenever their fitness is important and where examples are known such that not reevaluating solutions can lead to catastrophic performance losses.

*Author-generated version.

1 Introduction

Many real-world optimization problems consist of multiple, often conflicting objectives. For these, a single optimal solution usually does not exist. Consequently, a common solution concept is to compute a set of solutions which cannot be improved in one objective without worsening in another one (Pareto optima) and then let a decision maker select one of these.

Due to their population-based nature, evolutionary algorithms (EAs) are an obvious heuristic approach to such problems, and in fact, such multi-objective evolutionary algorithms (MOEAs) have been successfully applied to many multi-objective problems [Zhou *et al.*, 2011].

Evolutionary algorithms are known to be robust against different types of stochastic disturbances such as noise or dynamic changes of the problem instance [Jin and Branke, 2005]. Surprisingly, as regularly pointed out in the literature [Liefooghe *et al.*, 2007; Gutjahr, 2012; Gutjahr and Pichler, 2016], only very little is known on how MOEAs cope with such stochastic optimization problems. In particular, while it is known that single-objective evolutionary algorithms without any specific adjustments can stand a certain amount of noise in the evaluation of the objective function, we are not aware of any such result in multi-objective optimization.

We approach this question via the methodology of mathematical runtime analysis [Neumann and Witt, 2010; Auger and Doerr, 2011; Jansen, 2013; Doerr and Neumann, 2020]. This field, for more than twenty years, has significantly enlarged our understanding of the working principles of all kinds of randomized search heuristics, including both MOEAs [Brockhoff, 2011] and single-objective evolutionary algorithms solving stochastic optimization problems [Neumann *et al.*, 2020]. Despite this large amount of work, there is not a single runtime analysis discussing how a standard MOEA computes or approximates the Pareto front of a multi-objective problem in the presence of noise (and this is what we shall do in the present work). The only paper that conducts a mathematical runtime analysis of a MOEA in a noisy setting analyzes a combination of the adaptive Pareto sampling frameworks with the simple evolutionary multi-objective optimizer (SEMO) [Gutjahr, 2012], so this algorithm definitely is not anymore a standard MOEA.

To start closing this gap, we conduct a runtime analysis of a simple MOEA, namely the SEMO, on the classic

benchmark problem ONEMINMAX, in the presence of one-bit prior noise. We prove that when the noise rate is at most α/n, where α is a suitable constant, then the population of this MOEA, despite the noise, after an expected number of $O(n^2 \log n)$ iterations witnesses the full Pareto front. This is the same bound on the runtime that is known for the setting without noise [Giel and Lehre, 2010; Covantes Osuna *et al.*, 2020]. We note that in comparable single-objective settings, not much more noise can be tolerated. For example, only for $p = O(\log(\log(n))/n)$ it could be shown in [Dang-Nhu *et al.*, 2018] that the $(1+1)$ EA retains its noise-free $O(n \log n)$ runtime on the ONEMAX benchmark. Already for $p = \omega(\log(n)/n)$, the runtime is super-polynomial. Considering this and taking into account that the multi-objective ONEMINMAX problem is naturally harder (we aim at a population containing exactly one solution for each Pareto optimum), our result indicates that MOEAs cope with noise surprisingly well.

However, our work also shows one important difference to the noisy optimization of single-objective problems. Our result above assumes that each solution is evaluated only once, namely when it is generated. This possibly noisy objective value is stored with the solution unless the solution is discarded at some time. This approach is natural in that it avoids costly reevaluations, but it differs from the standards in single-objective evolutionary computation. Being afraid that a faulty fitness value can negatively influence the future optimization process, almost all works there assume that each time a solution competes with others, its fitness is reevaluated. That noisy fitness values without reevaluating solutions can significantly disturb the optimization process was rigorously shown for an ant-colony optimizer [Doerr *et al.*, 2012a].

Given that in single-objective evolutionary computation it is more common to assume that solutions are evaluated anew when they compete with others, we also analyzed the runtime of the SEMO on ONEMINMAX under this assumption. While this avoids sticking to faulty objective values for a long time, our mathematical runtime analysis shows that this approach can only tolerate much lower levels of noise. We can prove an $O(n^2 \log n)$ runtime only for $p \leq \beta/n^2$, β a suitable constant, and we prove that for $p = \omega(\log(n)/n^2)$ the algorithms needs super-polynomial time to compute the full Pareto front of the ONEMINMAX benchmark. So clearly, the reevaluation strategy recommended in single-objective optimization is less suitable in multi-objective optimization (in addition to the significantly higher computational cost of a single iteration).

Overall, this first runtime analysis work of a standard MOEA in a noisy environment shows that MOEAs without specific adjustments are reasonably robust to noise and that such stochastic processes can be analyzed with mathematical means, but also that insights from the single-objective setting can be fundamentally wrong in multi-objective noisy optimization.

2 Previous Work

Given the success of evolutionary algorithms both in multi-objective and in stochastic optimization, there is a large body of literature on both topics. For the general picture, we refer to the surveys [Zhou *et al.*, 2011] and [Jin and Branke, 2005; Bianchi *et al.*, 2009].

Much less understood is the intersection of both areas, that is, how MOEAs solve stochastic optimization problems, see [Gutjahr, 2011, Section 4]. Interestingly, all works described there adapt the MOEA to deal with the stochasticity of the problem. This is justified for problems that have a strong stochasticity known in advance. However, it is known that evolutionary algorithms often can tolerate a certain amount of stochastic disturbance, in particular, noisy function evaluations without any adjustments (and hence, without that the user has to be aware of this noise). In contrast to single-objective optimization, we are not aware of any such result for MOEAs.

This being a theoretical work, we now describe the mathematical runtime analyses closest to our work. The mathematical runtime analysis of MOEAs was started in [Laumanns *et al.*, 2002a; Giel, 2003; Thierens, 2003]. These and most subsequent works regarded artificial toy algorithms like the simple evolutionary multi-objective optimizer (SEMO) or the global SEMO (GSEMO). The hope is that results proven on such basic algorithms, or at least the general insights drawn from them, extend to more realistic algorithms. That this hope is not unrealistic can be seen, e.g., from the fact that the runtimes shown for the SEMO on the LOTZ and COCZ benchmarks in [Laumanns *et al.*, 2002b] could much later be proven also for the NSGA-II [Zheng *et al.*, 2022; Bian and Qian, 2022], the most prominent MOEA.

For our work, naturally, the results on the ONEMINMAX benchmark are most relevant. In [Giel and Lehre, 2010], it was proven that the SEMO finds the Pareto front of this benchmark in an expected number of $O(n^2 \log n)$ iterations. A matching lower bound was proven in [Covantes Osuna *et al.*, 2020]. The $O(n^2 \log n)$ upper bound also holds for the GSEMO (never formally proven, but easy to see from the proof in [Giel and Lehre, 2010]), the hypervolume-based $(\mu+1)$ SIBEA with suitable population size [Nguyen *et al.*, 2015], and the NSGA-II with suitable population size [Zheng *et al.*, 2022]. For the NSGA-III, a runtime analysis exists only for the 3-objective version of ONEMINMAX [Doerr and Wietheger, 2023], but it is obvious that this result immediately extends to an $O(n^2 \log n)$ bound in the case of two objectives. The main argument in all these analyses (which is not anymore true in the noisy setting) is that Pareto points cannot be lost, that is, once the population of the algorithm contains a solution with a certain Pareto-optimal objective value, it does so forever.

Due to their randomized nature, it is not surprising that EAs can tolerate a certain amount of stochastic disturbance. In the first runtime analysis of an EA for discrete search spaces in the presence of noise [Droste, 2004], Droste investigated how the $(1+1)$ EA optimizes the ONEMAX benchmark in the presence of bit-wise prior noise with noise rate p (see Section 3.2 for a precise definition of this noise model). He showed that the runtime remains polynomial (but most likely higher than the well-known $O(n \log n)$ runtime of the noiseless setting) when $p = O(\log(n)/n)$. When $p = \omega(\log(n)/n)$, a super-polynomial runtime results. These

bounds have been tightened and extended in the future, in particular, an $O(n \log n)$ runtime bound for noise rate $p = O(\log(\log(n))/n)$ with implicit constant small enough was shown in [Dang-Nhu *et al.*, 2018]. The level of noise an algorithm can stand depends strongly on the problem, for example, for the LEADINGONES benchmark the $(1+1)$ EA has a polynomial runtime if and only if the noise rate is at most $p = O(\log(n)/n^2)$ [Sudholt, 2021]. We refer to [Neumann *et al.*, 2020] for more results.

We note that all these works regard the standard version of the EA without any particular adjustments to better cope with noise. It is well-studied that resampling techniques can increase the robustness to noise [Akimoto *et al.*, 2015; Qian *et al.*, 2018; Doerr and Sutton, 2019], however, this requires the algorithm user to be aware of the presence of noise and have an estimate of its strength.

Reevaluating solutions: When running a standard EA in a noisy environment, there is one important implementation choice, namely whether to store the possibly noisy fitness of a solution or to reevaluate the solution whenever it becomes important in the algorithm. The latter, naturally, is more costly due to the higher number of fitness evaluations, but since most EAs generate more solutions per iteration than what they take into the next generation, this often is only a constant-factor performance loss. On the other hand, sticking to the objective value seen in the first evaluation carries the risk that a single unlucky evaluation has a huge negative impact on the remaining run of the algorithm.

The question which variant to prefer has not been discussed extensively in the literature. However, starting with the first runtime analysis of an EA in a noisy environment [Droste, 2004], almost all runtime analyses of EAs in a noisy environment assume that each solution is reevaluated whenever it plays a role in the algorithm, in particular, whenever solution qualities are compared.

The only example we are aware of where this is done differently is the analysis of how an ant-colony optimizer (ACO) solves a stochastic shortest path problem in [Sudholt and Thyssen, 2012]. This work regards a variant of the Max-Min Ant System [Stützle and Hoos, 2000] with best-so-far pheromone update where the best-so-far solution is kept without reevaluation. While this algorithm was provably able to solve some stochastic path problems, it was not able to solve others (except possibly in exponential time). The same algorithm, but with the best-so-far solution being reevaluated whenever it competes with a new solution, was analyzed in [Doerr *et al.*, 2012a] and it was shown to be able to solve many problems efficiently which could not be solved by the other variant.

Noisy evolutionary multi-objective optimization: While we apparently have a good understanding on how EAs cope with noise and how they can be used to solve multi-objective problems, there has been very little research on how EAs solve multi-objective problems in the presence of noise. The only mathematical runtime analysis in this direction is [Gutjahr, 2012]. It analyzes how the *adaptive Pareto sampling* framework together with a variant of the SEMO (called with suitable noiseless instances) allows to solve noisy instances. So like all other non-mathematical works on noisy heuristic multi-objective optimization, e.g., [Teich, 2001; Ding *et al.*, 2006; Liefooghe *et al.*, 2007; Boonma and Suzuki, 2009; Fieldsend and Everson, 2015; Rakshit and Konar, 2015], this work does not analyze how a standard MOEA copes with noise, but proposes a specific way how to solve noisy multi-objective optimization problems.

3 Preliminaries: Multi-objective Optimization of the ONEMINMAX Benchmark in the Presence of Noise

3.1 The ONEMINMAX Benchmark

In multi-objective optimization over a set X using an evaluation function $g : X \to \mathbb{R}^d$, we say that a vector $x \in X$ is a Pareto optimal solution if there is no $y \in X$ with $g(x) \prec g(y)$. We denote as X^* the set of Pareto optimal vectors, and define the Pareto front as $g(X^*)$. Note that we will use the partial order on $\mathbb{R}^d$ where $x \preceq y$ if and only if for all $i \in [1..d]$, we have $x_i \leq y_i$. The induced strict ordering will then be defined as $x \prec y \iff (x \preceq y \wedge x \neq y)$.

The ONEMINMAX benchmark is a bi-objective optimization problem defined over the decision space $X = \{0,1\}^n$. It was introduced in [Giel and Lehre, 2010] as a bi-objective version of the classic ONEMAX benchmark. The objective function is defined as

$$g(x) = (f(x), n - f(x)),$$

where the first objective $f(x) = \text{ONEMAX}(x)$ is the number of ones in x. The goal of the algorithms analyzed in this paper is to find a minimal subset of X which has a direct image by g equal to the Pareto front of X for this benchmark. Since any solution to the ONEMINMAX benchmark is Pareto optimal, the Pareto front of this problem is $X^* = \{(k, n-k) \mid k \in [0..n]\}$.

3.2 One-bit Noise

In this article, we consider that every evaluation of an objective vector is subject to noise. Let $p \in [0,1]$ be the noise rate in the whole article.

We define the noisy vector evaluation function $\tilde{x}$ as the vector x but where a random bit uniformly chosen has been flipped with probability p. Each noisy evaluation is independent from one another. This noise model is known as *one-bit noise*.

For any function $\phi : X \to Y$, we define

$$\tilde{\phi}(x) \coloneqq \left\{ \begin{array}{l} X \to Y \\ x \mapsto \phi(\tilde{x}) \end{array} \right. .$$

It is therefore possible to replace the objective function g by its noisy version $\tilde{g}$ in any evaluation.

3.3 Modified SEMO Algorithms for Optimizing Noisy Objective Functions

When running an EA on a noisy problem, one can work with the first noisy fitness value received for a solution or to reevaluate it each time it is relevant. We present two versions of the same algorithm for these two possibilities. In all our algorithms, populations $\mathcal{P}_t$ or $\mathcal{P}'_t$ are multisets. This is necessary

even for a SEMO algorithm (which stores at most one solution per objective value, and hence not multiple copies of an individual) since the noise fitness may let copies of an individual appear different due to their noise fitness.

SEMO Without Reevaluation

The first algorithm is the original SEMO algorithm, but it evaluates values of an element entering the population only once and save its value as long as this element stays in the population.

In Algorithm 1, we store the noisy evaluations of elements of $\mathcal{P}_t$ in $\mathcal{W}_t$. The function $\mathcal{W}_t$ can be seen as a map that associates each element of $\mathcal{P}_t$ with its noisy value as computed at the time of its addition.

Algorithm 1: SEMO without reevaluation

```
1  x_init is uniformly chosen from {0,1}^n
2  P_0 ← {x_init}
3  W_0(x_init) ← g̃(x_init)
4
5  for t = 0 to ∞ do
6      Choose x_t uniformly from P_t
7      Sample x'_t from x_t by flipping one uniformly
         chosen bit in x_t
8
9      w ← g̃(x'_t)
10
11     if there is no x ∈ P_t such that w ≺ W_t(x) then
12         P_{t+1} ← {x'_t}
13         W_{t+1}(x'_t) ← w
14
15         foreach x ∈ P_t do
16             if W_t(x) ⋠ w then
17                 P_{t+1} ← P_{t+1} ∪ {x}
18                 W_{t+1}(x) ← W_t(x)
19     else
20         P_{t+1} ← P_t
21         W_{t+1} ← W_t
```

Note that we state this algorithm in a general way to deal with any multi-objective optimization. For OneMinMax, this algorithm could be simplified, since the condition in line **11** is always true and the inequality in line **16** becomes an equality.

SEMO With Reevaluation

We now describe a SEMO algorithm which reevaluates each solution in each iteration. Reevaluations have been recommended in noisy settings to avoid working with the same, faulty fitness value for a long time. For the SEMO, this requires some adjustments since now it does not suffice anymore to integrate the new offspring into the existing population. Due to the changing fitness values of individuals, it may also happen that two old solutions appear equal, so one of the two has to be removed.

This algorithm uses the elim function defined below. Given a multiset $E \subset X$ and a possibly random evaluation function $h : X \to Y$ where Y is a partially ordered set, $G \coloneqq \mathrm{elim}(E, h)$ will be a minimal sub-multiset of E that Pareto-dominates E. Pareto-dominance meaning that for every $x \in E$, there exists $y \in G$ with $h(x) \preceq h(y)$, and minimality meaning that for every $x, y \in G$, $h(x)$ and $h(y)$ are not comparable.

Algorithm 2: SEMO with reevaluation

```
1  P_0 ← {x_init}, where x_init is uniformly chosen from
     {0,1}^n
2
3  for t = 0 to ∞ do
4      Choose x_t uniformly from P_t
5      Sample x'_t from x_t by flipping one uniformly
         chosen bit in x_t
6
7      P'_t ← P_t ∪ {x'_t}
8      P_{t+1} ← elim(P'_t, g̃)
```

Algorithm 3: Elimination function elim. Inputs are the initial set E and the evaluation function h. Output is G, a minimal Pareto-dominant sub-multiset of E.

```
1  G ← ∅
2  Temp ← ∅
3
4  foreach x ∈ E selected in a random order do
5      v_x ← h(x)
6
7      if there is no v ∈ Temp such that v_x ≺ v then
8          foreach y in G with v_y ⪯ v_x do
9              G ← G \ {y}
10         G ← G ∪ {x}
11         Temp ← Temp ∪ {v_x}
12 return G
```

For OneMinMax, the elim function can be simplified, with the condition in line **7** being always verified and the inequality in line **8** being an equality.

4 Runtime Analysis of the SEMO Without Reevaluation

In this section, we analyze the expected number of iterations of the main loop needed to reach the Pareto front in the SEMO without reevaluation. Our main technical tool is drift analysis, that is, a collection of mathematical tools that allow to translate estimates for the one-step change $X_{t+1} - X_t$ of a random process (X_t) into estimates for hitting times of this process. See [Lengler, 2020] for a recent survey on this method. Unfortunately, for reasons of space, we cannot present the proofs of our results in full detail. They can be found in the long version [Dinot *et al.*, 2023].

We define the total time to reach the Pareto front as a stopping time

$$T_{\mathrm{total}} \coloneqq \min\{t \mid f(\mathcal{P}_t) = [0..n]\}$$

and show that there is a constant α such that for all noise rates $p \leq \frac{\alpha}{n}$ we have $E[T_{\text{total}}] = O(n^2 \log(n))$.

4.1 Observations on the SEMO Without Reevaluation

The function $\mathcal{W}_t$ is the map from the elements of $\mathcal{P}_t$ to their noisy evaluated value at the moment they entered the population. We call $\mathcal{V}_t$ the first component of W_t, that is $W_t = (\mathcal{V}_t, n - \mathcal{V}_t)$. At time t, two situations can occur.

- The noisy value of the offspring x'_t is not in $\mathcal{V}_t(\mathcal{P}_t)$, so it is added to the population and no element is removed.
- There is an element $x \in \mathcal{P}_t$ such that $\mathcal{V}_t(x) = \mathcal{V}_{t+1}(x'_t)$, so the value is already in the set. In that case, x will be removed from the population and x'_t will be added.

From these observations, we deduce the two following lemmas.

Lemma 1. $\mathcal{V}_t(\mathcal{P}_t) \subset \mathcal{V}_{t+1}(\mathcal{P}_{t+1})$

Proof. An element is removed from the population only if the offspring has the same value. □

Lemma 2. *$\mathcal{V}_t$ is one-to-one.*

Proof. It is true at time $t = 0$ because there is only one vector in $\mathcal{P}_t$. If it is true at time t, then the only possible collision is with the offspring x'_t. In that case the element of the same value will be removed from the population and the offspring will be added. V_{t+1} is still one-to-one. The result is true by induction. □

Another important lemma is the following.

Lemma 3. $\forall t, \forall k \in [0..n], |\{x \in \mathcal{P}_t \mid f(x) = k\}| \leq 3$.

Proof. If $f(x) = k$, then because of one-bit noise $\mathcal{V}_t(x) \in \{k-1, k, k+1\}$. Since $\mathcal{V}_t$ is one-to-one, there are at most 3 elements of value k by f. □

4.2 Time Needed to Find the Extreme Values

We define

$$T_{\text{ex}} = \min\{t \mid \{0, n\} \subseteq \mathcal{V}_t(\mathcal{P}_t)\},$$

the first time when the extreme values are noisily found.

Theorem 4. *There is a constant $\alpha > 0$ such that if $p \leq \frac{\alpha}{n}$, then*

$$E[T_{\text{ex}}] = O(n^2 \log(n)).$$

Proof. We denote as j_t the minimum value of $\mathcal{V}_t(\mathcal{P}_t)$ and we introduce the potential function $\ell_t = j_t - \mathbb{1}_{\exists x \in \mathcal{P}_t, \mathcal{V}_t(x) = f(x) = j_t} + 1$. This variable equals $j_t + 1$, but with a bonus downward if the unique element x with $\mathcal{V}_t(x) = j_t$ is correctly evaluated, that is $f(x) = j_t$. We analyze the drift, that is the expected change of the process ℓ_t.

We consider first the case where $f(x) = j_t$.

- The probability of selecting x as a parent for mutation is $\frac{1}{|\mathcal{P}_t|}$. The probability that its offspring has a lower value is $\frac{j_t}{n}$ (namely, selecting a 1 in x and flipping it to a 0). The probability of correctly evaluating the offspring is $1 - p$. The overall probability of this event is $\frac{j_t(1-p)}{n|\mathcal{P}_t|}$. In that case $\ell_t - \ell_{t+1} = 1$. Therefore, when $f(x) = j_t$, the downward component of the drift is at least $\frac{j_t(1-p)}{n|\mathcal{P}_t|}$.
- For the upward component of the drift, since $j_{t+1} \leq j_t$ as stated in Lemma 1, a negative shift only happens if the offspring x'_t is wrongly evaluated to j_t. We would have $j_{t+1} = j_t$ but $\mathbb{1}_{\exists x \in \mathcal{P}_{t+1}, \mathcal{V}_{t+1}(x) = f(x) = j_t} = 0$ and $\ell_t - \ell_{t+1} = -1$. Since in general $\mathcal{V}_t(x) \in \{f(x) - 1, f(x), f(x)+1\}$, the value of the offspring is $j_t - 1$ or $j_t + 1$. The mutation is also one-bit, so the parent x_t at time t must be of value $j_t - 2$, j_t or $j_t + 2$. The value $j_t - 2$ is not an option, as the noisy value of x_t would be lower than $j_t - 1$ so strictly lower than j_t, so there is at most 6 elements verifying this property according to Lemma 3. The probability of selecting one of those to create the offspring is lower than $\frac{6}{|\mathcal{P}_t|}$. The probability of this offspring to be wrongly evaluated to j_t is lower than p.

The total drift when $f(x) = j_t$ is $\frac{j_t(1-p)}{n|\mathcal{P}_t|} - \frac{6p}{|\mathcal{P}_t|}$.

Now consider the case where $f(x) \neq j_t$. In that case, $\ell_t - \ell_{t+1} \geq 0$ so the drift can only be downward. With similar arguments, we can show that in that case the drift is at least $\frac{j_t(1-p)}{n|\mathcal{P}_t|}$ which is greater than $\frac{j_t(1-p)}{n|\mathcal{P}_t|} - \frac{6p}{|\mathcal{P}_t|}$.

In both cases, the drift is greater than

$$\frac{j_t(1-p)}{n|\mathcal{P}_t|} - \frac{6p}{|\mathcal{P}_t|}.$$

This is enough to conclude by the multiplicative drift theorem [Doerr *et al.*, 2012b] that the lowest value is reached in $O(n^2 \log(n))$ when $p \leq \frac{1}{14n}$. As a consequence of Lemma 1, it will remain in $\mathcal{V}_t(\mathcal{P}_t)$. Symmetrically, the time to reach the highest value is identical. □

4.3 Filling the Pareto Front Once the Extreme Values are Found

To derive the total time, we additionally analyze the average time to fill the full Pareto front after the extreme values are found.

Theorem 5. *We consider the same algorithm, but where the extreme elements are initially in the set, that is $0, n \in \mathcal{V}_0(\mathcal{P}_0)$. We note this event Y_0. Then there exists a constant $\alpha > 0$ such that*

$$E_{Y_0}[T_{\text{total}}] = O(n^2).$$

Proof. The proof relies on the fact that since the extreme values have been found, we know there are elements $x, y \in \mathcal{P}_t$ verifying $\mathcal{V}_t(x) = 0, \mathcal{V}_t(y) = n$ so $f(x) \leq 1$ and $f(y) \geq n - 1$.

If $\left[1..\left\lfloor \frac{n}{2} \right\rfloor\right] \setminus f(\mathcal{P}_t)$ is non-empty, then let us define

$$m = \min\left(\left[1..\left\lfloor \frac{n}{2} \right\rfloor\right] \setminus f(\mathcal{P}_t)\right).$$

Since $x \in \mathcal{P}_t$, $\left[0..\left\lfloor \frac{n}{2} \right\rfloor\right] \cap f(\mathcal{P}_t)$ is non-empty. So there is an element $z \in \mathcal{P}_t$ such that $f(z) = m - 1$. The probability of selecting it to generate the offspring x'_t is $\frac{1}{|\mathcal{P}_t|} \geq \frac{1}{n+1}$. The probability that this offspring ends up with a real value m is $\frac{n-f(z)}{n} \geq \frac{1}{2}$. The probability of evaluating correctly this offspring is $1-p$. A symmetrical argument can be made for $\left[\left\lceil \frac{n}{2} \right\rceil..n-1\right] \setminus f(\mathcal{P}_t)$ which yields the same result.

Hence, we see that the variable counting elements in the population that are not extreme elements and are correctly evaluated features an upward drift of at least $\frac{1-p}{2n}$. For the downward drift, observe that no more than one element is deleted at each step, and that elements that are correctly evaluated can only be lost if the offspring is miss-evaluated, which happens with probability p. Thus, the total drift on this count variable is at least $\frac{1-p}{2n} - p$.

As for the variable counting correctly evaluated extreme elements, an argument similar to the proof of Theorem 4 shows it features a positive drift component of at least $\frac{1}{n+1}\frac{1}{n}(1-p)$ and a negative drift of no more than $\frac{12p}{n}$, and so a total drift of at least $\frac{1}{n+1}\frac{1}{n}(1-p) - \frac{12p}{n}$.

For a well-chosen constant $\alpha > 0$, if $p \leq \frac{\alpha}{n}$ then both drifts are positive. By considering a weighted sum of the two count variables, with a weight of 1 for the first and a weight of n for the second, the additive drift theorem [He and Yao, 2001] yields the desired result. □

So, the total runtime of the SEMO without reevaluation is $O(n^2 \log(n)) + O(n^2) = O(n^2 \log(n))$ as announced.

We observe that this second phase of the run is finished in a relatively short time. The reason for this is that once we have the extremal elements in the population, we can generate missing solutions (in particular, those which have been lost due to faulty fitness evaluations) from a neighboring solution closer to the boundaries. This is relatively easy since mutation has a constant probability to generate the desired offspring from such a parent. Here we profit from the diversity mechanism of the SEMO, which tries to keep one solution per objective value in the population. This seems to be the main reason for the relatively high robustness of the SEMO to noise, and we would speculate that other MOEAs profit from their diversity mechanisms (needed to construct the whole Pareto front) in a similar fashion.

5 Runtime Analysis of the SEMO With Reevaluation

In this section, we analyze the expected number of iterations of the main loop needed to reach the Pareto front in the SEMO with the SEMO with reevaluation 2. We define the stopping time

$$\begin{aligned} T_{\text{total}} &\coloneqq \min\{t \mid g(\mathcal{P}_t) \text{ is equal to the Pareto front}\} \\ &= \min\{t \mid f(\mathcal{P}_t) = [0..n]\} \end{aligned}$$

and show the following theorem.

Theorem. *There is a constant $\beta > 0$ such that the following holds. For all noise rates $p \leq \frac{\beta}{n^2}$, the SEMO with reevaluations finds the complete Pareto front of the* ONEMINMAX *benchmark in expected time*

$$E[T_{\text{total}}] = O(n^2 \log(n)).$$

We will assume in the whole section that $p \leq \frac{\beta}{n^2}$, with $\beta > 0$ a small enough constant. We use $\mathcal{P}_t$ for the population at time t, $\mathcal{P}'_t = \mathcal{P}_t \cup \{x'_t\}$ for the population plus the mutated vector. L_t is the cardinality of $f(\mathcal{P}_t)$ and L'_t the cardinality of $\mathcal{P}'_t$. Some of these notations are already used in the definition of Algorithm 2.

5.1 Time to Find the Extreme Values

Here, we define $T_{\text{ex}} = \min\{t \mid 0, n \in f(\mathcal{P}_t)\}$ in terms of real values. Contrary to the other algorithm, these extreme values can be lost because of the noisy reevaluations. We show that still $E[T_{\text{ex}}] = O(n^2 \log(n))$.

To prove this result, we consider the variable $d_t = n + \min f(\mathcal{P}_t) - \max f(\mathcal{P}_t)$ and apply the multiplicative drift theorem [Doerr *et al.*, 2012b] to it. We note that the mutation step gives a drift (towards zero) of order at least $\frac{d_t}{n^2}$, same as in a non-noisy setting. Different from the noiseless case, the selection can then lead to drift away from zero of order $\mathcal{O}(p)$. Note that in any case, d_t can change by at most one. By our assumption on p the drift into the wrong direction is small compared to the drift towards zero, so asymptotically we have a "multiplicative" drift of order at least $\frac{d_t}{n^2}$, which allows a straight-forward application of the multiplicative drift theorem to prove the announced result.

5.2 Computing the Pareto Front Once the Extreme Values are Found

We deal with the last phase by regarding an artificially modified process of the original SEMO with reevaluation. We denote this new process as the K-extreme values keeping SEMO, $K \in \mathbb{N} \cup \{+\infty\}$. This process is identical to the SEMO with reevaluation, except that it will tzappend the extreme elements to the population at the end of each iteration if they have been lost and if less than K iterations have been executed. Variables related to this new process feature a hat and a super-index K (e.g. $\hat{\mathcal{P}}_t^K$). In the whole subsection, we will use the event $Z_0 = (\{0, n\} \subset f(\mathcal{P}_0))$. We also define L_t as the cardinality of $f(\mathcal{P}_t)$.

By using the additive drift theorem [He and Yao, 2001] on the variable $\hat{L}_t^K$, we show the following lemma.

Lemma 6.

$$\Pr_{Z_0}[\hat{T}_{total}^K \geq K] = \mathcal{O}\left(\frac{n^2}{K}\right).$$

The following lemma shows that the two algorithms behave in the same way with high probability.

Lemma 7. *For any $K \in \mathbb{N}$, we have*

$$\Pr_{Z_0}[T_{\text{total}} = \hat{T}_{total}^K] \geq (1-p)^{10K}.$$

Indeed, of the classical process does not lose its extreme values in the K first iterations, then it is identical to the K-extreme values keeping process.

We can finally deduce the last result.

Lemma 8. *There exist constants $M > 0$ and $\mu \in (0,1)$ such that for any $n \in \mathbb{N}$, with $t_0 = Mn^2 \log(n)$ and any distribution π of the initial population $\mathcal{P}_0$,*

$$\Pr_\pi[T_{\text{total}} > t_0] \leq \mu.$$

This lemma states that the probability for the process, starting in any position, to end in time $t_0 = \mathcal{O}(n^2 \log(n))$ time is bounded by a constant. Therefore, the average number of phases of length t_0 is constant, hence the final result.

6 Lower Bound on the SEMO With Reevaluation

In this section, we will show that the previous $\frac{\beta}{n^2}$ bound for p for the SEMO with reevaluation to run in $O(n^2 \log(n))$ is nearly tight. Namely, if $p = \omega(\frac{\log(n)}{n^2})$ and for some constant $\lambda \in (0,1)$ we have $p \leq \frac{\lambda}{n}$, then the runtime is super-polynomial.

Here, we will use the notation from the previous section. Our main ingredient for the proof will be the simplified drift theorem [Oliveto and Witt, 2011; Oliveto and Witt, 2012] used on the random sequence $(L_t)_{t\in\mathbb{N}}$, where $L_t = |f(\mathcal{P}_t)|$.

In the whole proof, we will assume that $p = \omega(\frac{\log(n)}{n^2})$ and that there exists a constant $\lambda \in]0,1[$ such that $p \leq \frac{\lambda}{n}$.

Lemma 9. *We denote $a_n = \max(n + 1 - \beta pn^2, \frac{3n}{4})$, with $\beta > 0$ to be determined. There exists a constant $\delta > 0$ such that $E[L_t - L_{t+1} \mid L_t, L_t \geq a_n] \geq \delta np$.*

This lemma indicates the presence of a negative drift of L_t. Indeed, when that variable is close to n, each wrong evaluation will cause on average a deletion. This effect will dominate the gains from the mutation if the noise rate is big enough. We then show that

$$\Pr[|L_t - L_{t+1}| \geq j \mid L_t] = O_{j\to+\infty}(\lambda^j).$$

These are the two hypotheses for the Simplified Drift Theorem, as stated in [Oliveto and Witt, 2012]. The theorem implies the existence of a constant $B > 0$ such that $\Pr[T_{\text{total}} \leq 2^{Bpn^2}] \leq 2^{-\Omega(pn^2)}$. As $pn^2 = \omega(\log(n))$, this yields the final result.

7 Conclusion

We conducted the first mathematical runtime analysis, and to the best of our knowledge also the first analysis at all, aimed at understanding how MOEAs without particular adjustments behave in the presence of noise. Taking the SEMO algorithm and the OneMinMax benchmark with one-bit noise as an example simple enough to admit a mathematical analysis, we show that noise rates up to $p = O(1/n)$ can be tolerated without suffering from an asymptotic increase of the expected runtime (whereas runtime we regard the time it takes until for the first time the population of the algorithm witnesses the full Pareto front). This robustness is very close to the $p = O(\log(\log(n))/n)$ noise tolerance known for the much simpler single-objective OneMax problem and suggests that MOEAs, despite the more complex problem of finding the whole Pareto front, can stand similar noise levels as single-objective EAs.

Interestingly, our result only holds when solutions are evaluated only once and the algorithm then continues with this, possibly faulty, objective value. If solutions are reevaluated whenever they are compared to other solutions (here in each iteration), then any noise rate $p = \omega(\log(n)/n^2)$ leads to a super-polynomial runtime. This is in drastic contrast to single-objective evolutionary computation, where the general recommendation (backed up by an analysis of an ant colony optimizer) is to reevaluate each solution when its fitness is relevant so that a single faulty fitness evaluation cannot harm the future optimization process for a long time.

From the proofs of our results we are optimistic that our general finding that MOEAs without particular adjustments are robust to a moderate amount of noise is not restricted to the particular setting we analyzed, but holds for a broad range of algorithms and benchmarks. A reasonable next step to support this belief would be to regard the global SEMO algorithm (using bit-wise mutation instead of one-bit flips) and a bit-wise noise model (where the fitness observed for x is the fitness of a search point obtained from x by flipping each bit independently with probability $q = p/n$). With this scaling, we would expect very similar results to hold as shown in this work. We note that this appears to be a problem very close to ours, but past research has shown that both global mutation in MOEAs and bit-wise noise can render mathematical analyses much harder (different from the SEMO, there is no good lower bound for the global SEMO on the LOTZ benchmark [Doerr *et al.*, 2013]; the first mathematical runtime analysis in the presence of noise [Droste, 2004], namely one-bit noise, was extended to bit-wise noise only more than ten years later [Gießen and Kötzing, 2016] and with much deeper methods).

A second interesting direction to extend this work would be by regarding other problems. The OneMinMax problem has the particular properties that all solutions lie on the Pareto front and the one-bit flips are sufficient to explore the whole front. Hence analyses for the COCZ or LOTZ benchmarks having non-Pareto optimal solutions [Laumanns *et al.*, 2004] or the OneJumpZeroJump benchmark having larger gaps in the Pareto front [Doerr and Zheng, 2021] would be very interesting. Understanding how existing analyses for combinatorial optimization problems such as [Neumann, 2007; Cerf *et al.*, 2023] extend to the noisy setting could be a subsequent step.

A third interesting direction for future research would be to regard posterior noise models (where the noisy objective value of a search point is its true objective value modified in some stochastic way, e.g., by adding Gaussian noise). For single-objective evolutionary algorithms, often comparable results could be obtained with similar arguments [Gießen and Kötzing, 2016; Dang-Nhu *et al.*, 2018]. Since posterior noise can lead to objective values that cannot be obtained as noiseless fitness values, we have some doubts that our results extend to posterior noise as well. In particular, we would speculate that for posterior noise, different from what our result suggests for prior noise, reevaluating search points is the better strategy as otherwise it is not clear how to remove a noisy objective value that cannot occur as noiseless one.

Acknowledgments

This work was supported by a public grant as part of the Investissements d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH and a fellowship via the International Exchange Program of École Polytechnique.

References

[Akimoto *et al.*, 2015] Youhei Akimoto, Sandra Astete Morales, and Olivier Teytaud. Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Theoretical Computer Science*, 605:42–50, 2015.

[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.

[Bianchi *et al.*, 2009] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009.

[Boonma and Suzuki, 2009] Pruet Boonma and Junichi Suzuki. A confidence-based dominance operator in evolutionary algorithms for noisy multiobjective optimization problems. In *IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2009*, pages 387–394. IEEE Computer Society, 2009.

[Brockhoff, 2011] Dimo Brockhoff. Theoretical aspects of evolutionary multiobjective optimization. In Anne Auger and Benjamin Doerr, editors, *Theory of Randomized Search Heuristics*, pages 101–140. World Scientific Publishing, 2011.

[Cerf *et al.*, 2023] Sacha Cerf, Benjamin Doerr, Benjamin Hebras, Jakob Kahane, and 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. To appear.

[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.

[Dang-Nhu *et al.*, 2018] Raphaël Dang-Nhu, Thibault Dardinier, Benjamin Doerr, Gautier Izacard, and Dorian Nogneng. A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 1467–1474. ACM, 2018.

[Ding *et al.*, 2006] Hongwei Ding, Lyès Benyoucef, and Xiaolan Xie. A simulation-based multi-objective genetic algorithm approach for networked enterprises optimization. *Engineering Applications of Artificial Intelligence*, 19:609–623, 2006.

[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. *CoRR*, abs/2305.10259, 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 Sutton, 2019] Benjamin Doerr and Andrew M. Sutton. When resampling to cope with noise, use median, not mean. In *Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 242–248. ACM, 2019.

[Doerr and Wietheger, 2023] Benjamin Doerr and Simon Wietheger. 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. To appear.

[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.*, 2012a] Benjamin Doerr, Ashish Ranjan Hota, and Timo Kötzing. Ants easily solve stochastic shortest path problems. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 17–24. ACM, 2012.

[Doerr *et al.*, 2012b] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.

[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.

[Droste, 2004] Stefan Droste. Analysis of the (1+1) EA for a noisy OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2004*, pages 1088–1099. Springer, 2004.

[Fieldsend and Everson, 2015] Jonathan E. Fieldsend and Richard M. Everson. The rolling tide evolutionary algorithm: a multiobjective optimizer for noisy optimization problems. *IEEE Transactions on Evolutionary Computation*, 19:103–117, 2015.

[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.

[Gießen and Kötzing, 2016] Christian Gießen and Timo Kötzing. Robustness of populations in stochastic environments. *Algorithmica*, 75:462–489, 2016.

[Gutjahr and Pichler, 2016] Walter J. Gutjahr and Alois Pichler. Stochastic multi-objective optimization: a survey on non-scalarizing methods. *Annals of Operations Research*, 236:475–499, 2016.

[Gutjahr, 2011] Walter J. Gutjahr. Recent trends in metaheuristics for stochastic combinatorial optimization. *Central European Journal on Computer Science*, 1:58–66, 2011.

[Gutjahr, 2012] Walter J. Gutjahr. Runtime analysis of an evolutionary algorithm for stochastic multi-objective combinatorial optimization. *Evolutionary Computation*, 20:395–421, 2012.

[He and Yao, 2001] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:51–81, 2001.

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

[Jin and Branke, 2005] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317, 2005.

[Laumanns *et al.*, 2002a] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10:263–282, 2002.

[Laumanns *et al.*, 2002b] 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.

[Lengler, 2020] Johannes Lengler. Drift analysis. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 89–131. Springer, 2020. Also available at https://arxiv.org/abs/1712.00964.

[Liefooghe *et al.*, 2007] Arnaud Liefooghe, Matthieu Basseur, Laetitia Jourdan, and El-Ghazali Talbi. Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem. In *Evolutionary Multi-Criterion Optimization, EMO 2007*, volume 4403, pages 457–471. Springer, 2007.

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

[Neumann *et al.*, 2020] Frank Neumann, Mojgan Pourhassan, and Vahid Roostapour. Analysis of evolutionary algorithms in dynamic and stochastic environments. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 323–357. Springer, 2020. Also available at https://arxiv.org/abs/1806.08547.

[Neumann, 2007] Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181:1620–1629, 2007.

[Nguyen *et al.*, 2015] Anh Quang Nguyen, Andrew M. Sutton, and Frank Neumann. Population size matters: rigorous runtime results for maximizing the hypervolume indicator. *Theoretical Computer Science*, 561:24–36, 2015.

[Oliveto and Witt, 2011] Pietro S. Oliveto and Carsten Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59:369–386, 2011.

[Oliveto and Witt, 2012] Pietro S. Oliveto and Carsten Witt. Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation. *CoRR*, abs/1211.7184, 2012.

[Qian *et al.*, 2018] Chao Qian, Yang Yu, Ke Tang, Yaochu Jin, Xin Yao, and Zhi-Hua Zhou. On the effectiveness of sampling for evolutionary optimization in noisy environments. *Evolutionary Computation*, 26:237–267, 2018.

[Rakshit and Konar, 2015] Pratyusha Rakshit and Amit Konar. Differential evolution for noisy multiobjective optimization. *Artificial Intelligence*, 227:165–189, 2015.

[Stützle and Hoos, 2000] Thomas Stützle and Holger H. Hoos. MAX-MIN ant system. *Future Generation Computer Systems*, 16:889–914, 2000.

[Sudholt and Thyssen, 2012] Dirk Sudholt and Christian Thyssen. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica*, 64:643–672, 2012.

[Sudholt, 2021] Dirk Sudholt. Analysing the robustness of evolutionary algorithms to noise: refined runtime bounds and an example where noise is beneficial. *Algorithmica*, 83:976–1011, 2021.

[Teich, 2001] Jürgen Teich. Pareto-front exploration with uncertain objectives. In *Evolutionary Multi-Criterion Optimization, EMO 2001*, pages 314–328. Springer, 2001.

[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.

[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.