First Mathematical Runtime Analyses of Multi-Objective Evolutionary Algorithms for Multi-Valued Decision Variables

Mingfeng Li[1] , **Zheng Cheng**[1] , **Weijie Zheng**[1,2*] , **Benjamin Doerr**[3]
[1]School of Computer Science and Technology, National Key Laboratory of Smart Farm Technologies and Systems, International Research Institute for Artificial Intelligence, Harbin Institute of Technology, Shenzhen
[2]Pengcheng Laboratory, Shenzhen, China
[3]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
limingfeng@stu.hit.edu.cn, zhengweijie@hit.edu.cn, lastname@lix.polytechnique.fr

Abstract

Problems defined on binary decision spaces have been intensively studied in the theory of multi-objective evolutionary algorithms (MOEAs). In contrast, no mathematical runtime analyses exist so far for MOEAs dealing with decision variables that take a finite number $r > 2$ of values, despite the prevalence of such problems in practice. In this work, we begin to fill this research gap. We analyze how the classic SEMO algorithm with unit-strength local mutation computes the Pareto front of an r-valued counterpart of the classic OneMinMax benchmark. For the expected number of function evaluations until the Pareto front is covered by the population of this MOEA, we prove an upper bound of $O(n^2r^2 \log n)$ and a near-tight lower bound of $\Omega(n^2r(r+\log n))$. We can close the small remaining gap between these two bounds by considering a variant of the algorithm that accepts only strictly better solutions; for this variant, we show an upper bound of $O(n^2r(r + \log n))$, matching our lower bound (which also holds for this variant). Our results suggest that classic MOEAs encounter no significant additional difficulties when dealing with multi-valued decision variables. However, significantly more advanced tools may be required to obtain tight bounds for algorithms with more complex population dynamics.

1 Introduction

Evolutionary algorithms are widely used for solving complex optimization problems, in particular, for those with multiple conflicting objectives. Multi-objective evolutionary algorithms (MOEAs) address such problems by exploring the different trade-offs between the objectives in a single run. Well-known MOEAs such as NSGA-II, NSGA-III, SPEA2, and SMS-EMOA are widely used in both research and applications. Their empirical success has motivated a considerable body of theoretical works aimed at understanding how MOEAs work and why they perform well in practice. These include, among others, guarantees on the time required to cover the entire Pareto front [Laumanns *et al.*, 2002; Zheng and Doerr, 2023; Wietheger and Doerr, 2023; Zheng and Doerr, 2024; Ren *et al.*, 2024], analyses on the benefits of crossover operators [Doerr and Qu, 2023b; Dang *et al.*, 2024; Opris, 2026a], the role of diversity preservation and archiving mechanisms [Covantes Osuna *et al.*, 2020; Bian *et al.*, 2024; Ren *et al.*, 2025], the impact of non-elitist strategies [Zheng *et al.*, 2024; Li *et al.*, 2025b; Bian *et al.*, 2025; Bendahi *et al.*, 2025], and approximation guarantees for situations in which the full Pareto front is too costly to be computed [Horoba and Neumann, 2008; Zheng and Doerr, 2025; Deng *et al.*, 2025; Alghouass *et al.*, 2025; Li *et al.*, 2025a].

The vast majority of the theoretical works (in discrete optimization) have focused on problems defined on binary decision spaces. To the best of our knowledge, only two runtime analyses of MOEAs move beyond binary decision spaces, and they do so by considering the other extreme of unbounded integer-valued search domains [Rudolph, 2023; Doerr *et al.*, 2025b]. Among them, Rudolph [2023] analyzed subproblems of a multi-objective problem in integer search spaces, but the analysis remains a single-objective analysis. Following this work, Doerr et al. [2025b] studied evolutionary algorithms on unbounded integer domains, focusing on how different mutation strength distributions affect the runtime. Till now, no mathematical runtime analyses exist for MOEAs dealing with decision variables that take a finite number r of values that is larger than two.

Our Contributions: In this work, we undertake a first attempt to analyze MOEAs for problems with multi-valued decision variables with mathematical means. We first extend the classical bi-objective OneMinMax benchmark to an r-valued domain, $r > 2$. This benchmark preserves the simplicity and structural clarity of its binary counterpart, but (naturally) has a larger Pareto front. We then consider the classic SEMO algorithm with unit-strength local mutation, which can be regarded as the natural multi-valued analogue of one-bit mutation. For this algorithm, we prove an upper bound of $O(n^2r^2 \log n)$ and a lower bound of $\Omega(n^2r(r+\log n))$ on the expected number of iterations until the population covers the

*Corresponding author.

entire Pareto front. The dependence on r in these bounds is moderate. Although the search space has size r^n, the Pareto front of our benchmark grows only linearly with r. Since the runtime is measured by the time to cover this front, its dependence on r is tied more to the linearly growing front than to the exponentially growing search space. The (small) gap between these bounds stems from the population dynamics, which (as this work shows) are much more complex in the multi-valued setting.

We further consider a very mild modification of the tie-breaking rule in the selection of the next population, namely that we accept only strictly better solutions (that is, in case of equal function values, we keep the old solution and discard the new one). This modification allows for a more tractable analysis of the population dynamics, and allows us to prove an improved upper bound of $O(n^2 r(r + \log n))$, which matches our lower bound (that is valid also for this modification of the algorithm).

We do not see any reason why the modified algorithm should be faster than the original one. Our (simple) experimental results also could not detect a significant difference (rather the modified algorithm appeared slightly slower). To check that this observation is not specific to r-valued OneMinMax, where all solutions are Pareto optimal, we additionally introduce and experimentally study an r-valued analogue of LOTZ. This benchmark has the same Pareto-front size, but, among many other differences, most solutions are not Pareto-optimal. Again, we observe no noticeable runtime difference between the original and the modified algorithm. For these reasons, we conjecture that the $O(n^2 r(r + \log n))$ bound, proven for the modified SEMO on r-valued OneMinMax and matching our lower bound, is also the true runtime for the original SEMO. Proving this conjecture, unfortunately, seems out of reach with the current methods of this field, and might be a worthwhile challenge for the future, that could give deep insights into the more complex population dynamics in multi-valued search spaces.

2 Related Work

The theory of randomized search heuristics is a relatively young subarea of artificial intelligence. While first mathematical runtime analyses for single-objective algorithms appeared already in the 1990s, say [Mühlenbein, 1992; Bäck, 1993], the first such works for multi-objective algorithms took around another ten years [Laumanns *et al.*, 2002] and the truly practically relevant algorithms could only be analyzed very recently [Li *et al.*, 2016; Zheng and Doerr, 2023; Wietheger and Doerr, 2023; Bian *et al.*, 2025; Ren *et al.*, 2024].

As stated already in the introduction, there are no theoretical works analyzing how an MOEA solves a problem defined over decision variables taking a finite number $r > 2$ of values. Closest to our work are the two recent papers [Rudolph, 2023; Doerr *et al.*, 2025b] that analyze discrete multi-objective problems defined on $\mathbb{Z}^n$. Only the second work considers a true MOEA, namely the SEMO regarded also in this work and the related GSEMO. The main difficulty in this work, however, is dealing with the unboundedness of the set of values taken by the decision variables. Consequently, the methods employed there are of little help for our problem setting.

In contrast, in the theory of single-objective evolutionary algorithms a few sporadic works on multi-valued optimization exist. Doerr et al. [2011] analyzed the $(1 + 1)$ EA with uniform mutation strength, that is, resetting a component to a random other value, on linear functions over $\{0, 1, 2\}^n$. They proved an expected runtime of $O(n \log n)$, matching the well-known bound for the binary case. The result was then extended by Doerr and Pohl [2012], where they studied r-valued linear functions and derived an upper bound of $O(rn \log n) + O(r^3 n \log \log n)$ and a lower bound of $\Omega(rn \log n)$.

Subsequent work considered different and more complex mutation operators and in return only regarded the simpler OneMax problem. Kötzing et al. [2015] analyzed the $(1 + 1)$ EA with unit-strength mutation on dynamically changing OneMax where each dimension has r different possible values. Doerr et al. [2018] considered the $(1 + 1)$ EA with three different mutation strengths and proved asymptotically tight runtimes for a variety of OneMax-type test functions over an alphabet of size r. They further analyzed the performance of randomized local search, a variant of the $(1 + 1)$ EA using local mutation, with a self-adjusting mutation strength and proved an expected optimization time of $\Theta(n(\log n + \log r))$, which is optimal among all comparison-based search heuristics. Harder et al.[2024] studied the $(1 + 1)$ EA with different mutation operators on the unbounded integer search space $\mathbb{Z}^n$. All these works are hardly comparable to ours since they regard single-objective optimization via an algorithm with parent and offspring population equal to one. Hence the population dynamics are less interesting here.

In [Qian *et al.*, 2018a; Qian *et al.*, 2018b], submodular functions with multi-valued discrete domains are studied. Both works formulate a constrained single-objective problem as a bi-objective problem. They then use problem-specific algorithms and algorithms similar to classic MOEAs to solve these. However, since their objective is to find a single solution with a particular constraint satisfaction (equivalent to a particular value of the second objective), they do not analyze how the full Pareto front is computed or approximated, but only how long it takes until such a particular solution is contained in the population. For that reason, we could not derive deeper insights on multi-objective optimization from these works, and feel that they should be viewed as single-objective optimization works.

More recently, estimation-of-distribution algorithms (EDAs) have been analyzed in multi-valued domains, see, e.g., [Ben Jedidia *et al.*, 2024; Adak and Witt, 2024]. Due to the very different algorithmic concept of an EDA, these works are hardly comparable to ours.

Overall, these results demonstrate that runtime analysis in multi-valued decision spaces is feasible in the single-objective setting, but typically requires more involved techniques than in the binary case.

3 Preliminaries

Throughout the paper, for all $a \leq b$, we write $[a..b] := \{i \in \mathbb{Z} \mid a \leq i \leq b\}$ to denote the interval of integers between a and b.

3.1 Multi-Objective Optimization

This paper considers multi-objective optimization problems defined on the search space $[0..r-1]^n$, where $r > 2$. Such a problem is given by $f : [0..r-1]^n \to \mathbb{R}^m$. Our goal is to maximize the objectives. For two solutions $x, y \in [0..r-1]^n$, we say that x *weakly dominates* y if $f_i(x) \geq f_i(y)$ holds for all $i \in [1..m]$, denoted as $x \succeq y$. If, in addition, there exists at least one objective j such that $f_j(x) > f_j(y)$, then x is said to *dominate* y, denoted as $x \succ y$. A solution is called *Pareto optimal* if it is not dominated by any other solution in the search space. The set of all Pareto-optimal solutions is referred to as the *Pareto set*. The *Pareto front* consists of the objective vectors corresponding to these solutions. As common in the evolutionary computation theory community [Zhou *et al.*, 2019; Doerr and Neumann, 2020], we measure the runtime by the number of function evaluations for the population to cover the entire Pareto front. We often use asymptotic notation (big-Oh notation, Landau symbols). If so, then the underlying variable is the problem size n. We allow other quantities to be functionally dependent on n, for example, the number r of different values taken by the decision variables. This is natural, for example, in rooted tree problems in graphs on n vertices, we have $n-1$ decision variables (describing the predecessor of a non-root node on the path to the source), and each of them can take any other node as value (so there are $n-1$ different values).

3.2 The Multi-Valued Algorithm

The *simple evolutionary multi-objective optimizer (SEMO)* was the algorithm first used in a mathematical runtime analysis of an MOEA [Laumanns *et al.*, 2002]. It is still the algorithm best understood from a theoretical perspective. Starting from a single random solution, the algorithm repeatedly selects a random parent from the population, generates an offspring via mutation, and keeps as next population a largest pair-wise non-dominated subset of the old population and the offspring, giving preference to the offspring in case of ties.

In the classic binary setting, the SEMO employs one-bit mutation, which flips a single bit chosen uniformly at random from the n bit positions. In the multi-valued setting considered in this work, we employ a *unit-strength local mutation* operator, which has, to the best of our knowledge, first been proposed and argued for in the single-objective work [Doerr *et al.*, 2018]. This operator, as one of several alternatives, was also studied in the recent work on multi-objective optimization for unbounded integer-valued problems [Doerr *et al.*, 2025b]. When applying this mutation operator to an individual $x \in [0..r-1]^n$, one position $i \in [1..n]$ is chosen uniformly at random, and the value x_i of this position is increased or decreased by 1 with equal probability. A mutation that would lead to a value outside the allowed interval $[0..r-1]$ is considered infeasible and discarded (that is, the parent of the mutation operation is returned as offspring). The pseudocode of the algorithm is given in Algorithm 1.

Algorithm 1: The r-valued SEMO with unit-strength local mutation to maximize $f : [0..r-1]^n \to \mathbb{R}^m$.

```
Initialize x ∈ [0..r−1]^n uniformly at random and set P_0 ← {x};
for t = 0, 1, 2, ... do
    Select one individual x uniformly at random from P_t;
    Generate y via choosing one entry of x uniformly at random and modifying it by ±1 with equal probability;
    if there is no z ∈ P_t such that z ≻ y then
        P_{t+1} ← {z ∈ P_t | y ⊁ z} ∪ {y};
    else
        P_{t+1} ← P_t;
```

3.3 The Multi-Valued Benchmark

The ONEMINMAX problem is the best studied benchmark in the theory of MOEAs. Its two objectives are the number of zeros in the bit string and the number of ones. This is equivalent to saying that the objectives are the sum of the bit values and the sum of the inverses of the bit values. We therefore extend the binary ONEMINMAX benchmark to the multi-valued search space $[0..r-1]^n$ by letting the first objective be the sum of the values of the entries of x and the second objective be the sum of the inverses. This choice of the objective order makes the definition closer to the COCZ benchmark, but has no effect on the theoretical results. Moreover, the same objective order was used in [Zheng, 2026], where the corresponding binary problem was called ONEMAXMIN. Since in the single-objective multi-valued setting, Adak and Witt [2024] called this first objective G-ONEMAX, we refer to our multi-valued extension of the ONEMINMAX benchmark as the *generalized* ONEMAXMIN problem, denoted by G-ONEMAXMIN.

Definition 1. *Let $n \in \mathbb{N}$ and $r \in \mathbb{N}_{\geq 2}$. The* G-ONEMAXMIN *function $f(x) = (f_1(x), f_2(x)) : [0..r-1]^n \to \mathbb{R}^2$ is defined by*

$$f_1(x) = \sum_{i=1}^{n} x_i, \qquad f_2(x) = \sum_{i=1}^{n} (r-1-x_i).$$

The following lemma characterizes the Pareto set and the Pareto front of G-ONEMAXMIN. As in the binary case, every solution in the search space is Pareto optimal, but the size of the Pareto front also increases with r. Due to space limitations, all proofs are omitted. A complete version is available on arXiv [Li *et al.*, 2026].

Lemma 2. *The Pareto set of* G-ONEMAXMIN *is*

$$S^* = [0..r-1]^n.$$

The Pareto front is

$$F^* = \{(a, nr-n-a) \mid a \in [0..nr-n]\},$$

whose size is $nr-n+1$.

As a second benchmark in the experimental section, we use a multi-valued analogue of the classic LOTZ benchmark. With very few solutions Pareto-optimal, it can be seen as a problem complementary to G-ONEMAXMIN. The first objective follows the idea of the r-valued LEADINGONES benchmark [Ben Jedidia *et al.*, 2024], but refines it by also giving value-progress credit to the first position after the certified prefix. Hence a complete prefix of entries equal to $r-1$ contributes fully, the first subsequent position contributes its current value, and all later positions contribute nothing. Symmetrically, the second objective measures the same progress from the right towards value 0. Values lying strictly between the active leading position and the active trailing position are deliberately ignored: they are neither part of the prefix of $r-1$ values nor part of the suffix of 0 values. This construction leaves only a narrow set of Pareto-optimal solutions, while preserving a Pareto front of size $n(r-1)+1$, as in G-ONEMAXMIN.

Definition 3. *Let $n \in \mathbb{N}$ and $r \in \mathbb{N}_{\geq 2}$. The* G-LOTZ *function $f(x) = (\text{G-LO}(x), \text{G-TZ}(x)) : [0..r-1]^n \to \mathbb{R}^2$ is defined by*

$$\text{G-LO}(x) = \sum_{i=1}^{n} \left(\prod_{j=1}^{i-1} \mathbf{1}[x_j = r-1] \right) x_i,$$

$$\text{G-TZ}(x) = \sum_{i=1}^{n} \left(\prod_{j=i+1}^{n} \mathbf{1}[x_j = 0] \right) (r-1-x_i).$$

For $r = 2$, this definition coincides with the usual binary LOTZ benchmark. The following lemma characterizes the Pareto-optimal solutions and Pareto front of this benchmark.

Lemma 4. *For each $k \in [0..n(r-1)]$, write $q = \lfloor k/(r-1) \rfloor$ and $a = k - q(r-1)$. Let*

$$x^{(k)} = \begin{cases} (\underbrace{r-1, \dots, r-1}_{q}, a, \underbrace{0, \dots, 0}_{n-q-1}), & \text{if } q < n, \\ (r-1, \dots, r-1), & \text{if } q = n. \end{cases}$$

The Pareto set of G-LOTZ *is the set of solutions*

$$S^*_{\text{G-LOTZ}} = \{x^{(k)} \mid k \in [0..n(r-1)]\}$$

and its Pareto front is

$$F^*_{\text{G-LOTZ}} = \{(k, n(r-1)-k) \mid k \in [0..n(r-1)]\}.$$

Hence both have size $n(r-1)+1$.

4 Runtime Analysis

4.1 Mathematical Tools

Before presenting the mathematical runtime analyses, we first introduce several tools that will be used later. To bound the objective value of the initial solution, we employ the following Chernoff bound.

Theorem 5 ([Chernoff, 1952])**.** *Let $X_1, \dots, X_n$ be independent random variables taking values in $[0, r-1]$. Let $X = \sum_{i=1}^{n} X_i$. Let $\delta \in [0,1]$. Then*

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

The following tail bound for sums of independent geometric random variables provides strong guarantees for analyzing population dynamics in our lower-bound analysis.

Theorem 6 ([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 $T^* := \sum_{i\in[k]} D_i$, $s := \sum_{i\in[k]} \frac{1}{p_i^2}$, and $p_{\min} := \min\{p_i \mid i \in [k]\}$. Then for all $\lambda \in \mathbb{R}_{\geq 0}$, we have*

$$\Pr[T^* \geq \mathbb{E}[T^*] + \lambda] \leq \exp\left(-\frac{1}{4}\min\left\{\frac{\lambda^2}{s}, \lambda p_{\min}\right\}\right),$$

$$\Pr[T^* \leq \mathbb{E}[T^*] - \lambda] \leq \exp\left(-\frac{\lambda^2}{2s}\right).$$

Drift analysis is a fundamental tool in the runtime analysis of randomized search heuristics [He and Yao, 2001]. The multiplicative drift theorem [Doerr *et al.*, 2012] applies when the expected progress is proportional to the current distance to the target.

Theorem 7 ([Doerr *et al.*, 2012])**.** *Let $X^{(0)}, X^{(1)}, \dots$ be a random process taking values in $S := \{0\} \cup [s_{\min}, \infty) \subseteq \mathbb{R}$. Assume that $X^{(0)} = s_0$ with probability one. Assume that there is a $\delta > 0$ such that for all $t \geq 0$ and all $s \in S$ with $\Pr[X^{(t)} = s] > 0$ we have*

$$\mathbb{E}[X^{(t+1)} \mid X^{(t)} = s] \leq (1-\delta)s.$$

Then $T := \min\{t \geq 0 \mid X^{(t)} = 0\}$ satisfies

$$\mathbb{E}[T] \leq \frac{\ln(s_0/s_{\min}) + 1}{\delta}.$$

4.2 Upper Bound for the Runtime of the SEMO on G-ONEMAXMIN

We first bound the size of a pair-wise non-dominated set w.r.t. G-ONEMAXMIN.

Lemma 8. *Let $P \subseteq [0..r-1]^n$ be any set of solutions that none weakly dominates the other w.r.t.* G-ONEMAXMIN. *Then $|P| \leq nr - n + 1$.*

The following theorem provides an upper bound on the expected number of iterations until Algorithm 1 covers the whole Pareto front of G-ONEMAXMIN. In contrast to the binary ONEMINMAX problem, where the objective value directly determines the number of flippable bits, solutions with the same objective value in our setting have a much more diverse structure. Hence a key part of the proof is to estimate the number of entries of a solution that can be mutated to derive an improvement in one objective. More precisely, once the population contains an objective value, a neighboring value can be created by selecting a corresponding individual and mutating one suitable non-boundary coordinate in the correct direction. Summing the resulting waiting times over all missing front values yields the upper bound.

Theorem 9. *Consider Algorithm 1 optimizing* G-ONEMAXMIN. *Then the expected number of iterations until the population covers the whole Pareto front is $O(n^2 r^2 \log n)$.*

4.3 Lower Bound for the SEMO Algorithm

To derive a lower bound on the expected runtime of the SEMO, a technical difficulty is controlling the population size before the Pareto front is fully covered. Following the proof strategy introduced in [Doerr *et al.*, 2025a], we consider a *delayed* SEMO that simplifies the analysis in this phase. The delayed SEMO is identical to the original SEMO (Algorithm 1) except for the parent selection in line 3, where it proceeds as follows. In each iteration, a value $i \in [0..nr-n]$ is chosen uniformly at random. If there is no individual $z \in P$ with $f_1(z) = i$, the iteration is skipped; otherwise, the unique individual x with $f_1(x) = i$ is selected and the algorithm proceeds exactly as in line 4 of Algorithm 1. This modification may introduce iterations that do not change the state of the algorithm, but it has no influence otherwise. Hence it can only slow down the process. In particular, structural statements like the shape of the population at a particular stopping time are not affected. The advantage of working with the delayed SEMO is the more regular way parents are selected: In each iteration, the probability to choose a parent with some existing objective value is exactly the reciprocal of the size of the Pareto front. Since we will introduce another variant of the SEMO algorithm later on, we present the pseudocode of the delayed SEMO algorithm here to clearly distinguish and refer to the different algorithms. See Algorithm 2.

Algorithm 2: The delayed r-valued SEMO with unit-strength local mutation to maximize $f : [0..r-1]^n \to \mathbb{R}^m$.

```
1  Initialize x ∈ [0..r−1]^n uniformly at random and set P_0 ← {x};
2  for t = 0, 1, 2, ... do
3      Choose i ∈ [0..nr − n] uniformly at random;
4      if there is no x ∈ P_t such that f_1(x) = i then
5          Continue;
6      else
7          Select one individual x such that f_1(x) = i;
8      Generate y via choosing one entry of x uniformly
         at random and modifying it by ±1 with equal
         probability;
9      if there is no z ∈ P_t such that z ≻ y then
10         P_{t+1} ← {z ∈ P_t | y ⊁ z} ∪ {y};
11     else
12         P_{t+1} ← P_t;
```

We use Algorithm 2 for analysis until the size of the population has grown to linear order $\Theta(n(r-1))$ and then switch back to the original SEMO to then analyze the time required to cover the Pareto front. In Lemma 10, we show that the size of the population grows to $\Theta(n(r-1))$ within $O(n^2(r-1)^2)$ iterations with high probability. The key idea of the proof is that as long as the population is still small, there always exists a missing neighboring objective value, which can be generated with a constant probability conditioned on selecting a suitable individual. Repeatedly filling such gaps, the size of the population increases steadily until it reaches the desired order. This result provides a lower bound on the population size that will be crucial in the final lower-bound analysis.

Lemma 10. *Consider Algorithm 2 optimizing* G-ONEMAXMIN. *Then with probability* $1-\exp(-\Theta(n))$*, the population contains at least* $n(r-1)/130$ *individuals after at most* $8n^2(r-1)^2/129$ *iterations.*

For a population P, we call the two extreme objective vectors $(0, n(r-1))$ and $(n(r-1), 0)$ the *Pareto borders*. We measure the distance of P to these borders by

$$d_{PF}(P) := \min\left\{ \min_{z\in P} f_1(z), \min_{z\in P} f_2(z) \right\}.$$

Hence $d_{PF}(P) = 0$ is necessary for covering the full Pareto front. The following lemma shows that, with high probability, the distance of the population to the Pareto borders is still considerable within $n^2(r-1)^2/16$ iterations. The proof idea is as follows. By a Chernoff bound, the random initial individual lies in a central region of the objective space with high probability, and is therefore far from both Pareto borders. Afterwards, unit-strength local mutation can move the population towards a fixed Pareto border only gradually, since each mutation can change the relevant objective value by at most one. Moreover, such progress requires selecting an individual with the minimum value in the corresponding objective. Combining these with a concentration argument for the resulting waiting times shows that the population cannot approach either Pareto border too quickly.

Lemma 11. *Consider Algorithm 2 optimizing* G-ONEMAXMIN. *Then with probability at least* $1-\exp(-\Theta(n))$*, in all iterations* $t \in [0..n^2(r-1)^2/16]$*, the distance of the population to the Pareto borders is at least* $n(r-1)/8$.

As mentioned above, we switch back to the analysis of the original SEMO algorithm. Conditioning on a sufficiently large population but a considerable distance to the Pareto borders, we analyze the expected time required to reach the Pareto borders. Since reaching one of the borders is necessary for covering the full Pareto front, it suffices to analyze the time needed to reach one fixed border, say the border with first objective value 0; the other case is symmetric. The proof splits this progress into a far-from-border phase and a close-to-border phase. In both phases, we upper-bound the probability of decreasing the remaining distance by one, which yields lower bounds on the required waiting times.

Lemma 12. *Consider Algorithm 1 optimizing* G-ONEMAXMIN*, starting with a population size of at least* $\frac{n(r-1)}{130}$ *and the distance to the Pareto borders of at least* $n(r-1)/8$*. Then, the expected number of iterations until the population covers the whole Pareto front is* $\Omega(n^2 r(r+\log n))$.

Combining the above three lemmas, we derive the following theorem.

Theorem 13. *Consider Algorithm 1 optimizing* G-ONEMAXMIN. *Then the expected number of iterations until the population covers the whole Pareto front is* $\Omega(n^2 r(r+\log n))$.

4.4 A Tighter Bound for a Variant of the SEMO Algorithm

We note that the two bounds proven so far are not absolutely tight, but differ by a factor of $O(\log n)$. Due to the complicated population dynamics in a run of an MOEA, tight runtime bounds are generally hard to obtain, and in fact, the only ones in this field we are aware of are those in [Laumanns *et al.*, 2002; Doerr and Zheng, 2021; Bossek and Sudholt, 2024; Doerr and Qu, 2023a; Doerr *et al.*, 2025a; Opris, 2026b]. To try to shed more light on the true runtime of the multi-valued SEMO, we now consider a mild modification of the SEMO. This variant is not introduced as an artificial algorithmic improvement. It differs from the original algorithm only in the tie-breaking rule in line 5 of Algorithm 1, where now we accept only offspring that are not even weakly dominated. In other words, if a solution with some objective value is already present in the population, then it is not replaced by a later solution with equal objective value. Since equal objective vectors represent the same point on the Pareto front, this modification affects only the representative solution stored for a front point, but not which objective vectors are already covered. We note that this is exactly the algorithm proposed in the original work introducing the SEMO [Laumanns *et al.*, 2002], and only later the research community gradually adopted the version presented in Algorithm 1. To avoid any ambiguity, the pseudocode of the algorithm regarded in this subsection can be found in Algorithm 3.

Algorithm 3: The variant of the r-valued SEMO that does not accept solutions which are weakly dominated. As before, we use unit-strength local mutation and maximize $f : [0..r-1]^n \to \mathbb{R}^m$.

```
1 Initialize x ∈ [0..r−1]^n uniformly at random and set P_0 ← {x};
2 for t = 0, 1, 2, ... do
3     Select one individual x uniformly at random from P_t;
4     Generate y via choosing one entry of x uniformly at random and modifying it by ±1 with equal probability;
5     if there is no z ∈ P_t such that z ⪰ y then
6         P_{t+1} ← {z ∈ P_t | y ⋡ z} ∪ {y};
7     else
8         P_{t+1} ← P_t;
```

For Algorithm 3, we can prove an upper bound that agrees with the previously shown lower bound (which is valid here as well). The proof builds on an idea from [Doerr *et al.*, 2018], where a potential function was used to analyze how randomized local search optimizes a variant of the r-valued single-objective ONEMAX problem. In our setting, the modified tie-breaking rule removes neutral replacements. Once a front point is covered by the population, its representative can no longer be changed by another solution with the same objective value. The analysis does not have to track neutral replacements among the many solutions with the same objective value. This is in sharp contrast to Algorithm 1, where such replacements may change the internal structure of a front point without changing its objective value. These neutral changes appear harmless algorithmically, but they obscure the drift of natural potential functions. For Algorithm 3, we can therefore define a potential function that decreases only when real progress towards one of the two extreme Pareto-optimal solutions, 0^n and $(r-1)^n$, is made. A multiplicative-drift argument then gives the desired upper bound. Since we do not believe that the small difference of accepting or not solutions with equal objective value changes the asymptotic runtime (which will be supported by our experiments), we see this result as a strong indication for the true runtime of the r-valued SEMO.

Theorem 14. *Consider optimizing* G-ONEMAXMIN *via Algorithm 1 with the modification that it accepts only strictly better solutions. Then the expected number of iterations until the population covers the whole Pareto front is* $\Theta(n^2 r(r + \log n))$.

5 Experiments

Above, we used Algorithm 3 as a model for the SEMO which we could analyze well enough to obtain asymptotically tight bounds for the runtime. Our strong belief was that the small difference in the selection procedure has little influence on the performance. We now support this belief with experiments on both G-ONEMAXMIN and G-LOTZ. The reason for including G-LOTZ is that G-ONEMAXMIN has the special property that every solution is Pareto optimal. Consequently, neutral replacements are particularly frequent and may appear more harmless than in problems with dominated regions. G-LOTZ removes this special feature while keeping the same Pareto-front size, and therefore provides a check for our comparison of the two tie-breaking rules.

For G-ONEMAXMIN, we conduct two sets of experiments. In the first set, we fix $r = 4$ and consider the problem sizes $n \in \{20, 40, 60, 80, 100\}$; in the second, we fix $n = 100$ and consider the r-values $r \in \{2, 3, 4, 5\}$. For each setting, we conduct 100 independent runs of both algorithms and record the first iteration after which the population covers the full Pareto front. We repeat the same experimental setup for G-LOTZ, which has the same front size $n(r-1)+1$ but only one Pareto-optimal solution for each objective vector. This allows us to test whether our empirical conclusions are robust beyond the special G-ONEMAXMIN case, where all solutions are Pareto optimal.

Figures 1 and 2 display the observed runtimes (averages and standard deviations). The results indicate not greater differences between the original SEMO algorithm (Algorithm 1) and its variant Algorithm 3; possibly the original SEMO performs slightly better. Figures 3 and 4 show the corresponding G-LOTZ results. Again Algorithm 3 behaves similarly to the original SEMO algorithm. Across both benchmarks and both parameter regimes, we observe no systematic effect of accepting or rejecting neutral replacements. Hence our experiments support our belief that the small variation in the selection does not drastically change the algorithm behavior, and more specifically, that our improved upper bound on the

runtime shown for Algorithm 3 is likely to hold also for the SEMO.

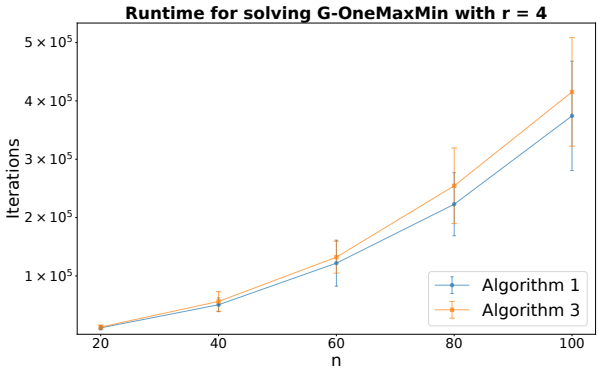

Figure 1: The mean (with standard deviations) number of iterations of Algorithms 1 and 3 for solving G-OneMaxMin with $n \in \{20, 40, 60, 80, 100\}$ and $r = 4$ in 100 independent runs.

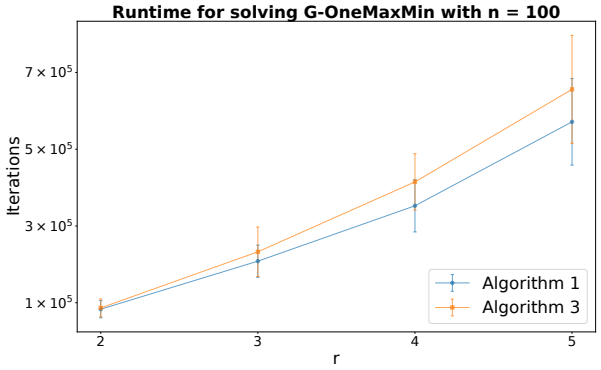

Figure 2: The mean (with standard deviations) number of iterations of Algorithms 1 and 3 for solving G-OneMaxMin with $n = 100$ and $r \in [2..5]$ in 100 independent runs.

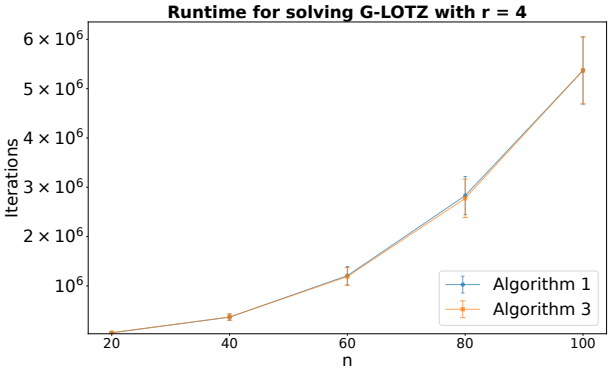

Figure 3: The mean (with standard deviations) number of iterations of Algorithms 1 and 3 for solving G-LOTZ with $n \in \{20, 40, 60, 80, 100\}$ and $r = 4$ in 100 independent runs.

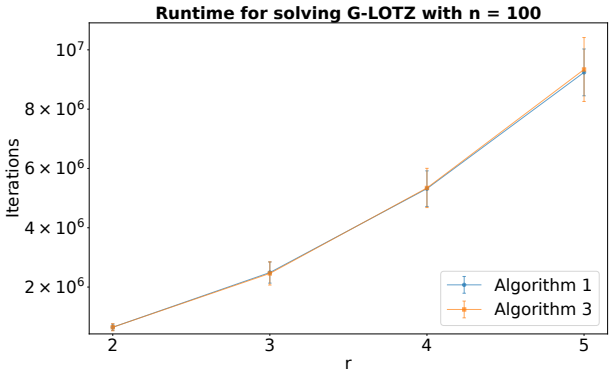

Figure 4: The mean (with standard deviations) number of iterations of Algorithms 1 and 3 for solving G-LOTZ with $n = 100$ and $r \in [2..5]$ in 100 independent runs.

6 Conclusion

This paper conducted the first mathematical runtime analyses of MOEAs for multi-valued decision variables. We extended the classic OneMinMax problem and the SEMO multi-objective algorithm from the binary setting to finite multi-valued domains. We proved an upper bound of $O(n^2r^2 \log n)$ on the expected number of iterations required to compute the entire Pareto front. We also proved a near-tight lower bound of $\Omega(n^2r(r+\log n))$. Both agree with the known $\Theta(n^2 \log n)$ runtime for the binary case.

These results suggest that existing MOEAs, when equipped with natural extensions of the mutation operator to multi-valued domains, perform well on r-valued problems. The moderate increase in runtime as r grows is to be expected, among others, because the size of the Pareto front of our optimization problem grows linearly with r.

In contrast, the current mathematical analysis techniques appear insufficient to completely understand such multi-valued optimization processes. At present, we can determine the true asymptotic order of magnitude of the runtime only for the variant of the SEMO that accepts only strictly dominating solutions. Here we prove a runtime of $\Theta(n^2r(r + \log n))$, showing a linear increase of the runtime with r when r is at most logarithmic in n. Both from our understanding of the two algorithms and our experiments, we conjecture that both algorithms have the same asymptotic runtime on the G-OneMaxMin benchmarks, which then would be $\Theta(n^2r(r + \log n))$. Proving such a result formally, however, appears to be challenging and might need considerably new analysis methods.

Acknowledgments

This work was supported by Guangdong Basic and Applied Basic Research Foundation (Grant No. 2025A1515011936), National Natural Science Foundation of China (Grant No. 62306086), and Xinjiang Tianshan Innovative Research Team (2025D14009). This research benefited from the support of the FMJH Program PGMO.

References

[Adak and Witt, 2024] Sumit Adak and Carsten Witt. Runtime analysis of a multi-valued compact genetic algorithm on generalized OneMax. In *Parallel Problem Solving from Nature, PPSN 2024, Proceedings, Part III*, pages 53–69, 2024.

[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*, pages 8833–8841. ijcai.org, 2025.

[Bäck, 1993] Thomas Bäck. Optimal mutation rates in genetic search. In *International Conference on Genetic Algorithms, ICGA 1993*, pages 2–8. Morgan Kaufmann, 1993.

[Ben Jedidia *et al.*, 2024] Firas Ben Jedidia, Benjamin Doerr, and Martin S. Krejca. Estimation-of-distribution algorithms for multi-valued decision variables. *Theoretical Computer Science*, 1003:114622, 2024.

[Bendahi *et al.*, 2025] Abderrahim Bendahi, Benjamin Doerr, Adrien Fradin, and Johannes F. Lutzeyer. Speeding up hyper-heuristics with Markov-chain operator selection and the only-worsening acceptance operator. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*, pages 8850–8857. ijcai.org, 2025.

[Bian *et al.*, 2024] Chao Bian, Shengjie Ren, Miqing Li, and Chao Qian. An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2024*, pages 6905–6913. ijcai.org, 2024.

[Bian *et al.*, 2025] Chao Bian, Yawen Zhou, Miqing Li, and Chao Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. *Artificial Intelligence*, 341:104308, 2025.

[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.

[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 *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*, pages 8867–8875. ijcai.org, 2025.

[Doerr and Neumann, 2020] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020.

[Doerr and Pohl, 2012] Benjamin Doerr and Sebastian Pohl. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 1317–1324. ACM, 2012.

[Doerr and Qu, 2023a] 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 Qu, 2023b] Benjamin Doerr and Zhongdi Qu. Runtime analysis for the NSGA-II: provable speed-ups from crossover. In *Conference on Artificial Intelligence, AAAI 2023*, pages 12399–12407. 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.*, 2011] Benjamin Doerr, Daniel Johannsen, and Martin Schmidt. Runtime analysis of the (1+1) evolutionary algorithm on strings over finite alphabets. In *Foundations of Genetic Algorithms, FOGA 2011*, pages 119–126. ACM, 2011.

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

[Doerr *et al.*, 2018] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica*, 80:1732–1768, 2018.

[Doerr *et al.*, 2025a] Benjamin Doerr, Martin S. Krejca, and Andre Opris. Tight runtime guarantees from understanding the population dynamics of the GSEMO multi-objective evolutionary algorithm. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*, pages 8876–8884. ijcai.org, 2025.

[Doerr *et al.*, 2025b] Benjamin Doerr, Martin S. Krejca, and Günter Rudolph. Runtime analysis for multi-objective evolutionary algorithms in unbounded integer spaces. In *Conference on Artificial Intelligence, AAAI 2025*, pages 26955–26963. AAAI Press, 2025.

[Harder *et al.*, 2024] Jonathan Gadea Harder, Timo Kötzing, Xiaoyue Li, Aishwarya Radhakrishnan, and Janosch Ruff. Run time bounds for integer-valued OneMax functions. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, pages 1569–1577. ACM, 2024.

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

[Horoba and Neumann, 2008] Christian Horoba and Frank Neumann. Benefits and drawbacks for the use of epsilon-dominance in evolutionary multi-objective optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2008*, pages 641–648. ACM, 2008.

[Kötzing *et al.*, 2015] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. (1+1) EA on generalized dynamic OneMax. In *Foundations of Genetic Algorithms, FOGA 2015*, pages 40–51. ACM, 2015.

[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.

[Li *et al.*, 2016] Yuan-Long Li, Yu-Ren Zhou, Zhi-Hui Zhan, and Jun Zhang. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20:563–576, 2016.

[Li *et al.*, 2025a] Mingfeng Li, Qiang Zhang, Weijie Zheng, and Benjamin Doerr. Why popular MOEAs are popular: Proven advantages in approximating the Pareto front. In *Advances in Neural Information Processing Systems, NeurIPS 2025*, 2025. To appear.

[Li *et al.*, 2025b] 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*, pages 8885–8893. ijcai.org, 2025.

[Li *et al.*, 2026] Mingfeng Li, Cheng Zheng, Weijie Zheng, and Banjamin Doerr. First mathematical runtime analyses of multi-objective evolutionary algorithms for multi-valued decision variables. In *International Joint Conference on Artificial Intelligence, IJCAI 2026*. ijcai.org, 2026. To appear. See full version in https://arxiv.org/abs/2605.14836.

[Mühlenbein, 1992] Heinz Mühlenbein. How genetic algorithms really work: mutation and hillclimbing. In *Parallel Problem Solving from Nature, PPSN 1992*, pages 15–26. Elsevier, 1992.

[Opris, 2026a] Andre Opris. Many-objective problems where crossover is provably essential. *Artificial Intelligence*, 350:104453, 2026.

[Opris, 2026b] Andre Opris. Towards a rigorous understanding of the population dynamics of the NSGA-III: Tight runtime bounds. In *Conference on Artificial Intelligence, AAAI 2026*, pages 37125–37133. AAAI Press, 2026.

[Qian *et al.*, 2018a] Chao Qian, Jing-Cheng Shi, Ke Tang, and Zhi-Hua Zhou. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22:595–608, 2018.

[Qian *et al.*, 2018b] 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.

[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.

[Ren *et al.*, 2025] Shengjie Ren, Zimin Liang, Miqing Li, and Chao Qian. Stochastic population update provably needs an archive in evolutionary multi-objective optimization. In *International Joint Conference on Artificial Intelligence, IJCAI 2025*, pages 8921–8929. ijcai.org, 2025.

[Rudolph, 2023] Günter Rudolph. Runtime analysis of (1+1)-EA on a biobjective test function in unbounded integer search space. In *IEEE Symposium Series on Computational Intelligence, SSCI 2023*, pages 1380–1385. IEEE, 2023.

[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.

[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, 2023] Weijie Zheng and Benjamin Doerr. Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). *Artificial Intelligence*, 325:104016, 2023.

[Zheng and Doerr, 2024] Weijie Zheng and Benjamin Doerr. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*, pages 20874–20882. AAAI Press, 2024.

[Zheng and Doerr, 2025] Weijie Zheng and Benjamin Doerr. Approximation guarantees for the non-dominated sorting genetic algorithm II (NSGA-II). *IEEE Transactions on Evolutionary Computation*, 29:891–905, 2025.

[Zheng *et al.*, 2024] Weijie Zheng, Mingfeng Li, Renzhong Deng, and Benjamin Doerr. How to use the Metropolis algorithm for multi-objective optimization? In *Conference on Artificial Intelligence, AAAI 2024*, pages 20883–20891. AAAI Press, 2024.

[Zheng, 2026] Weijie Zheng. Proven advantage of multiobjective evolutionary algorithms for problems with different degrees of conflict. *Artificial Intelligence*, 2026. To appear. Available at https://arxiv.org/abs/2408.04207.

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