Runtime Analysis for Multi-Objective Evolutionary Algorithms in Unbounded Integer Spaces

Benjamin Doerr[1], **Martin S. Krejca**[1], **Günter Rudolph**[2]

[1]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[2]Department of Computer Science, TU Dortmund University, Dortmund, Germany
{first-name.last-name}@polytechnique.edu, guenter.rudolph@tu-dortmund.de

Abstract

Randomized search heuristics have been applied successfully to a plethora of problems. This success is complemented by a large body of theoretical results. Unfortunately, the vast majority of these results regard problems with binary or continuous decision variables – the theoretical analysis of randomized search heuristics for unbounded integer domains is almost nonexistent. To resolve this shortcoming, we start the runtime analysis of multi-objective evolutionary algorithms, which are among the most successful randomized search heuristics, for unbounded integer search spaces. We analyze single- and full-dimensional mutation operators with three different mutation strengths, namely changes by plus/minus one (unit strength), random changes following a law with exponential tails, and random changes following a power-law. The performance guarantees we prove on a recently proposed natural benchmark problem suggest that unit mutation strengths can be slow when the initial solutions are far from the Pareto front. When setting the expected change right (depending on the benchmark parameter and the distance of the initial solutions), the mutation strength with exponential tails yields the best runtime guarantees in our results – however, with a wrong choice of this expectation, the performance guarantees quickly become highly uninteresting. With power-law mutation, which is an essentially parameter-less mutation operator, we obtain good results uniformly over all problem parameters and starting points. We complement our mathematical findings with experimental results that suggest that our bounds are not always tight. Most prominently, our experiments indicate that power-law mutation outperforms the one with exponential tails even when the latter uses a near-optimal parametrization. Hence, we suggest to favor power-law mutation for unknown problems in integer spaces.

Introduction

For more than thirty years, the mathematical runtime analysis of randomized search heuristics has supported the design and analysis of these important algorithms, both in single- and in multi-objective optimization (Neumann and Witt 2010; Auger and Doerr 2011; Jansen 2013; Zhou, Yu, and Qian 2019; Doerr and Neumann 2020). While in practice heuristics are successfully employed for all types of decision variables, the mathematical analysis is mostly concentrated on binary or continuous variables. Discrete spaces with more than two variable values are considered far more infrequently. Theoretical works include runtime analyses for evolutionary algorithms, ant-colony optimizers, and estimation-of-distribution algorithms on categorical variables, e.g., Scharnow, Tinnefeld, and Wegener (2004); Baswana et al. (2009); Sudholt and Thyssen (2012); Doerr, Happ, and Klein (2012); Ben Jedidia, Doerr, and Krejca (2024); Adak and Witt (2024). The theoretical research for cardinal variables was started by Doerr, Johannsen, and Schmidt (2011); Doerr and Pohl (2012); Kötzing, Lissovoi, and Witt (2015) with analyses how the $(1+1)$ EA optimizes multi-valued linear functions. Doerr, Doerr, and Kötzing (2018, 2019) showed that for multi-valued variables larger mutation rates can be advantageous. Submodular functions with multi-valued discrete domain were studied by Qian et al. (2018a,b). We are only aware of two analyses for search spaces consisting of unbounded integer variables. The first is (Rudolph 2023), which is a single-objective analysis of subproblems of a multi-objective problem. The second is (Harder et al. 2024), which considers a single-objective problem and shows the benefit of larger mutation rates. We are not aware of any true multi-objective works with multi-valued variables, despite considerable recent theoretical research on multi-objective heuristics (Zheng, Liu, and Doerr 2022; Dang et al. 2023; Cerf et al. 2023; Do et al. 2023; Dinot et al. 2023; Zheng and Doerr 2024; Zheng et al. 2024; Bian et al. 2024; Ren et al. 2024).

Since multi-objective optimization is an area where heuristics, in particular, evolutionary algorithms, are intensively used and with great success (see, e.g., the famous NSGA-II algorithm (Deb et al. 2002) with more than $50\,000$ citations on Google Scholar), we start in this work the mathematical runtime analysis for truly multi-objective optimization problems in unbounded integer search spaces. For the benchmark problems proposed by Rudolph (2023), we analyze the performance of the *simple evolutionary multi-objective optimizer (SEMO)* (Laumanns et al. 2002) and the *Global SEMO (GSEMO)* (Giel 2003), the two most prominent evolutionary algorithms in the runtime analysis of multi-objective evolutionary algorithms. Our objective is to understand, via theoretical means, what are suitable mutation operators for unbounded integer domains. We propose three natural operators, changing variables (i) by plus or minus one (unit muta-

tion strength), (ii) by a random value drawn from a symmetric distribution with exponential tails, and (iii) by a random value drawn from a symmetric power-law distribution.

We conduct an extensive mathematical runtime analysis for these two algorithms with the three mutation operators (with general parameters, where applicable) on the benchmark with general width parameter a of the Pareto front for a general initial solution $x^{(0)}$. Our results, presented in more detail later in this work when all ingredients are made precise, indicate the following. All algorithm variants can compute the full Pareto front of our benchmark problem in reasonable time. The unit mutation strength, not surprisingly can lead to a slow progress towards the Pareto front when the initial solution is far, but it also results in a slow exploration of the Pareto front after having reached it. The performance with the exponential-tail mutation strength depends heavily on the variance parameter q (which is essentially the reciprocal of the expected absolute change). With an optimal choice of q, depending on the benchmark parameter a and the starting solution $x^{(0)}$, this operator leads to the best performance guarantee among our results. However, our runtime guarantees strongly depend on the relation of q, a, and $x^{(0)}$; hence a suboptimal choice of q can quickly lead to very weak performance guarantees. The power-law mutation strength is a good one-size-fits-all solution. Being an essentially parameter-less operator, it achieves a good performance uniformly over all values of a and $x^{(0)}$, clearly beating the unit mutation strength for larger instances or distances of the starting solution from the Pareto front.

We complement our theoretical results by an empirical analysis, aimed to understand how tight our guarantees are. We observe that the best-possible parameter regime for the exponential-tail mutation is within the range of our mathematical findings. However, surprisingly, the exponential-tail mutation is not able to outperform the power-law mutation, even with a near-optimal parametrization of the former. This suggests that our bounds for the power-law mutation are not tight. In fact, our experiments indicate that the actual expected runtime for this operator in the considered setting is linear in the size of the Pareto front, whereas our guarantees bound it by a polynomial with an exponent between 1 and 2, depending on the parametrization of the operator. We speculate that the actual linear runtime is a result of the complex population dynamics, which, to the best of our knowledge, have not been studied in the detail necessary for an improvement in any theoretical study of the (G)SEMO algorithms.

Overall, our work shows that standard heuristics with appropriate mutations can deal well with certain multi-objective problems with unbounded integer decision variables. We strongly suggest the parameter-less power-law operator, as it has shown the best performance empirically and also has a uniformly good performance guarantee in a wide range of situations in our theoretical findings.

Our proofs are in the full version Doerr et al. (2024a).

Preliminaries

The natural numbers $\mathbb{N}$ include 0. For $a, b \in \mathbb{R}$, let $[a..b] = [a, b] \cap \mathbb{N}$, define $[a] := [1..a]$, and let $\mathbb{N}_{\geq a} = [a, \infty) \cap \mathbb{N}$.

Algorithm 1: Algorithmic framework for evolutionary multi-objective minimization of a given d-objective function $f\colon \mathbb{Z}^n \to \mathbb{R}^d$. The framework requires an initial individual $x^{(0)} \in \mathbb{Z}^n$ and a mutation operator "mutation". If this operator modifies exactly one position, the resulting algorithm is the SEMO. If it modifies each position randomly and independently, the resulting algorithm is the GSEMO.

$P^{(0)} = \{x^{(0)}\}$;
$t \leftarrow 0$;
while *termination criterion not met* **do**
  choose $x^{(t)}$ from $P^{(t)}$ uniformly at random;
  $y^{(t)} \leftarrow \mathrm{mutation}(x^{(t)})$;
  $Q^{(t)} \leftarrow P^{(t)} \setminus \{z \in P^{(t)} \colon f(y^{(t)}) \preceq f(z)\}$;
  if $\nexists z \in Q^{(t)} \colon f(z) \prec f(y^{(t)})$ **then**
    $P^{(t+1)} \leftarrow Q^{(t)} \cup \{y^{(t)}\}$;
  else $P^{(t+1)} \leftarrow Q^{(t)}$;
  $t \leftarrow t + 1$;

Given $n, d \in \mathbb{N}_{\geq 2}$, we call $f\colon \mathbb{Z}^n \to \mathbb{R}^d$ a *d-objective function*, which we aim to minimize. We call a point $x \in \mathbb{Z}^n$ an *individual*, and $f(x)$ the *objective value of* x. For $i \in [n]$ and $j \in [d]$, let x_i denote the i-th component of x, and let $f_j(x)$ denote the j-th component of $f(x)$.

The objective values of a d-objective function f follow a weak partial order, denoted by $\preceq$. For all $u, v \in \mathbb{R}^d$, we say that u *weakly dominates* v ($u \preceq v$) if and only if for all $i \in [d]$, we have $u_i \leq v_i$. We say that u *strictly* dominates v ($u \prec v$) if and only if at least one of these inequalities is strict. We extend this notation to individuals, where a dominance holds if and only if it holds for the respective objective values.

We consider the minimization of f, that is, we are interested in $\preceq$-minimal images. The set of all objective values that are not strictly dominated, that is, the set $F^* := \{f(y) \in \mathbb{R}^d \mid y \in \mathbb{Z}^n \wedge \nexists x \in \mathbb{Z}^n \colon f(x) \prec f(y)\}$, is the *Pareto front of* f. Furthermore, the individuals that are mapped to the Pareto front, that is, the set $f^{-1}(F^*)$, is the *Pareto set of* f.

Algorithmic Framework

We consider the framework of evolutionary multi-objective minimization (Algorithm 1), aimed at finding the Pareto front of a given d-objective function. The algorithm acts iteratively and maintains a multi-set of individuals (the *population*) that are not strictly dominated by any of the so-far found individuals. In each iteration, the algorithm creates a new individual y from a random individual from the population by applying an operation known as *mutation*. Afterward, all individuals weakly dominated by y are removed from the population, and y is added if it is not strictly dominated by any of the remaining individuals in the population.

Mutation. We consider *single-* and *full-dimensional* mutation. Either acts on a *parent* $x \in \mathbb{Z}^n$, requires a distribution D on $\mathbb{Z}$ (the *mutation strength*; see also *Runtime Analysis*), and returns an *offspring* $y \in \mathbb{Z}^n$.

Single-dimensional mutation chooses a single component $i \in [n]$ as well as an independent sample $Z \sim D$ and then sets $y_i = x_i + Z$. All other components remain unchanged, that is, for all $j \in [n] \setminus \{i\}$, it holds that $y_j = x_j$. Algorithm 1 with single-dimensional mutation results in the *SEMO* algorithm (Laumanns et al. 2002). Full-dimensional mutation does the following independently for each component $i \in [n]$ of x: With probability $1/n$, draw an independent sample $Z_i \sim D$, and set $y_i = x_i + Z_i$. With the remaining probability, set $y_i = x_i$. Hence, in expectation, exactly one component of x is changed, while any number of components may be changed. Algorithm 1 with full-dimensional mutation results in the *global SEMO* (GSEMO) (Giel 2003).

Runtime. The *runtime* of Algorithm 1 is the (random) number of function evaluations until the objective values of the population contain the Pareto front. We do not re-evaluate individuals, but equal individuals are separately evaluated. That is, the initial individual is evaluated once, and the algorithm evaluates exactly one solution (namely $y^{(t)}$) each iteration. Hence, the runtime is 1 plus the number of iterations until the Pareto front is covered.

Benchmark Problem

We consider the following bi-objective benchmark function, introduced by Rudolph (2023). Given a parameter $a \in \mathbb{N}$, the function $f\colon \mathbb{Z}^n \to \mathbb{N}^2$ is defined as

$$x \mapsto \begin{pmatrix} |x_1 - a| + \sum_{i \in [2..n]} |x_i| \\ |x_1 + a| + \sum_{i \in [2..n]} |x_i| \end{pmatrix}.$$

This function aims at minimizing the distance to two target points, which is the same idea present in similar benchmarks (ONEMAX and ONEMINMAX (Giel and Lehre 2010)) that are used as initial problems for related settings.

The Pareto set and front of f satisfy (Rudolph 2023)

$$X^* = \{(k, 0, \dots, 0)^\top \in \mathbb{Z}^n \mid k \in [-a, a] \cap \mathbb{Z}\} \text{ and}$$
$$F^* = \{(k, 2a - k)^\top \in \mathbb{N}^2 \mid k \in [0..2a]\}.$$

We note that $|F^*| = 2a+1$, as this value plays a crucial role in our analyses *(Runtime Analysis)*.

Useful Properties of the Benchmark Problem We study useful properties of f, partially in the context of Algorithm 1. Throughout this section, we assume that $a \in \mathbb{N}$ and that $n \in \mathbb{N}_{\geq 2}$. When we consider an instance of Algorithm 1, we allow for *arbitrary* mutation.

Lemma 1 shows when a solution whose first coordinate is not in $(-a, a)$ is comparable (w.r.t. f) to any other point.

Lemma 1. *Let $x \in \mathbb{Z}^n$ and $y \in \mathbb{Z}_{\geq a} \times \mathbb{Z}^{n-1}$ such that $f_1(x) \leq f_1(y)$. Then $f(x) \preceq f(y)$.*

Similarly, let $x \in \mathbb{Z}^n$ and $y \in \mathbb{Z}_{\leq -a} \times \mathbb{Z}^{n-1}$ such that $f_2(x) \leq f_2(y)$. Then $f(x) \preceq f(y)$.

Lemma 1 implies that the population has at most one solution with its first component at most $-a$ and at most one with its first component at least a, formalized below.

Lemma 2. *Let $t \in \mathbb{N}$. Then $|P^{(t)} \cap (\mathbb{Z}_{\geq a} \times \mathbb{Z}^{n-1})| \leq 1$ and $|P^{(t)} \cap (\mathbb{Z}_{\leq -a} \times \mathbb{Z}^{n-1})| \leq 1$.*

Lemma 3 shows that the algorithm contains at most one solution per x_1-value in $[-a..a]$.

Lemma 3. *Let $t \in \mathbb{N}$, and let $i \in [-a..a]$. Then $|P^{(t)} \cap (\{i\} \times \mathbb{Z}^{n-1})| \leq 1$.*

The bounds on the population size from Lemmas 2 and 3 imply the following bound on the overall population size.

Lemma 4. *Let $t \in \mathbb{N}$. Then $|P^{(t)}| \leq 2a + 1$.*

Lemma 5 shows that the dominance of two points implies an order with respect to the L1-norm of the two points.

Lemma 5. *Let $x, y \in \mathbb{Z}^n$. If $f(x) \preceq f(y)$, then $\|x\|_1 \leq \|y\|_1$.*

Lemma 5 implies that the minimum L1-norm of the population of the algorithm cannot increase over time. The following lemma formalizes this property. It is the main driving force of our theoretical analyses in *Runtime Analysis*.

Lemma 6. *Let $t \in \mathbb{N}$, and let $z^* \in P^{(t)}$ be such that $\|z^*\|_1 = \min_{z \in P^{(t)}} \|z\|_1$. Moreover, assume for the offspring that $\|y^{(t)}\|_1 > \|z^*\|_1$. Then $z^* \in P^{(t+1)}$.*

Optimization Dynamics The lemmas above show that the dynamics of Algorithm 1 on f are restricted to individuals with their first component in $[-a + 1..a - 1]$ as well as at most two individuals with their first component at most $-a$ or at least a, respectively. Since Lemma 5 shows that the L1-norm of such individuals cannot increase, their distance to the Pareto cannot increase either. Thus, if improvements occur, the population eventually covers the entire Pareto front.

Runtime Analysis

We analyze the expected runtime of Algorithm 1 instantiated as the SEMO and as the GSEMO *(Algorithmic Framework)* when optimizing function f *(Benchmark Problem)*. We assume for f implicitly that $a \in \mathbb{N}$ and that $n \in \mathbb{N}_{\geq 2}$.

We consider three different mutation strengths, each characterized by a law over $\mathbb{Z}$: the uniform law over $\{-1, 1\}$, a law with an exponential tail, and a power-law.

Although the mutation strength greatly affects the expected runtime of the algorithm, our analyses follow the same general outline. Each analysis is split into two phases. The first phase considers the time until the algorithm contains the all-0s vector, which is part of the Pareto front F^* of f. The second phase considers the remaining time until the objective values of the population contain F^*. The total runtime bound is then the sum of the bounds of both phases.

For the first phase, we use that the minimum L1-distance of the population to the all-0s vector never increases (Lemma 6). Formally, we define a potential function that measures this distance, and we utilize tools introduced below for deriving the expected runtime for this phase.

Throughout our analyses, we use that the population consists by Lemma 4 of at most $2a + 1$ individuals. This results in a probability of at least $1/(2a + 1)$ for choosing a specific individual for mutation and, thus, in an expected time of $2a + 1$ for making such a choice. In addition, the probability to change exactly one component of a solution is for both algorithms in the order of $1/n$. Combining this with the

choice for a specific individual yields an overall waiting time of about $(2a+1)n$, which all of our results have in common.

The speed of each phase is heavily determined by the mutation strength, leading to different analyses.

Mathematical Tools

Variable drift theorems translate information about the expected progress of a random process into bounds on expected hitting times. This concept was independently developed by Mitavskiy et al. (2009) and Johannsen (2010). The following variant is from Doerr et al. (2020, Theorem 6).

Theorem 7 (Discrete variable drift, upper bound)**.** *Let $(X_t)_{t\geq 0}$ be a sequence of random variables in $[0..n]$, and let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists a monotonically increasing function $h\colon [n] \to \mathbb{R}_0^+$ such that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq h(X_t)$ holds for all $t < T$. Then*

$$\mathrm{E}[T \mid X_0] \leq \textstyle\sum_{i\in[X_0]} \frac{1}{h(i)}.$$

Additive drift is a simplification of variable drift to the case that the expected progress of a random process is bounded by a constant value. It dates back to a theorem by He and Yao (He and Yao 2004). The following simplified version is from Kötzing and Krejca (2019, Theorem 7).

Theorem 8 (Additive drift, unbounded search space)**.** *Let $(X_t)_{t\in\mathbb{N}}$ be random variables over $\mathbb{R}_{\geq 0}$, and let $T = \inf\{t \in \mathbb{N} \mid X_t = 0\}$. Furthermore, suppose that there is some value $\delta \in \mathbb{R}_{>0}$ such that for all $t < T$ holds that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq \delta$. Then $\mathrm{E}[T \mid X_0] \leq \frac{X_0}{\delta}$.*

Unit-Step Mutation

Unit-step mutation uses the uniform law over $\{-1, 1\}$. When modifying the value $x_i \in \mathbb{Z}$ of an individual x, then x_i is increased by 1 with probability $\frac{1}{2}$ and else decreased by 1.

Our main result for unit-step mutation is Theorem 9, showing that the expected runtime of the SEMO and the GSEMO scales linearly in the L1-norm of the initial solution $x^{(0)}$ (plus $2a$), and linearly in the dimension n and the radius a of the Pareto front of f. The linear scaling in an is due to waiting to make progress, as discussed at the start of *Runtime Analysis*. The linear scaling in $\|x^{(0)}\|_1$ is due to the first phase, as the mutation changes a component by only 1 and needs to cover a distance of $\|x^{(0)}\|_1$. Then, since the mutation changes a component only by 1, an additional time linear in a is required for covering the entire Pareto front.

Theorem 9. *Consider the SEMO or the GSEMO with unit-step mutation optimizing f, given $x^{(0)}$. Let $T = \inf\{t \in \mathbb{N} \mid F^* \subseteq f(P^{(t)})\}$. Then $\mathrm{E}[T \mid x^{(0)}] \leq 2n(2a+1)(\|x^{(0)}\|_1 + 2a)$ for the SEMO, and $\mathrm{E}[T \mid x^{(0)}] \leq 2en(2a+1)(\|x^{(0)}\|_1 + 2a)$ for the GSEMO.*

The following lemma bounds the expected time of the first phase. In each iteration, we have the same probability to improve the solution that is closest in L1-distance to the all-0s vector, resulting in a linear bound in the distance $\|x^{(0)}\|_1$.

Lemma 10. *Let $T_1 = \inf\{t \in \mathbb{N} \mid (0,\dots,0)^\top \in P^{(t)}\}$. Then $\mathrm{E}[T_1 \mid x^{(0)}] \leq 2n \cdot (2a+1) \cdot \|x^{(0)}\|_1$ for the SEMO, and $\mathrm{E}[T_1 \mid x^{(0)}] \leq 2en \cdot (2a+1) \cdot \|x^{(0)}\|_1$ for the GSEMO.*

The next lemma bounds the expected time of the second phase. Since the mutation only changes components by 1, starting from the all-0s vector, the Pareto front is covered in a linear fashion, expanding to either side. Hence, the resulting time is linear in the size of the Pareto front $(2a+1)$ as well as in the inverse of the probability to choose a specific individual and mutate it correctly (in the order of an).

Lemma 11. *Let $S \in \mathbb{N}$ be a (possibly random) iteration such that $P^{(S)}$ contains the individual $(0,\dots,0)^\top$, and let $T_2 = \inf\{t \in \mathbb{N} \mid t \geq S \wedge F^* \subseteq f(P^{(t)})\} - S$. Then $\mathrm{E}[T_2 \mid S, x^{(0)}] \leq 2n \cdot (2a+1) \cdot 2a$ for the SEMO, and $\mathrm{E}[T_2 \mid S, x^{(0)}] \leq 2en \cdot (2a+1) \cdot 2a$ for the GSEMO.*

Exponential-Tail Mutation

Exponential-tail mutation utilizes a symmetric law with exponential tails, parameterized by a value $q \in (0,1)$. For a random variable Z following this law, for all $k \in \mathbb{Z}$ holds

$$\Pr[Z = k] = \tfrac{q}{2-q}\,(1-q)^{|k|}. \tag{1}$$

We call this law a *bilateral geometric law* (Rudolph 1994).

Our main result for exponential-tail mutation is Theorem 12, which is clearly separated into the two phases of our analysis. Besides the common factor of an from making progress, as discussed at the beginning of this section, the expected runtime strongly depends on the mutation parameter q. We discuss the two terms in the following in detail.

Theorem 12. *Let $c \in (0,1)$ be constant. Consider the SEMO or the GSEMO with exponential-tail mutation with $q \in (0,c)$ optimizing f, given $x^{(0)}$. Let $T = \inf\{t \in \mathbb{N} \mid F^* \subseteq f(P^{(t)})\}$. Then there is a sufficiently large constant C such that for both the SEMO and the GSEMO holds that*

$$\mathrm{E}[T \mid x^{(0)}] = C\Big(an\Big(\frac{n}{q} + \|x^{(0)}\|_1 q + \max\Big\{\frac{\ln(a+1)}{aq}, aq + \ln(a+1)\Big\}\Big)\Big).$$

Our analysis is based on the following elementary result about the bilateral geometric law.

Lemma 13. *Let Z be a bilateral geometric random variable with parameter $q \in (0,1)$. Let $z \in \mathbb{N}$, and let Z_z be the random variable defined by $Z_z = Z$, if $0 \leq Z \leq z$, and $Z_z = 0$ otherwise. Then $E[Z_z] = \frac{1-q}{q(2-q)}(1-(1-q)^z(1+zq))$. Especially, for all constants $C \in (0,1)$, there is a constant $K \in \mathbb{R}_{>0}$ such that for all $q \leq C$ and all $z \in \mathbb{N}$, we have*

$$\mathrm{E}[Z_z] \geq K \min\{z^2 q, \tfrac{1}{4q}\}.$$

We utilize Lemma 13 in our proofs by estimating by how much a component of an individual chosen for mutation is improved. Since certain changes can be too large, we consider such progress to be 0. All other values are acceptable. Hence, we consider overall only a subset of values of the bilateral geometric law, which is well reflected in the lemma.

We now tend to the analysis of Theorem 12. The first phase is separated into two regimes, depending on how close the minimum L1-distance of the population is to the all-0s

vector. If this distance is at least in the order of n/q, the expected distance covered by a successful mutation is in the order of $1/q$, leading to the term $\|x^{(0)}\|_1 q$ (in addition to the factor of an from the waiting time for an improving mutation). Once the population gets closer than n/q to the all-0s vector, the progress is slowed down and essentially driven by unit changes, resulting in the term n/q.

Lemma 14. *Let $T_1 = \inf\{t \in \mathbb{N} \mid (0,\dots,0)^\top \in P\}$. Then*

$$\mathrm{E}[T_1 \mid x^{(0)}] \leq \frac{(2a+1)en}{K}\Big(\frac{n\pi^2}{6q} + 4\|x^{(0)}\|_1 q\Big),$$

where K is the constant from Lemma 13.

The expected runtime of the second phase depends on how $1/q$ compares to a. We split the runtime into two parts. The first part concerns covering a subset of the Pareto front that chooses individuals that are roughly $1/q$ apart, resulting in about aq intervals of roughly equal size. If $1/q > 2a+1$, then we consider the entire Pareto front as a single interval.

The second part concerns covering all intervals. Since uncovered points in each interval are at most apart by about $1/q$, we wait for such a rate to be chosen. This can happen for any interval, leading to independent trials. We then use a Chernoff-like concentration bound that provides us with a runtime bound that holds with high probability. Via a restart argument, this bound is turned into an expectation. The concentration bound yields the logarithmic factors.

Lemma 15. *Assume that at some time t_2, the population of the algorithm contains the solution $(0,\dots,0)$. Let $T_2 = \inf\{t \in \mathbb{N} \mid F^* \subseteq f(P^{(t_2+t)})\}$ the additional number of iterations until the Pareto front is computed. Then there is a sufficiently large constant $C \in \mathbb{R}_{>0}$ such that*

$$\mathrm{E}[T_2] = C\Big(an \max\Big\{\frac{\ln(a+1)}{aq}, aq + \ln(a+1)\Big\}\Big).$$

Power-Law Mutation

Power-law mutation utilizes a symmetric power-law, parameterized by the power-law exponent $\beta \in (1,2)$ and defined via the Riemann zeta function ζ. For a random variable Z following this law, it holds for all $k \in \mathbb{Z} \setminus \{0\}$ that

$$\Pr[Z = k] = |k|^{-\beta}/\big(2\,\zeta(\beta)\big).$$

This operator, introduced by (Doerr et al. 2017), was shown to be provably beneficial in various settings (Friedrich, Quinzan, and Wagner 2018; Corus, Oliveto, and Yazdani 2021; Antipov, Buzdalov, and Doerr 2022; Dang et al. 2022; Doerr and Qu 2023; Doerr and Rajabi 2023; Doerr, Krejca, and Vu 2024; Krejca and Witt 2024).

Our main result for power-law mutation is Theorem 16. Similar to the result for mutation strengths with exponential tails (Theorem 12), the expected runtimes of both phases are well separated. We explain the details for each phase below.

Theorem 16. *Consider the SEMO or the GSEMO with power-law mutation with constant $\beta \in (1,2)$ optimizing f, given $x^{(0)}$. Let $T = \inf\{t \in \mathbb{N} \mid F^* \subseteq f(P^{(t)})\}$. Then for both the SEMO and the GSEMO, it holds that*

$$\begin{aligned}
&\mathrm{E}[T \mid x^{(0)}] \\
&\leq (2a+1)\cdot 2en\zeta(\beta)\Big(2^{1/(2-\beta)} + 2\frac{2-\beta}{\beta-1}\|x^{(0)}\|_1^{\beta-1}\Big) \\
&\quad + 4\ln(2)en\frac{\zeta(\beta)(\beta-1)}{1-(3/2)^{1-\beta}}\frac{1}{\big(1-2^{1-\beta}\big)^2}(2a+1)^\beta \\
&\quad + (2a+1)\cdot 2en\zeta(\beta)\big(\ln(a+1)+1\big).
\end{aligned}$$

We make use of the following theorem, which estimates sums of monotone functions via a definite integral. This is useful, as our analyses involve many possible values for how to improve a specific individual. These cases lead to sums over of powers of $-\beta$, as the values follow a power-law. Integrating such a polynomial is easier than determining the exact value of the discrete sum. The theorem below shows that we make almost no error when considering the integral.

Theorem 17 ((Cormen et al. 2001, Inequality (A.12)))**.** *Let $g\colon \mathbb{R} \to \mathbb{R}$ be a monotonically non-increasing function, and let $\alpha, \beta \in \mathbb{R}$ with $\alpha \leq \beta$. Then*

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

We now consider Theorem 16. The expected runtime of the first phase, besides the factor an, is of order $\|x^{(0)}\|_1^{\beta-1}$. Ignoring the factor an, this is because the algorithm makes an improvement of size k with probability $k^{-\beta}$ and has up to about $\|x^{(t)}\|_1$ choices for an improvement. This leads to an expected improvement of $k^{1-\beta}$ per component of $\|x^{(t)}\|_1$. Integrating this expression results in an overall expected improvement of order $\|x^{(t)}\|_1^{2-\beta}$. By the variable drift theorem (Theorem 7), estimating the sum via an integral, this translates to an overall runtime of order $\|x^{(0)}\|_1^{\beta-1}$.

Lemma 18. *Let $T_1 = \inf\{t \in \mathbb{N} \mid (0,\dots,0)^\top \in P^{(t)}\}$. Then*

$$\begin{aligned}
&\mathrm{E}[T_1 \mid x^{(0)}] \\
&\leq (2a+1)\cdot 2en\zeta(\beta)\Big(2^{1/(2-\beta)} + 2\tfrac{2-\beta}{\beta-1}\|x^{(0)}\|_1^{\beta-1}\Big).
\end{aligned}$$

The second phase advances in several steps. In each step, the number of points covered on the Pareto front of f roughly doubles and is spread evenly across the Pareto front. If the maximum distance of consecutively uncovered points is ℓ, the probability to create a new point is at least $\ell^{1-\beta}$. At the beginning, since we assume that we only have a single point on the Pareto front, ℓ is in the order of a, and the probability to choose the correct point and mutate it correctly is in the order $1/(an)$. Hence, the waiting time for the first step is in the order of na^β, which dominates the remaining time, as the each additional point on the Pareto front increases the chance of creating a new one. Since the length of the Pareto front is not a power of 2, our result below has two terms. The first one bounds the time that the doubling procedure above covers at least half of the Pareto front. The second term bounds the time to cover the remaining points.

	$1/q$	1st hit	cover	total
U		510 006± 25	342 916±44	852 922±11
E	5	73 034± 8	23 115±31	96 148±10
	10	25 288± 9	18 346±25	43 634±11
	20	9 028± 8	15 050±22	24 078±14
	50	2 810± 11	15 237±18	18 048±16
	100	1 604± 34	18 401±24	20 004±23
	200	1 613± 63	24 295±20	25 908±20
	500	3 544±104	43 693±20	47 236±23
P		1 301± 47	14 263±16	15 565±15

Table 1: Means and standard deviations in percent of 1st hitting time (phase 1), Pareto set cover time (phase 2), and total runtime (# evaluations of f) for scenario 1 for the GSEMO optimizing f with the mutation operators: unit-step (U), exponential-tail (E), and power-law (P). Column $1/q$ refers to the *step size* of parameter q of E. For P, we chose $\beta = \frac{3}{2}$. The runs were started with $a = 200$ and $x^{(0)} = (0, 100a)$, with 50 independent runs per row for $n = 2$.

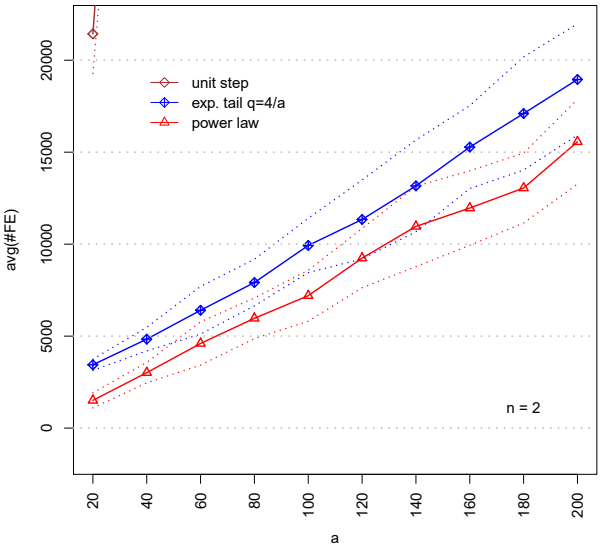

Figure 1: The results of scenario 2. Average evaluations of f for varying a for the GSEMO optimizing f with the mutation operators: unit-step (diamonds), exponential-tail (cross diamonds) with $\frac{1}{q} = \frac{a}{4}$, and power-law (triangles) with $\beta = \frac{3}{2}$. Each point is based on 50 independent runs, with $x^{(0)} = (0, 100a)$. The dotted lines depict the std. deviations.

Lemma 19. *Let $S \in \mathbb{N}$ be a (possibly random) iteration such that $P^{(S)}$ contains the individual $(0, \dots, 0)^\top$, and let $T_2 = \inf\{t \in \mathbb{N} \mid t \geq S \wedge F^* \subseteq f(P^{(t)})\} - S$. Then*

$$\begin{aligned}&\mathrm{E}[T_2 \mid S, x^{(0)}]\\ &\quad\leq 4\ln(2)en\frac{\zeta(\beta)(\beta-1)}{1-(3/2)^{1-\beta}}\frac{1}{(1-2^{1-\beta})^2}(2a+1)^\beta\\ &\qquad+(2a+1)\cdot 2en\zeta(\beta)\big(\ln(a+1)+1\big).\end{aligned}$$

Empirical Analysis

We aim to assess how far our theoretical upper bounds are from actual, empirical values. We also aim to understand whether the exponential-tail law actually has a parametrization that is favorable to the power-law mutation, as suggested by our theoretical results. To this end, we first discuss what runtime behavior we expect, based on our theoretical bounds. Then we explain our experimental setup and discuss and present our findings. For the sake of simplicity, we only consider the GSEMO here, as it is the more general algorithm, although benchmark f is simple enough such that the SEMO would also be sufficient. We note that when optimizing f, the expected runtime of the GESMO should be worse by a factor of at most e compared to the SEMO. Our code is publicly available (Doerr, Krejca, and Rudolph 2024b).

Similar to our discussion at the start of *Runtime Analysis*, we split the run of the GSEMO into two phases: The first phase counts the number of function evaluations until the population contains an individual on the Pareto front for the first time. The second phase counts the remaining number of function evaluations until the Pareto front is covered.

Theoretical considerations. In order to compare our theoretical results easily, we assume that the L1-norm of the initial point $x^{(0)}$ is in the order of the parameter a of benchmark f. Furthermore, we assume that the problem size n is constant, as we consider small values for n in our experiments, due to the search space being unbounded in any case. Then the expected runtime for unit-step mutation (Theorem 9) is in the order of a^2. For exponential-tail mutation with parameter q (Theorem 12), it is in the order of $a^2q + a\max\{\frac{\ln(a+1)}{aq}, aq + \ln(a+1)\}$. Last, for power-law mutation with power-law exponent β (Theorem 16), it is in the order of a^β. We see that choosing $q = \frac{1}{a}$ minimizes the maximum expression for the runtime of the exponential-tail mutation, resulting in a runtime bound in the order of $a\ln(a)$. For this setting, the exponential-tail mutation is fastest and has a quasi-linear runtime. It is followed by the runtime of power-law mutation, which is a polynomial with degree of β, which is between 1 and 2. Last, we have the unit-step mutation with a quadratic runtime.

Experimental setup. We aim to recreate a setting as described above, placing an emphasis on both phases though, as we do not know how tight our theoretical bounds actually are. To this end, we choose $n = 2$ and $x^{(0)} = (0, 100a) =: (0, y_0)$. Furthermore, we choose $\beta = \frac{3}{2}$, which is generally a good choice (Doerr et al. 2017). Our choice for n is based on all of our results holding for any value of $n \in \mathbb{N}_{\geq 2}$, and a smaller choice of n lets us run more experiments. Nonetheless, we note that we consider larger values of n in the full version, and the observations are qualitatively the same as they are for $n = 2$, just with an even clearer distinction between the different mutation operators.

We consider two different scenarios: The first scenario aims to determine a good value q for the exponential-tail mutation. To this end, we choose $a = 200$ as well as $\frac{1}{q} \in \{5, 10, 20, 50, 100, 200, 500\}$, which covers a broad, quickly increasing, thus diverse, range of values for $\frac{1}{q}$.

For each parameter combination of each scenario, we log the number of function evaluations of 50 independent runs, always for both phases, even for identical parameter values.

The second scenario observes the runtime behavior of all three mutation operators with respect to a, given a good value for q determined by scenario 1. We choose $y_0 = 100a$.

First scenario. Table 1 depicts our results. For unit step-mutation, we see that it is by far the worst operator for either phase. For exponential-tail mutation, we see that the choice of q has an apparently convex impact on the runtime for both phases, with the minimum taken over different values of q. This conforms mostly with our theoretical insights, where a too small value of $\frac{1}{q}$ takes needlessly long (like in the case of unit steps), but a too large value of $\frac{1}{q}$ requires too wait long for actually useful values to appear. While we expected the choice $\frac{1}{q} = a = 200$ to be best, it is actually $\frac{1}{q} = 50$. This is mostly due to our theoretical considerations ignoring constant factors and due to the term $\ln(a+1)$ in the runtime being multiplicative in one case and additive in the other.

More surprisingly, the mean runtime of the power-law mutation for either of the two phases is better than the mean of the exponential-tail distribution for any of our choices of $1/q$. This suggests that our runtime estimates are not tight.

Second scenario. Given the results from the first scenario, we fix $\frac{1}{q} = \frac{a}{4}$ here. We now range a from 20 to 200 in steps of 20. Our results are depicted in Figure 1.

We see that the runtime for exponential-tail mutation is essentially linear, which is what we expected. However, we also see that the runtime for power-law mutation is roughly linear, which is far better than our theoretical bounds.

One explanation for this discrepancy is that our theoretical analyses always assume that the algorithm's population is maximal, that is, it contains $2a+1$ individuals. This assumption seems to be too pessimistic, given preliminary empirical results. In phase 1, if a mutation performs a larger change to an individual, this new individual is likely to strictly dominate multiple solutions in the current population, thus reducing the population size. This potentially speeds up the first phase. In phase 2, for similar reasons, the initial population can be small and only grows large once almost the entire Pareto front is covered. Working with smaller populations in between can reduce the runtime of the second phase.

Conclusion

In this work, we initiated the runtime analysis of multi-objective evolutionary algorithms for unbounded integer spaces. To this end, we considered variants of the well-known SEMO and GSEMO algorithms. For each algorithm, we analyzed three different distributions of their mutation strengths. We derived runtime guarantees for a simple bi-objective problem with Pareto front of size $\Theta(a)$. Our theoretical results show a complex parameter landscape, depending on the different characteristics of the problem – namely, the problem size and a – and of the algorithm – namely, the mutation strength.

For all reasonable problem parameter choices, the unit-step mutation is the worst update strength, since the progress in each dimension is always bounded by the lowest possible value. The comparison of the other two mutation strengths is more delicate. Our theoretical results suggest that the exponential-tail mutation can outperform the power-law mutation if its parameter q is chosen carefully with respect to the problem parameters. However, in our experiments the power-law mutation always gave results superior to the exponential-tail mutation using a tuned parameter. Moreover, our experiments indicate a linear total runtime for both the exponential-tail and the power-law mutation (for certain parameter settings), which is a better runtime behavior for the power-law algorithm than what our upper bounds guarantee. We speculate this is a consequence of our pessimistic assumption of the algorithms' population size always being maximum. However, we note that, to the best of our knowledge, this is how essentially any theoretical consideration of the (G)SEMO up to date operates, for example (Bian, Qian, and Tang 2018; Doerr and Zheng 2021; Dang et al. 2023). Overall, our empirical analysis indicates that either our theoretical guarantees are not tight for the power-law algorithm or that the asymptotic effects are only witnessed for larger problem sizes, noting that our empirical observations seem to hold even more clearly for larger values of n. In any case, as both our theoretical and empirical results indicate that the power-law mutation has the best expected runtime for a wide range of problem parameters and starting points, and it does not require a careful parameter choice, our general recommendation is to prefer this algorithm for the optimization of problems with unbounded integer variables and no further problem-specific knowledge.

An interesting next step is to analyze whether our theoretical bounds are tight. We speculate that a more careful study of the algorithms' population dynamics is required. Theoretical analyses of this level of detail have not been conducted for the (G)SEMO so far. Hence, more refined analysis techniques than the state of the art seem to be necessary.

Another interesting direction is to prove lower bounds for our considered settings. Such analyses can shed more light onto certain behavioral aspects of the algorithms, and they require a deeper understanding of the dynamics of the population size. Thus, they are challenging to derive but can lead to insights that suggest how to improve our upper bounds.

In addition, it would be interesting to analyze other multi-objective evolutionary algorithms, e.g., the very prominent NSGA-II (Deb et al. 2002) as well as the NSGA-III (Deb and Jain 2014), SPEA2 (Zitzler, Laumanns, and Thiele 2001), and the SMS-EMOA (Beume, Naujoks, and Emmerich 2007). Moreover, these algorithms have a more complex procedure of updating their population in comparison to that of the (G)SEMO. This can further prove more challenging. However, a comparison to other multi-objective algorithms would greatly improve our current theoretical knowledge of the unbounded integer domain.

Acknowledgments

This research benefited from the support of the FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

References

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

Antipov, D.; Buzdalov, M.; and Doerr, B. 2022. Fast mutation in crossover-based algorithms. *Algorithmica*, 84: 1724–1761.

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

Baswana, S.; Biswas, S.; Doerr, B.; Friedrich, T.; Kurur, P. P.; and Neumann, F. 2009. Computing single source shortest paths using single-objective fitness. In *Foundations of Genetic Algorithms, FOGA 2009*, 59–66. ACM.

Ben Jedidia, F.; Doerr, B.; and Krejca, M. S. 2024. Estimation-of-distribution algorithms for multi-valued decision variables. *Theoretical Computer Science*, 1003: 114622.

Beume, N.; Naujoks, B.; and Emmerich, M. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181: 1653–1669.

Bian, C.; Qian, C.; and Tang, K. 2018. A general approach to running time analysis of multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2018*, 1405–1411. IJCAI.

Bian, C.; Ren, S.; Li, M.; and Qian, C. 2024. An archive can bring provable speed-ups in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2024*, 6905–6913. ijcai.org.

Cerf, S.; Doerr, B.; Hebras, B.; Kahane, J.; and Wietheger, S. 2023. 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*, 5522–5530. ijcai.org.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. The MIT Press, 2 edition.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2021. Fast immune system-inspired hypermutation operators for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 25: 956–970.

Dang, D.; Eremeev, A. V.; Lehre, P. K.; and Qin, X. 2022. Fast non-elitist evolutionary algorithms with power-law ranking selection. In *Genetic and Evolutionary Computation Conference, GECCO 2022*, 1372–1380. ACM.

Dang, D.-C.; Opris, A.; Salehi, B.; and Sudholt, D. 2023. A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In *Conference on Artificial Intelligence, AAAI 2023*, 12390–12398. AAAI Press.

Deb, K.; and Jain, H. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18: 577–601.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6: 182–197.

Dinot, M.; Doerr, B.; Hennebelle, U.; and Will, S. 2023. Runtime analyses of multi-objective evolutionary algorithms in the presence of noise. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5549–5557. ijcai.org.

Do, A. V.; Neumann, A.; Neumann, F.; and Sutton, A. M. 2023. Rigorous runtime analysis of MOEA/D for solving multi-objective minimum weight base problems. In *Advances in Neural Information Processing Systems, NeurIPS 2023*.

Doerr, B.; Doerr, C.; and Kötzing, T. 2018. Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica*, 80: 1732–1768.

Doerr, B.; Doerr, C.; and Kötzing, T. 2019. Solving problems with unknown solution length at almost no extra cost. *Algorithmica*, 81: 703–748.

Doerr, B.; Doerr, C.; and Yang, J. 2020. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science*, 801: 1–34.

Doerr, B.; Happ, E.; and Klein, C. 2012. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, 425: 17–33.

Doerr, B.; Johannsen, D.; and Schmidt, M. 2011. Runtime analysis of the (1+1) evolutionary algorithm on strings over finite alphabets. In *Foundations of Genetic Algorithms, FOGA 2011*, 119–126. ACM.

Doerr, B.; Krejca, M. S.; and Rudolph, G. 2024a. Runtime Analysis for Multi-Objective Evolutionary Algorithms in Unbounded Integer Spaces. *CoRR*, abs/arXiv:2412.11684.

Doerr, B.; Krejca, M. S.; and Rudolph, G. 2024b. Runtime Analysis for Multi-Objective Evolutionary Algorithms in Unbounded Integer Spaces (Code). Zenodo. https://zenodo.org/records/14506854.

Doerr, B.; Krejca, M. S.; and Vu, N. 2024. Superior genetic algorithms for the target set selection problem based on power-law parameter choices and simple greedy heuristics. In *Genetic and Evolutionary Computation Conference, GECCO 2024*. ACM.

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

Doerr, B.; and Neumann, F., eds. 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.

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

Doerr, B.; and Qu, Z. 2023. A first runtime analysis of the NSGA-II on a multimodal problem. *IEEE Transactions on Evolutionary Computation*, 27: 1288–1297.

Doerr, B.; and Rajabi, A. 2023. Stagnation detection meets fast mutation. *Theoretical Computer Science*, 946: 113670.

Doerr, B.; and Zheng, W. 2021. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, 12293–12301. AAAI Press.

Friedrich, T.; Quinzan, F.; and Wagner, M. 2018. Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, 293–300. ACM.

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

Giel, O.; and Lehre, P. K. 2010. On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation*, 18: 335–356.

Harder, J. G.; Kötzing, T.; Li, X.; Radhakrishnan, A.; and Ruff, J. 2024. Run Time Bounds for Integer-Valued OneMax Functions. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, 1569–1577. ACM.

He, J.; and Yao, X. 2004. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3: 21–35.

Jansen, T. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.

Johannsen, D. 2010. *Random Combinatorial Structures and Randomized Search Heuristics*. Ph.D. thesis, Universität des Saarlandes.

Kötzing, T.; and Krejca, M. S. 2019. First-hitting times under drift. *Theoretical Computer Science*, 796: 51–69.

Kötzing, T.; Lissovoi, A.; and Witt, C. 2015. (1+1) EA on Generalized Dynamic OneMax. In *Foundations of Genetic Algorithms, FOGA 2015*, 40–51. ACM.

Krejca, M. S.; and Witt, C. 2024. A flexible evolutionary algorithm with dynamic mutation rate archive. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, 1578–1586. ACM.

Laumanns, M.; Thiele, L.; Zitzler, E.; Welzl, E.; and Deb, K. 2002. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving from Nature, PPSN 2002*, 44–53. Springer.

Mitavskiy, B.; Rowe, J. E.; and Cannings, C. 2009. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *International Journal on Intelligent Computing and Cybernetics*, 2: 243–284.

Neumann, F.; and Witt, C. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.

Qian, C.; Shi, J.; Tang, K.; and Zhou, Z. 2018a. Constrained monotone k-submodular function maximization using multiobjective evolutionary algorithms with theoretical guarantee. *IEEE Transactions on Evolutionary Computation*, 22: 595–608.

Qian, C.; Zhang, Y.; Tang, K.; and Yao, X. 2018b. On multiset selection with size constraints. In *Conference on Artificial Intelligence, AAAI 2018*, 1395–1402. AAAI Press.

Ren, S.; Bian, C.; Li, M.; and Qian, C. 2024. A first running time analysis of the Strength Pareto Evolutionary Algorithm 2 (SPEA2). In *Parallel Problem Solving from Nature, PPSN 2024, Part III*, 295–312. Springer.

Rudolph, G. 1994. An evolutionary algorithm for integer programming. In *Parallel Problem Solving from Nature, PPSN 1994*, 139–148.

Rudolph, G. 2023. 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*, 1380–1385. IEEE.

Scharnow, J.; Tinnefeld, K.; and Wegener, I. 2004. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, 3: 349–366.

Sudholt, D.; and Thyssen, C. 2012. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica*, 64: 643–672.

Zheng, W.; and Doerr, B. 2024. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*, 20874–20882. AAAI Press.

Zheng, W.; Li, M.; Deng, R.; and Doerr, B. 2024. How to use the Metropolis algorithm for multi-objective optimization? In *Conference on Artificial Intelligence, AAAI 2024*, 20883–20891. AAAI Press.

Zheng, W.; Liu, Y.; and Doerr, B. 2022. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Conference on Artificial Intelligence, AAAI 2022*, 10408–10416. AAAI Press.

Zhou, Z.-H.; Yu, Y.; and Qian, C. 2019. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer.

Zitzler, E.; Laumanns, M.; and Thiele, L. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK report*, 103.