Proven Runtime Guarantees for How the MOEA/D Computes the Pareto Front From the Subproblem Solutions

Benjamin Doerr[1][0000−0002−9786−220X], Martin S. Krejca[1][0000−0002−1765−1219], and Noé Weeks[2][0009−0000−3008−1372]

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

Abstract. The decomposition-based multi-objective evolutionary algorithm (MOEA/D) does not directly optimize a given multi-objective function f, but instead optimizes $N + 1$ single-objective subproblems of f in a co-evolutionary manner. It maintains an archive of all non-dominated solutions found and outputs it as approximation to the Pareto front. Once the MOEA/D found all optima of the subproblems (the g-optima), it may still miss Pareto optima of f. The algorithm is then tasked to find the remaining Pareto optima directly by mutating the g-optima.

In this work, we analyze for the first time how the MOEA/D with only standard mutation operators computes the whole Pareto front of the OneMinMax benchmark when the g-optima are a strict subset of the Pareto front. For standard bit mutation, we prove an expected runtime of $O(nN \log n + n^{n/(2N)} N \log n)$ function evaluations. Especially for the second, more interesting phase when the algorithm start with all g-optima, we prove an $\Omega(n^{(1/2)(n/N+1)} \sqrt{N} 2^{-n/N})$ expected runtime. This runtime is super-polynomial if $N = o(n)$, since this leaves large gaps between the g-optima, which require costly mutations to cover.

For power-law mutation with exponent $\beta \in (1, 2)$, we prove an expected runtime of $O\big(nN \log n + n^\beta \log n\big)$ function evaluations. The $O\big(n^\beta \log n\big)$ term stems from the second phase of starting with all g-optima, and it is independent of the number of subproblems N. This leads to a huge speedup compared to the lower bound for standard bit mutation. In general, our overall bound for power-law suggests that the MOEA/D performs best for $N = O(n^{\beta-1})$, resulting in an $O(n^\beta \log n)$ bound. In contrast to standard bit mutation, smaller values of N are better for power-law mutation, as it is capable of easily creating missing solutions.

Keywords: MOEA/D · multi-objective optimization · runtime analysis · power-law mutation.

1 Introduction

Many real-world problems require the simultaneous optimization of different objectives. In this setting, known as *multi-objective optimization*, different solutions

may not be comparable based on their objective values, as one solution can win over another solution in one objective, but lose in another objective. This results in a set of incomparable optimal objective values, commonly referred to as *Pareto front*. The aim in multi-objective optimization is to find the Pareto front of a problem, or a good approximation thereof.

Due to their population-based and heuristic nature, evolutionary algorithms lend themselves very well to multi-objective optimization, and they have been successfully applied for decades to a plethora of hard multi-objective optimization problems [32]. This strong interest has led to a variety of algorithms [24, 32], following different paradigms.

From the early days of theoretical analyses of evolutionary algorithms on, multi-objective evolutionary algorithms have been analyzed also via theoretical means [13, 17, 21, 22]. This area saw a boost of activity in the last two years, when the first mathematical runtime analysis of the NSGA-II [31] inspired many deep analyses of this important algorithm and variants such as the NSGA-III or SMS-EMOA [1–4, 10–12, 20, 23, 26–30].

A substantially different, yet also very important algorithm is the *decomposition-based multi-objective evolutionary algorithm* (MOEA/D) [25]. It decomposes a multi-objective optimization problem f into various single-objective subproblems of f. These subproblems are optimized in parallel. While doing so, the non-dominated solutions for f are maintained in an archive.

Despite its popularity in empirical research and good performance in real-world problems [24, 32], the MOEA/D has not been extensively studied theoretically [5, 15, 16, 19]. In particular, it is currently not known how the basic MOEA/D using only standard mutation operators as variation operators finds Pareto optima that are not already an optimum of one of the subproblems (we refer to Section 2 for a detailed discussion of the previous works). We recall that the MOEA/D solves a number of single-objective subproblems essentially via single-objective approaches. Hence, if all Pareto optima of the original problem appear as optima of subproblems (we call these *g-optima* in the remainder), this is the setting regarded, e.g., in [19], then the multi-objective aspect of the problem vanishes and the only question is how efficiently the single-objective approaches solve the subproblems.

Our Contribution. Naturally, in a typical application of the MOEA/D, one cannot assume that the subproblems are sufficiently numerous and evenly distributed so that each Pareto optimum appears as g-optimum. To better understand how the MOEA/D copes with such situations, we study in this work mathematically how the MOEA/D computes the full Pareto front when started with a population consisting of all g-optima. In this first work on this topic, as often in the mathematical runtime analysis, we consider the basic OneMinMax (OMM) benchmark [14]. As in most previous works, we assume that the $N+1$ subproblems are constructed in a manner that the corresponding g-optima are spread equidistantly, with respect to the Hamming distance, across the Pareto front. We regard the basic MOEA/D using standard bit mutation as only varia-

tion operator. Interestingly, this is the first theoretical analysis of the MOEA/D in this setting (apart from the case that the g-optima cover the Pareto front [19]). Hence our results (Theorem 2), together with standard estimates on this time to compute the g-optima (Corollary 3), also give the first estimates on the full runtime of the MOEA/D in this setting (Corollary 1).

Since our results show that the main bottleneck for computing points on the Pareto front that are far from all g-optima is that standard bit mutation rarely flips many bits (due to the concentration behavior of the binomial distribution), we also resort to the heavy-tailed mutation operator proposed by Doerr et al. [9]. Interestingly, this allows for drastic speed-ups when considering the phase of the optimization that starts with all g-optima (Theorem 3).

In detail, our various results provide us with expected-runtime estimates for the MOEA/D with either mutation operator to optimize OMM (Corollaries 1 and 2). These results prove, respectively, an expected number of $O(nN \log n + n^{n/(2N)} N \log n)$ function evaluations for the MOEA/D with standard bit mutation where n is the problem dimension and $N+1$ is the population size, and $O\big(nN \log n + n^\beta \log n\big)$ expected function evaluations for power-law mutation with power-law exponent $\beta \in \mathbb{R}_{>1}$ being a constant. In both results, the second term refers to the interesting phase where the algorithm is initialized with all g-optima and is tasked to find the remaining Pareto optima of OMM.

Our overall bound for standard bit mutation yields $O(n^2 \log n)$ in the best case of $N = n$, matching the result by Li et al. [19, Proposition 4]. For general N, the second term in our bound suggests that the MOEA/D performs best if $N \in [\frac{n}{2}..n]$ and that the runtime is super-polynomial once $N = o(n)$. Moreover, we prove a lower bound of $\Omega(n^{(1/2)(n/N+1)}\sqrt{N}2^{-n/N})$ for this second term (Theorem 2), which supports this runtime characterization. However, this lower bound is not necessarily applicable to the *entire* optimization of the MOEA/D on OMM, as we only prove it for a certain phase of the optimization process. Nonetheless, we believe that it is true for sufficiently large n. We go more into detail about this behavior in Section 5 and especially in Section 5.1. Overall, our bounds suggest that standard bit mutation performs better when N is large.

Our upper bound for power-law mutation is best once $N = O(n^{\beta-1})$, resulting in a runtime bound of $O(n^\beta \log n)$. The bound $O(n^\beta \log n)$ stems from the second phase of the optimization, where the algorithm is initialized with all g-optima (Theorem 3). It is noteworthy that this bound does not depend on N, as, roughly, the time to fill in the gaps between g-optima is inversely proportional to the cost of performing $N+1$ function evaluations each iterations. Hence, the parameter N only influences our bound for the first phase of finding all g-optima. Overall, this suggests that the MOEA/D performs better when N is small, which is opposite of the behavior with standard bit mutation. Moreover, the bound for power-law mutation is $O(n^2 \log n)$ in the worst case, matching the best case of our bound for standard bit mutation. This suggests that power-law mutation is more preferable than standard bit mutation in our setting, as power-law mutation exhibits a far more robust runtime behavior.

Last, for each of the two phases we consider, we get independent results that hold for a larger class of algorithms (Lemmas 3 and 4) or functions (Theorems 2 and 3). Moreover, we prove an anti-concentration bound for the hypergeometric distribution close around its expected value (Lemma 2). Due to the more general setting, we believe that these results are of independent interest.

2 Related Work

Theoretical analyses of the MOEA/D so far are scarce. Most results do not consider the MOEA/D with only standard mutation operators. And those that do make simplifying assumptions about the decomposition, using problem-specific knowledge. We also note that we could not find any proofs for how the MOEA/D finds its reference point (one of its parameters), which is important in the general scenarios. If the reference point is mentioned, it is immediately assumed to be best-possible.

Li et al. [19] conducted the first mathematical analysis of the MOEA/D. They study the runtime of the algorithm on the classic bi-objective OneMinMax (OMM) [14] and LeadingOnesTrailingZeros (LOTZ)[3] benchmarks of size n, that is, the number of objective-function evaluations until the MOEA/D finds all Pareto optima of the problem. Both benchmarks have a Pareto front of size $n+1$. The authors consider a decomposition of each problem into $n+1$ subproblems—matching the size of the Pareto front—, and they assume that the MOEA/D uses standard bit mutation. The authors prove that if the subproblems are chosen such that the $n+1$ g-optima correspond one-to-one to the Pareto front of the problem, then the MOEA/D optimizes OMM in $O(n^2 \log n)$ expected function evaluations, and LOTZ in $O(n^3)$. We note that this requires a different subproblem structure for OMM and LOTZ. Moreover, the authors show that if the MOEA/D uses the commonly used subproblem structure of OMM for LOTZ, then the g-optima do not cover the entire Pareto front of LOTZ, for sufficiently large n. In this case, the authors argue that optimization takes a long time, as the missing solutions need to be found, but these arguments are not made formal.

Huang and Zhou [15] analyze the MOEA/D when using contiguous hypermutations, a non-standard mutation operator stemming from artificial immune systems. The authors study two versions of contiguous hypermutation, and they consider the OMM and the LOTZ benchmarks as well as a deceptive bi-objective problem and one containing a plateau. Moreover, the authors consider a general decomposition of each problem into $N+1$ subproblems. This decomposition is always the same and corresponds to the commonly used one, which results in evenly spread g-optima on the Pareto front of OMM, also analyzed by Li et al. [19]

[3] We note that the authors call OMM COCZ, and that they consider a version of LOTZ, called LPTNO, that considers strings over $\{-1, 1\}$ instead of binary strings, effectively replacing 0s with -1s. This changes some values for certain parameters but effectively results in the same bounds, to the best of our knowledge. Hence, we use the names OMM and LOTZ throughout this section.

above. Huang and Zhou [15] prove that both MOEA/D variants optimize each of the four benchmarks in $O(Nn^2 \log n)$ expected function evaluations. Moreover, they prove the same runtime bounds for the 4-objective versions of OMM and LOTZ. Overall, their results suggest that a choice of $N = O(1)$ is more beneficial, as the hypermutation operators are capable to cover larger distances than standard bit mutation and can thus find Pareto optima efficiently that do not correspond to g-optima.

Huang et al. [16] analyze a variant of the MOEA/D that employs standard bit mutation as well as one-point crossover at a rate of 0.5. When performing crossover, one of the parents is the best solution of the current subproblem g and the other one is chosen uniformly at random among the non-dominated solutions of the subproblems closest to g with respect to the Euclidean distance in one of their parameters. The problem decomposition is always the commonly used one with equidistant g-optima, as in the two papers discussed above. The authors consider essentially the same four[4] problems as Huang and Zhou [15] above, and they also consider a general number of $N + 1$ subproblems. For LOTZ and the plateau function, the authors proved an expected runtime of $O(Nn^2)$ function evaluations, and an $O(Nn \log n)$ bound for OMM and the deceptive function.

Very recently, Do et al. [5] analyzed the MOEA/D on the multi-objective minimum-weight-base problem, which is an abstraction of classical NP-hard combinatorial problems. Their MOEA/D variant uses a decomposition based on weight scalarization, different from the previous works above. The authors then prove that this variant finds an approximation of the Pareto front of the problem within expected fixed-parameter polynomial time.

3 Preliminaries

We denote the natural numbers by $\mathbb{N}$, including 0, and the real numbers by $\mathbb{R}$. For $a, b \in \mathbb{R}$, let $[a..b] = [a, b] \cap \mathbb{N}$ and $[a] = [1..a]$.

Let $n \in \mathbb{N}_{\geq 1}$. We consider *bi-objective* optimization problems, that is, functions $f\colon \{0,1\}^n \to \mathbb{R}^2$. We always assume that the dimension $n \in \mathbb{N}_{\geq 1}$ is given implicitly. When using big-O notation, it refers to asymptotics in this n. In this sense, an event occurs *with high probability* if its probability is at least $1 - o(1)$.

We call a point $x \in \{0,1\}^n$ an *individual* and $f(x)$ the *objective value* of x. For all $i \in [n]$ and $j \in [2]$, we let x_i denote the i-th component of x and $f_j(x)$ the j-th component of $f(x)$. Moreover, let $|x|_0$ denote the number of 0s of x, and let $|x|_1$ denote its number of 1s.

For all $u, v \in \mathbb{R}^2$, we say that v *weakly dominates* u (written $v \succeq u$) if and only if for all $i \in [2]$ holds that $f_i(v) \geq f_i(u)$. We say that v *strictly dominates* u if and only if one of these inequalities is strict. We extend this notation to individuals, where a dominance holds if and only if it holds for their respective objective values.

[4] The authors actually treat LOTZ and LPTNO as two functions, but the results are the same, which is why we count it as one problem.

We consider the maximization of bi-objective functions f, that is, we are interested in $\succeq$-maximal elements, called *Pareto-optimal* individuals. The set of all objective values that are not strictly dominated, that is, the set $F^* \coloneqq \{v \in \mathbb{R}^2 \mid \exists y \in \{0,1\}^n \nexists x \in \{0,1\}^n \colon v = f(y) \wedge f(x) \succ v\}$, is called the *Pareto front* of f.

OneMinMax We analyze the OneMinMax (OMM) benchmark [14] problem, which returns for each individual the number of 0s as the first objective, and the number of 1s as the second objective. Formally, $\mathrm{OMM}\colon x \mapsto (|x|_0, |x|_1)$.

Note that each individual is Pareto optimal. The Pareto front of OneMinMax is $\{(i, n-i), i \in [0..n]\}$.

Mathematical Tools We use the well-known multiplicative drift theorem [7] with tail bounds [6]. We present the theorem in a fashion that is sufficient for our purposes. Throughout this article, if we write for a stopping time T and two unary formulas P and Q that for all $t \in \mathbb{N}$ with $t < T$ holds that $P(t) \geq Q(t)$, then we mean that for all $t \in \mathbb{N}$ holds that $P(t) \cdot \mathbb{1}\{t < T\} \geq Q(t) \cdot \mathbb{1}\{t < T\}$, where $\mathbb{1}$ denotes the indicator function.

Theorem 1 (Multiplicative drift [7], upper tail bound [6][18, Theorem 2.4.5]). *Let $n \in \mathbb{N}$, let $(X_t)_{t\in\mathbb{N}}$ be a random process over $[0..n]$, and let $T = \inf\{t \in \mathbb{N} \mid X_t = 0\}$. Moreover, assume that there is a $\delta \in \mathbb{R}_{>0}$ such that for all $t \in \mathbb{N}$ with $t < T$ holds that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq \delta X_t$. Then, $\mathrm{E}[T] \leq \frac{1}{\delta}(1 + \ln n)$, and for all $r \in \mathbb{R}_{\geq 0}$ holds that $\Pr[T > \frac{1}{\delta}(r + \ln n)] \leq \mathrm{e}^{-r}$.*

4 The MOEA/D

We analyze the *decomposition-based multi-objective evolutionary algorithm* (MOEA/D; Algorithm 1) [25] for multi-objective optimization. The MOEA/D decomposes its objective function into a pre-specified number of single-objective subproblems. These subproblems are optimized in parallel. The MOEA/D maintains an archive of the $\succeq$-best solutions found so far, allowing it to find Pareto-optimal solutions for the original problem while only explicitly optimizing the subproblems.

More formally, given, besides others, a bi-objective optimization problem f as well as a *decomposition number* $N \in [n]$ and a *weight vector* $w \in [0,1]^{N+1}$, the MOEA/D optimizes f by decomposing f into $N+1$ single-objective subproblems $\{g_i \colon \{0,1\}^n \times \mathbb{R}^2 \to \mathbb{R}\}_{i\in[0..N]}$, weighted by w. Each of these subproblems is subject to *minimization* (of an error). The MOEA/D maintains a population of $N+1$ individuals $(x_i)_{i\in[0..N]} \in (\{0,1\}^n)^{N+1}$ as well as a *reference point* $z^* \in \mathbb{R}^2$ such that for each $i \in [0..N]$, individual x_i is the currently best solution found for subproblem i with respect to z^*. Ideally, the reference point is a point such that for all $j \in [2]$, value z^*_j is optimal for objective f_j. To this end, the MOEA/D updates z^* whenever it optimizes a subproblem. Moreover, the algorithm maintains a set $P \subseteq \mathbb{R}^2$ (the *Pareto front*) of non-dominated objective values.

Algorithm 1: The MOEA/D [25] maximizing a bi-objective problem $f\colon \{0,1\}^n \to \mathbb{R}^2$. See also Section 4.

Input: A decomposition number $N \in \mathbb{N}_{\geq 1}$, a weight vector $w \in [0,1]^{N+1}$, subproblems $\{g_i\}_{i\in[0..N]}$, a mutation operator $\mathrm{mut}\colon \{0,1\}^n \to \{0,1\}^n$, and a termination criterion.

1 *Initialization*: for each $i \in [0..N]$, choose x_i uniformly at random from $\{0,1\}^n$; set $z_1^* = \max_{i\in[0..N]} f_1(x_i)$, $z_2^* = \max_{i\in[0..N]} f_2(x_i)$, and iteratively add $f(x_i)$ to P if there is no $j \in [0..i-1]$ such that x_i is weakly dominated by x_j;
2 **while** stopping criterion is not met **do**
3 **for** each subproblem $i \in [0..N]$ **do**
4 *Mutation*: $y \leftarrow \mathrm{mut}(x_i)$;
5 *Update* z^*: set $z_1^* \leftarrow \max(z_1^*, f_1(y))$, $z_2^* \leftarrow \max(z_2^*, f_2(y))$;
6 *Update* x_i: if $g_i(y, z^*) \leq g_i(x_i, z^*)$, then $x_i \leftarrow y$;
7 *Update* P: remove all elements weakly dominated by $f(y)$ from P and add $f(y)$ to P if it is not weakly dominated by an element of P;

We consider subproblems that measure the maximum distance to the reference point, known as Chebyshev approach. That is, for all $i \in [0..N]$, $x \in \{0,1\}^n$, and $z^* \in \mathbb{R}^2$, it holds that

$$g_i(x, z^*) = \max\big(w_i \cdot |z_1^* - f_1(x)|, (1-w_i) \cdot |z_2^* - f_2(x)|\big). \tag{1}$$

When minimizing subproblem $i \in [0..N]$, the MOEA/D picks x_i as parent and mutates it according to a given mutation operator. Afterward, it compares the offspring of the mutation to x_i and selects the better one.[5]

We define the *runtime* of the MOEA/D on f as the number of function evaluations of f until the Pareto front P of the algorithm is equal to the Pareto front F^* of f for the first time.

In this article, we consider the MOEA/D with different mutation operators.

Mutation Operators We consider both *standard bit mutation* as well as *power-law mutation* [9]. Let $x \in \{0,1\}^n$ be the input (the *parent*) of the mutation. Both operators create a new individual (the *offspring*) by first copying x and then adjusting its bit values. Standard bit mutation flips, for all $i \in [n]$, bit x_i independently with probability $1/n$.

Power-law mutation requires a parameter $\beta > 1$ (the *power-law exponent*) as input and utilizes the power-law distribution $\mathrm{Pow}(\beta, n)$ over $[n]$, defined as follows. Let $C_\beta = \sum_{i\in[n]} i^{-\beta}$ as well as $X \sim \mathrm{Pow}(\beta, n)$. For all $i \in [n]$, it holds that $\Pr[X = i] = i^{-\beta}/C_\beta$. The power-law mutation first draws $X \sim \mathrm{Pow}(\beta, n)$ (the *mutation strength*) and then flips an X-cardinality subset of positions in x chosen uniformly at random.

[5] We note that the general MOEA/D allows to specify neighborhoods among the subproblems, which exchange non-dominated solutions among each other. In this article, we focus on no exchange.

The following lemma bounds the probability to mutate an individual with at most $\frac{n}{4}$ 0s into one with at most $\frac{n}{2}$, showing that the probability is proportional to the distance in the number of 0s. Its proof makes use of the anti-concentration of the *hypergeometric distribution* (Lemma 2) around its expectation, which we discuss after Lemma 1. We note that, due to space restrictions, the proofs of these results are on arXiv [8].

Lemma 1. *Let $x \in \{0,1\}^n$ with $u \coloneqq |x|_0 \in [0..\frac{n}{4}]$ and let $v \in [u+1..\frac{n}{2}-1]$. Moreover, let $\beta \in \mathbb{R}_{>1}$, and let mut_β denote the power-law mutation with power-law exponent β. Then $\Pr[|\mathrm{mut}_\beta(x)|_0 = v] = \Omega\big((v-u)^{-\beta}\big)$.*

Hypergeometric distribution. The hypergeometric distribution (Hyp) takes three parameters, namely, $n \in \mathbb{N}$, $k \in [0..n]$, and $r \in [0..n]$, and it has a support of $[\max(0, r+k-n)..\min(k,r)]$. A random variable $X \sim \mathrm{Hyp}(n,k,r)$ describes the number of good balls drawn when drawing r balls uniformly at random without replacement from a set of n balls, out of which k are good. That is, for all $i \in [\max(0, r+k-n)..\min(k,r)]$ holds $\Pr[X=i] = \binom{k}{i}\binom{n-k}{r-i}/\binom{n}{r}$. Moreover, $\mathrm{E}[X] = r\frac{k}{n}$ as well as $\mathrm{Var}[X] = r\frac{k}{n}(1-\frac{k}{n})\frac{n-r}{n-1}$. In the context of power-law mutation, n represents the number of bits, k the number of specific bits to flip (for example, 0-bits), and r represents the mutation strength.

The following lemma shows that the hypergeometric distribution has a reasonable probability of sampling values that deviate only by the standard deviation from its expected value.

Lemma 2. *Let $n \in \mathbb{N}$, $k \in [0..\frac{n}{2}]$, and $r \in [0..\frac{3}{4}n]$.*

Moreover, let $H \sim \mathrm{Hyp}(n,k,r)$. Then there exists a constant $\gamma \in \mathbb{R}_{>0}$ such that for all $x \in [\mathrm{E}[H] - 2\sqrt{\mathrm{Var}[H]}, \mathrm{E}[H] + 2\sqrt{\mathrm{Var}[H]}]$ it holds that $\Pr[H=x] \geq \gamma/\sqrt{\mathrm{Var}[H]}$.

5 Runtime Analysis

We analyze the MOEA/D (Algorithm 1) on the OneMinMax (OMM) function (of dimension $n \in \mathbb{N}_{\geq 1}$) with subproblems spread uniformly across the Pareto front of OMM, which is the typical way to pick weights [25] and was also used in the first mathematical analysis of the MOEA/D [19]. To this end, we make the following assumptions about the input of the algorithm, which we refer to as the *parametrization*: We consider decomposition numbers $N \in [n]$, we define the weight vector as $w = (i\frac{n}{N})_{i \in [0..N]}$, and we consider the subproblems as defined in equation (1). In our calculations, we assume that N divides n, as this avoids rounding, but we note that all results are equally valid if N does not divide n, although the computations become more verbose, adding little merit. We note that, due to space restrictions, some of our proofs are only on arXiv [8].

Our main results are the following, bounding the expected runtime for both standard bit mutation and power-law mutation.

Corollary 1. *Consider the MOEA/D maximizing* OMM *with standard bit mutation and with the parametrization specified at the beginning of Section 5. Then its expected runtime is* $O(nN \log n + n^{n/(2N)} N \log n)$ *function evaluations.*

Corollary 2. *Consider the MOEA/D maximizing* OMM *with power-law exponent* $\beta \in \mathbb{R}_{>1}$ *and with the parametrization specified at the beginning of Section 5. Then its expected runtime is* $O\big(nN \log n + n^{\beta} \log n\big)$ *function evaluations.*

Both runtime results present two terms, which stem from the two *phases* into which we separate our analysis. The first term in the results is an upper bound of the first phase, which is the number of function evaluations it takes the MOEA/D to optimize all subproblems. We call the solutions to these subproblems *g-optima*. Our bounds for either mutation operator for the first phase are the same (Corollary 3). The optimization mainly resembles performing $N + 1$ times an optimization similar to that of the well-known $(1+1)$ evolutionary algorithm on the OneMax benchmark function. For $N = n$, our result for standard bit mutation recovers the result by Li et al. [19, Proposition 4]. We go into detail about the analysis in Section 5.1.

The second phase starts immediately after the first phase and stops once the Pareto front of the MOEA/D covers the Pareto front of OMM. During this analysis, we only assume that the MOEA/D found the g-optima so far. Thus, in the worst case, it still needs to find all other Pareto-optima of OMM. To this end, each such optimum needs to be created via mutation *directly* from one of the g-optima, as the latter are not being replaced, due to them being optimal. Depending on the *gap size*, that is, the number of Pareto optima between two g-optima of neighboring subproblems, the mutation operator makes a big difference on the expected runtime bound. We analyze both mutation operators separately in Section 5.2.

Regarding Corollary 1, we see that the upper bound for standard bit mutation only handles values for $N \in [\frac{n}{2}..n]$ without any slowdown in comparison to the first phase. For smaller values, the upper bound is dominated by the second term and becomes super-polynomial once $N = o(n)$. In Theorem 2, we prove a lower bound for the second phase that shows that the expected runtime is also at least super-polynomial once $N = o(n)$. However, as this lower bound only applies to the second phase, it may be possible during the whole optimization of OMM that the algorithm finds most of the OMM-Pareto optima already during the first phase, although we conjecture this not to happen for sufficiently small N.

For power-law mutation (Corollary 2), the bound for the second phase is independent of N (Theorem 3). This shows that the power-law mutation picks up missing Pareto optima fairly quickly. In fact, even for an optimal value of $N = n$ for standard bit mutation, which results in a bound of $O(n^2 \log n)$ for the second phase, the bound for power-law mutation is still smaller by a factor of $n^{\beta-2}$, which is less than 1 if we further assume that $\beta \in (1, 2)$, which is a typical range of values used for β in practice. Moreover, for the whole optimization, the upper bound for power-law mutation is best once $N = O(n^{\beta-1})$, that is, for smaller values of N. This is in contrast to the upper bound for standard bit mutation, which gets better for larger values of N.

Overall, our results suggest that standard bit mutation profits from having many subproblems, as creating initially skipped solutions may be hard to create once all subproblems are optimized. In contrast, power-law mutation is slowed down by optimizing many subproblems in parallel. Instead, it profits from optimizing fewer such subproblems and then creating initially skipped solutions.

In the following, we first analyze the first phase (Section 5.1) and then the second phase (Section 5.2).

5.1 First Phase

Recall that the first phase considers optimization of OMM only until all subproblems are optimized, that is, until all g-optima are found. Our main result is the following and shows that finding the g-optima is not challenging for the MOEA/D, regardless of the mutation operator.

Corollary 3. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Section 5. Then the expected time until P contains all g-optima of* OMM *is $O(nN \log n)$ function evaluations for both standard bit mutation and power-law mutation with power-law exponent $\beta \in \mathbb{R}_{>1}$.*

For our proof of Corollary 3, we split the first phase into two parts. The first part considers the time until the reference point z^* is optimal, that is, until $z^* = (n, n)$. For this part, only the optimization of g_0 and g_N is relevant.

The second part starts with an optimal reference point and considers the time until all g-optima are found. For this part, we consider the optimization of an arbitrary subproblem and multiply the resulting time by roughly $N \log N$, as we consider $N + 1$ subproblems and we wait until all of them are optimized.

In order to prove results that hold for the MOEA/D with standard bit mutation as well as with power-law mutation, we consider a general mutation operator, which we call *general mutation*. It takes one parameter $p_1 \in (0, 1]$ and works as follows: It chooses to flip exactly one bit in the parent with probability p_1 (and it flips any other number with probability $1 - p_1$). Conditional on flipping exactly one bit, it flips one of the n bits of the parent uniformly at random and returns the result. Note that standard bit mutation is general mutation with $p_1 = (1 - \frac{1}{n})^{n-1}$ and that power-law mutation with power-law exponent β is general mutation with $p_1 = 1/C_\beta$. Both of these values are constants.

For the first part, we get the following result.

Lemma 3. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Section 5 and with general mutation with parameter $p_1 \in (0, 1]$. Then the expected time until $z^* = (n, n)$ holds is $O(\frac{n}{p_1} N \log n)$ function evaluations.*

Proof. Let T be the first iteration such that $z_1^* = n$. Without loss of generality, we only analyze $\mathrm{E}[T]$, as the analysis for $z_2^* = n$ is identical when changing all 1s into 0s and vice versa in the individuals in all arguments that follow. Thus, the expected runtime that we aim to prove is at most $2\,\mathrm{E}[T]$ by linearity

of expectation. Hence, we are left to show that $\mathrm{E}[T] = O(\frac{n}{p_1} \log n)$, where the factor of N in the bound of Lemma 3 stems from the MOEA/D making $N+1$ function evaluations per iteration. To this end, we only consider the optimization of g_N.

By equation (1), the choice of the weight vector w, and the definition of OMM, it follows for all $x \in \{0,1\}^n$ and $z \in \mathbb{R}^2$ that $g_N(x,z) = \max(|z_1 - |x|_1|, 0) = |z_1 - |x|_1|$. In each iteration, let x_N denote the best-so-far solution for g_N at the beginning of the iteration, and let y denote its offspring generated via mutation. Note that, due to how z^* is initialized and updated, in each iteration, it holds that $z_1^* \geq \max(f_1(x_N), f_1(y)) = \max(|x_N|_1, |y|_1)$. Thus, if $|y|_1 > |x_N|_1$, then $g(y, z^*) < g(x_N, z)$, and thus x_N is updated to y at the end of the iteration. Hence, the optimization of g_N proceeds like a $(1+1)$-EA-variant with general mutation optimizing OneMax.

More formally, let $(X_t)_{t \in \mathbb{N}}$ such that for each $t \in \mathbb{N}$, the value X_t denotes the number of 0s in x_N at the end of iteration t. Note that $X_T = 0$. We aim to apply the multiplicative drift theorem (Theorem 1) to X with T. By the definition of the mutation operator, it follows for all $t < T$ that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq X_t \frac{p_1}{n}$, since it is sufficient to choose to flip one bit (with probability p_1) and then to flip one of the X_t 0s of x_N (at the beginning of the iteration), which are chosen uniformly at random. Thus, by Theorem 1, it follows that $\mathrm{E}[T] \leq \frac{n}{p_1}(1 + \ln n)$, concluding the proof. □

For the second part, we get the following result.

Lemma 4. *Consider the MOEA/D maximizing* OMM *with the parametrization specified at the beginning of Section 5 and with $z^* = (n, n)$ and with general mutation with parameter $p_1 \in (0, 1]$. Then the expected time until P contains all g-optima of* OMM *is $O(\frac{n}{p_1} N \log n)$ function evaluations.*

Proof. Let $i \in [0, N]$. We bound with high probability the time T until g_i is optimized, only counting the function evaluations for subproblem i. The result of Lemma 4 follows then by considering the maximum runtime among all values of i and multiplying it by $N+1$, as we perform $N+1$ function evaluations per iteration and optimize all subproblems in parallel. We bound the maximum with high probability by taking a union bound over all $N+1$ different values for i. If the maximum of T over all i is at least $B \in \mathbb{R}_{\geq 0}$ with probability at most $q \in [0, 1)$, then we get the overall expected runtime by repeating our analysis $\frac{1}{1-q}$ times in expectation, as the actual runtime is dominated by a geometric random variable with success probability $1 - q$. The overall expected runtime is then $O(BN\frac{1}{1-q})$ Thus, it remains to show that $\Pr[T > \frac{n}{p_1}\big(\ln(n) + 2\ln(N+1)\big)] \leq (N+1)^{-2}$, as it then follows that $q \leq (N+1)^{-1}$ and thus $\frac{1}{1-q} \leq 2$. We aim to prove this probability bound with the multiplicative drift theorem (Theorem 1).

Let $(X_t)_{t \in \mathbb{N}}$ be such that for all $t \in \mathbb{N}$, value X_t denotes the value of $g_i(x_i, z^*)$ at the beginning of the iteration. Note that $X_T = 0$ and that for all $t < T$ holds that $X_t \in [0..n]$ and that for X_t to reduce (if $X_t > 0$), it is sufficient to flip one of the X_t bits that reduce the distance. Thus, by the definition of the mutation

operator, it follows that $\mathrm{E}[X_t - X_{t+1} \mid X_t] \geq X_t \frac{p_1}{n}$. Overall, by (Theorem 1), it follows that $\Pr[T > \frac{n}{p_1}\big(\ln(n)+2\ln(N+1)\big)] \leq (N+1)^{-2}$. The proof is concluded by noting that $N \leq n$ and thus $\ln(n) + 2\ln(N+1) = O(\log n)$. □

By the linearity of expectation, the expected time of the first phase is the sum of the expected runtimes of both parts. Moreover, since standard bit mutation and power-law mutation are both a general mutation with $p_1 = \Theta(1)$, we can omit p_1 in the asymptotic notation. Overall, we obtain Corollary 3.

5.2 Second Phase

Recall that the second phase assumes that the MOEA/D starts with the g-optima as its solutions to the subproblems, and it lasts until all OMM-Pareto optima are found.

For this phase, the actual objective function is not very important. All that matters is that if the solutions $(x_i)_{i\in[0..N]}$ of the MOEA/D are such that for all $i \in [0..N]$ holds that $|x_i|_1 = i\frac{n}{N}$, then x_i is optimal for g_i. We refer to such a situation as *evenly spread g-optima*.

Since there is a drastic difference between the runtimes of standard bit mutation and power-law mutation, we analyze these two operators separately.

Standard Bit Mutation Since the standard bit mutation is highly concentrated around flipping only a constant number of bits, it does not perform well when it needs to fill in larger gaps. The following theorem is our main result, and it proves an upper and a lower bound for the expected runtime of the second phase. These bounds are not tight, but they show that the runtime is already super-polynomial once $N = o(n)$.

Theorem 2. *Consider the MOEA/D maximizing a bi-objective function with evenly spread g-optima and with standard bit mutation, using the parametrization specified at the beginning of Section 5. Moreover, assume that $\frac{n}{2N}$ is integer and that the algorithm is initialized with $(x_i)_{i\in[0..N]}$ such that for all $i \in [0..N]$ holds that $|x_i|_1 = i \cdot \frac{n}{N}$. Then the expected runtime until for each $j \in [0..n]$ at least one individual with j 0s is created via mutation is $O(n^{n/(2N)}N\log n) \cap \Omega(n^{(1/2)(n/N+1)}\sqrt{N}2^{-n/N})$ function evaluations.*

Power-Law Mutation The power-law mutation allows to create individuals at larger distance from its parent with far higher probability than standard bit mutation. Our main result is the following theorem, which shows that the MOEA/D with power-law mutation optimizes the second phase of OMM efficiently. As before, we state this theorem in a more general fashion.

Theorem 3. *Consider the MOEA/D optimizing a bi-objective function with evenly spread g-optima, using the parametrization specified at the beginning of Section 5. Moreover, assume that the algorithm is initialized with $\{i \cdot \frac{n}{N}\}_{i\in[0..N]}$. Then the expected runtime until for each $j \in [0..n]$ at least one individual with j 0s is created via mutation is $O(n^\beta \log n)$.*

The bound of Theorem 3 does not depend on N, in contrast to the bound on the first phase (Corollary 3). The reason for our bound not depending on N is roughly that the effort to fill in a gap between to g-optima is inversely proportional to the cost of an iteration, namely, $N+1$. Thus, a smaller value of N leads to faster iterations but more iterations spend to fill the gaps, and vice versa.

Our (omitted) proof of Theorem 3 makes use of the following lemma, which bounds the probability to create a specific individual from any of the g-optima in a single iteration of the MOEA/D.

Lemma 5. *Consider a specific iteration during the optimization of the MOEA/D of a bi-objective function with evenly spread g-optima, using the parametrization specified at the beginning of Section 5. Moreover, assume that the algorithm is initialized with $\{i \cdot \frac{n}{N}\}_{i\in[0..N]}$. Last, let $u \in [0..\frac{n}{2}]$. Then the probability that an individual with u 0-bits is produced during mutation in this iteration is $\Omega\big(N(n^{-\beta})\big)$.*

Proof. Let i be such that $ni/N < u \leq n(i+1)/N$. Clearly, $i \leq N/2$, as $u \leq n/2$. Note that are at least $i/4$ values of $j \in [0..N/4]$ such that $jn/N \leq n/4$. By Lemma 1, each individual x_j mutates into an individual with u 0-bits with probability $\Omega\big(\big((i+1)n/N\big)^{-\beta}\big)$. Because there are $\Omega(i)$ such possible mutations during an iteration, the probability of generating at least one during an iteration is $\Omega\big(i^{1-\beta}n^{-\beta} \cdot N^{\beta}\big) = \Omega(N\big(\frac{N}{i}\big)^{\beta-1}n^{-\beta}) = \Omega\big(Nn^{-\beta}\big)$, concluding the proof. □

6 Conclusion

We studied the impact of the decomposition number N of the MOEA/D [25] on the classic multi-objective benchmark OneMinMax (OMM) [14] theoretically. Our analyses considered subproblems that are evenly spread out on the Pareto front of OMM. Especially, we studied the expected runtime when starting with all optima of the subproblems (the *g-optima*) and requiring to find the remaining Pareto optima of OMM.

One of our theoretical results (Theorem 3) shows that using power-law mutation allows the MOEA/D to efficiently find the entire Pareto front of OMM even if it is initialized only with the g-optima. Interestingly, this bound is independent of the number of problems N and thus the number of initially missing Pareto optima between two neighboring g-optima. Together with our general bound for finding all g-optima (Corollary 3), this shows that the MOEA/D with power-law mutation always optimizes OMM efficiently (Corollary 2). Depending on N, our total-runtime bound ranges from $O(n^2 \log n)$ in the worst case to $O(n^{\beta} \log n)$ in the best case of $N = O(n^{\beta-1})$. This suggests that the MOEA/D is, in fact, slowed down when using many subproblems, despite large values of N implying that the subproblems cover the Pareto front of OMM better than smaller values. The reason is that a large value of N roughly translates to optimizing the same problem N times. With the power-law mutation, it is better to optimize fewer and therefore more diverse subproblems and to then find the remaining Pareto optima efficiently via the power-law mutation.

For standard bit mutation, when starting with all g-optima, we show (Theorem 2) that the MOEA/D is not capable of efficiently finding all Pareto optima if N is sufficiently small, as our lower bound is super-polynomial for $N = o(n)$. Nonetheless, for $N = \Theta(n)$, the expected runtime of the MOEA/D with standard bit mutation is polynomial in this part. This translates to an overall polynomial expected runtime (Corollary 1) for $N = \Theta(n)$ that it is even $O(n^2 \log n)$ for $N \in [\frac{n}{2}..n]$, matching our worst-case bound with power-law.

Overall, our results suggest a clear benefit of power-law mutation over standard bit mutation of the MOEA/D in the setting we considered. Not only is the power-law variant faster, it is also far more robust to the choice of N and thus to how the problem is decomposed.

For future work, it would be interesting to improve the upper bounds or prove matching lower bounds. A similar direction is to consider an exchange of best-so-far solutions among the subproblems. The classic MOEA/D supports such an exchange, which could potentially lead to finding the g-optima more quickly. Another promising direction is the study of different problem decompositions, for example, not-evenly spread subproblems or subproblem definitions different from equation (1). Last, we considered the OMM setting, with a stronger generalization some of our results (Theorems 2 and 3). However, it is not clear to what extent the benefit of power-law mutation carries over to problems with an entirely different structure, such as LeadingOnesTrailingZeros.

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

Disclosure of Interests. The authors declare no competing interests.

Bibliography

[1] Bian, C., Qian, C.: Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In: Parallel Problem Solving From Nature, PPSN 2022. pp. 428–441. Springer (2022)

[2] Cerf, S., Doerr, B., Hebras, B., Kahane, J., Wietheger, S.: 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. pp. 5522–5530. ijcai.org (2023)

[3] Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: Analysing the robustness of NSGA-II under noise. In: Genetic and Evolutionary Computation Conference, GECCO 2023. pp. 642–651. ACM (2023)

[4] Dang, D.C., Opris, A., Salehi, B., Sudholt, D.: A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In: Conference on Artificial Intelligence, AAAI 2023. pp. 12390–12398. AAAI Press (2023)

[5] Do, A.V., Neumann, A., Neumann, F., Sutton, A.M.: Rigorous runtime analysis of MOEA/D for solving multi-objective minimum weight base problems. In: Advances in Neural Information Processing Systems. pp. 36434–36448. Curran Associates (2023)

[6] Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**, 224–250 (2013)

[7] Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)

[8] Doerr, B., Krejca, M.S., Weeks, N.: Proven runtime guarantees for how the MOEA/D computes the pareto front from the subproblem solutions. CoRR **abs/2405.01014** (2024)

[9] Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2017. pp. 777–784. ACM (2017)

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

[11] Doerr, B., Qu, Z.: From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In: Conference on Artificial Intelligence, AAAI 2023. pp. 12408–12416. AAAI Press (2023)

[12] Doerr, B., Qu, Z.: Runtime analysis for the NSGA-II: provable speed-ups from crossover. In: Conference on Artificial Intelligence, AAAI 2023. pp. 12399–12407. AAAI Press (2023)

[13] Giel, O.: Expected runtimes of a simple multi-objective evolutionary algorithm. In: Congress on Evolutionary Computation, CEC 2003. pp. 1918–1925. IEEE (2003)

[14] Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimisation. Evolutionary Computation **18**, 335–356 (2010)
[15] Huang, Z., Zhou, Y.: Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In: Conference on Artificial Intelligence, AAAI 2020. pp. 2359–2366. AAAI Press (2020)
[16] Huang, Z., Zhou, Y., Chen, Z., He, X., Lai, X., Xia, X.: Running time analysis of MOEA/D on pseudo-Boolean functions. IEEE Transactions on Cybernetics **51**, 5130–5141 (2021)
[17] Laumanns, M., Thiele, L., Zitzler, E.: Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Transactions on Evolutionary Computation **8**, 170–182 (2004)
[18] Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 89–131. Springer (2020), also available at https://arxiv.org/abs/1712.00964
[19] Li, Y.L., Zhou, Y.R., Zhan, Z.H., Zhang, J.: A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation **20**, 563–576 (2016)
[20] Opris, A., Dang, D.C., Neumann, F., Sudholt, D.: Runtime analyses of NSGA-III on many-objective problems. In: Genetic and Evolutionary Computation Conference, GECCO 2024. ACM (2024), to appear
[21] Rudolph, G.: Evolutionary search for minimal elements in partially ordered finite sets. In: Evolutionary Programming, EP 1998. pp. 345–353. Springer (1998)
[22] Thierens, D.: Convergence time analysis for the multi-objective counting ones problem. In: Evolutionary Multi-Criterion Optimization, EMO 2003. pp. 355–364. Springer (2003)
[23] Wietheger, S., Doerr, B.: A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In: International Joint Conference on Artificial Intelligence, IJCAI 2023. pp. 5657–5665. ijcai.org (2023)
[24] Zhang, J., Xing, L.: A survey of multiobjective evolutionary algorithms. In: International Conference on Computational Science and Engineering (CSE). pp. 93–100. IEEE (2017)
[25] Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**, 712–731 (2007)
[26] Zheng, W., Doerr, B.: Better approximation guarantees for the NSGA-II by using the current crowding distance. In: Genetic and Evolutionary Computation Conference, GECCO 2022. pp. 611–619. ACM (2022)
[27] Zheng, W., Doerr, B.: Mathematical runtime analysis for the non-dominated sorting genetic algorithm II (NSGA-II). Artificial Intelligence **325**, 104016 (2023)
[28] Zheng, W., Doerr, B.: Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for many objectives. IEEE Transactions on Evolutionary Computation (2023), in press, https://doi.org/10.1109/TEVC.2023.3320278

[29] Zheng, W., Doerr, B.: Runtime analysis of the SMS-EMOA for many-objective optimization. In: Conference on Artificial Intelligence, AAAI 2024. AAAI Press (2024)
[30] Zheng, W., Li, M., Deng, R., Doerr, B.: How to use the metropolis algorithm for multi-objective optimization? In: Conference on Artificial Intelligence, AAAI 2024. AAAI Press (2024)
[31] Zheng, W., Liu, Y., Doerr, B.: A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In: Conference on Artificial Intelligence, AAAI 2022. pp. 10408–10416. AAAI Press (2022)
[32] Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multi-objective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation **1**, 32–49 (2011)