A Dynamic Decomposition Strategy for the MOEA/D With Proven Performance Guarantees

Benjamin Doerr[1], **Martin S. Krejca**[1], **Noé Weeks**[2]
[1] Laboratoire d'Informatique (LIX), CNRS, École Polytechnique,
Institut Polytechnique de Paris, Palaiseau, France
[2] École Normale Supérieure, Paris, France
firstname.lastname@polytechnique.edu, noe.weeks@ens.psl.eu

Abstract

The MOEA/D is one of the most successful multi-objective evolutionary algorithms. It operates by decomposing the optimization problem into single-objective subproblems, which are then solved in a co-evolutionary manner. The performance of the MOEA/D depends critically on this decomposition. In this work, we propose a novel way to adjust this decomposition on the fly, for the case of discrete bi-objective optimization. Different from all previous works in this direction, we support our new algorithm with a proven performance guarantee. This is the first work that rigorously proves the runtime of a dynamic multi-objective evolutionary algorithm. For monotonic distortions of the simple LOTZ benchmark, it is known that classic decompositions leave large holes in the Pareto front, making it very hard for the MOEA/D to cover the entire Pareto front. In contrast, we prove that our self-adjusting MOEA/D solves all these LOTZ problems in an expected number of $O(n^4)$ function evaluations. This relatively low runtime as well as the details of our mathematical proof show that the dynamic decomposition strategy indeed evolves a decomposition adjusted to each of these problem instances.

1 Introduction

Many real-world optimization problems feature several conflicting objectives. Such *multi-objective optimization problems* (MOOPs) usually do not possess a single optimal solution. Thus, a common approach is to compute a diverse set of *Pareto optima* (incomparable optimal trade-offs) and let a human decision maker pick one of these, based on their broader understanding of the problem.

Due to their population-based nature, genetic algorithms are one of the most prominent approaches for computing the Pareto front of a MOOP. The famous NSGA-II algorithm [DPAM02], for example, is massively used in practice and has reached over $50\,000$ citations on Google Scholar. We refer to [ZQL$^+$11] for a broader introduction to the use of genetic algorithms in MOO.

There are two general approaches to the design of multi-objective evolutionary algorithms (MOEAs). The classic one works with a population of individuals and employs diversity mechanisms to ensure that this population approaches the full Pareto front. This concept is found in the algorithms SPEA2 [ZLT01], NSGA-II [DPAM02], SMS-EMOA [BNE07], and NSGA-III [DJ14].

More recently, decomposition-based approaches were proposed. Here the idea is to decompose the MOOP into scalar subproblems and solve these in a co-evolutionary manner, that is, they are optimized in parallel and exchange solutions. The hope is that the co-evolutionary approach helps solve similar subproblems efficiently, but also that the evolutionary process finds Pareto optima that are not the optimum of one of the designated subproblems. We refer to a recent survey [XXM20] on decomposition-based MOEAs and note that the by far most prominent such algorithm is the MOEA/D developed by Zhang and Li [ZL07]. The survey reports that the MOEA/D with a static

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

decomposition oftentimes leads to a worse spread of the final solutions in terms of objective values when compared to a dynamic choice. This behavior shows repeatedly for continuous and for discrete combinatorial problems, and in the presence of crossover, oftentimes reported for the common Chebyshev decomposition [HJH19, ISMN17, LL11, LY20, QML$^+$14].

Algorithms using one joint population are relatively well understood. In particular, many mathematical analyses were conducted, first for the simplistic (G)SEMO family of algorithms (e.g., [DN21, Gie03, LTZ04, QYZ15, Thi03]), then for practically more relevant NSGA-II [CDH$^+$23, ZLD22], NSGA-III [ODNS24, WD23], SMS-EMOA [BZLQ23, ZD24], and SPEA2 [RBLQ24].

In contrast, the mathematical analysis of decomposition-based algorithms such as the MOEA/D is much less developed. Refraining from details, this area started only in 2016 with the seminal paper [LZZZ16], then seeing six follow-ups [DKW24, DNNS23, HZ20, HZC$^+$21, HZCH19, HZLL21].

With relatively few MOEA/D theory works, it comes as no surprise that these works often take simplistic assumptions. Most results assume that all Pareto optima appear also as optimum of one of the subproblems of the decomposition [HZC$^+$21, HZLL21, LZZZ16]; others employ non-standard variation operators [HZ20] or operators that are particularly suited for the problem at hand [HZCH19].

Without such optimistic assumptions, the performance of the MOEA/D is much less understood. Prominently, Li et al. [LZZZ16] showed that for certain monotonic rescalings of the common LOTZ benchmark, the common uniform decomposition results in subproblems whose optima cover very badly the Pareto front. The authors argue that the missing Pareto optima are hard to find. In addition, more recently, Doerr, Krejca, and Week [DKW24] showed rigorously that the MOEA/D can perform poorly on the simple OneMinMax benchmark if the number of subproblems is ill-suited to the Pareto front size and if a common mutation operator is used.

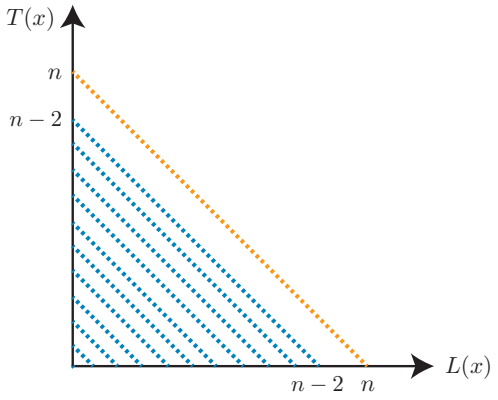
That a bad fit of the decomposition to the shape of the Pareto front can lead to drastic performance problems has been observed in several empirical works before, and several dynamic ways to adjust the decomposition exist. Due to their large number, we cannot detail them here and have to refer the reader to the survey [MYL$^+$20]. For none of them, proven performance guarantees exist.

Our contribution: We propose a simple dynamic decomposition for the bi-objective MOEA/D on discrete search spaces and analyze the performance of the resulting *self-adjusting MOEA/D* (sMOEA/D; Algorithm 2). In our dynamic decomposition scheme, some ideas present in other works can be found again, but overall we view it as a new development. In particular, since we aim at computing the full Pareto front of a discrete problem, we have to adjust the number of subproblems, whereas most previous schemes work with a static number of subproblems. Furthermore, our sMOEA/D makes use of a particularly good choice for new subproblems (see line 5 in Algorithm 2 and eq. (2)) that allows to find new useful subproblems quickly (see Lemma 7).

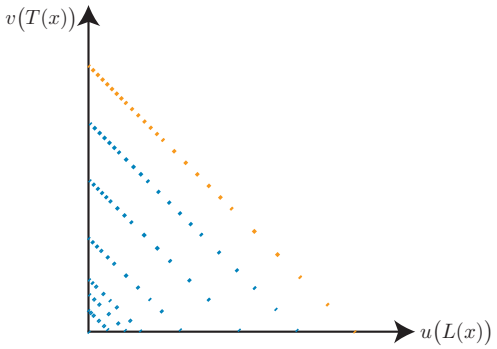
Most importantly, our sMOEA/D allows to be analyzed theoretically, resulting in the first mathematical runtime guarantees of a dynamic multi-objective evolutionary algorithm (MOEA); all previous runtime analyses for dynamic evolutionary algorithms only regarded single-objective algorithms [DD20]. Due to the drastically increased complexity rigorously analyzing a dynamic algorithm, we focus on bi-objective problems and do not consider more advanced operators, such as crossover. We note that such assumptions are standard practice for new research areas in this field and can be observed in the references we provided earlier and their references therein.

As our main result, we prove that the sMOEA/D solves *all* monotonically distorted versions of the common LOTZ benchmark of size n in expected time $O(n^4)$ (Theorem 2). This covers a huge range of different variants of the LOTZ benchmark and shows in a mathematically rigorous way that our sMOEA/D does not run into the problems detected by Li et al. citeLiZZZ16 for the classic MOEA/D. Furthermore, this runtime guarantee is independent on the distortion strength, emphasizing that the exact shape of the objective function is not important for the sMOEA/D.

Overall, our work provides a promising solution for finding a suitable decomposition for the MOEA/D on discrete problems, supported by the first mathematical runtime analysis of a dynamic MOEA.



(a) The original LOTZ benchmark. The Pareto front consists of all bit strings that start with a consecutive number of 1s and then continues with a consecutive number of 0s. All worse solutions have one or more 0s followed by one or more 1s.



(b) The monotonically distorted version of LOTZ scales both objectives of LOTZ, not necessarily in the same way, in a monotonic fashion. The Pareto front is not changed, but the values of the objective values and the gaps between them can differ drastically to the classic version.

Figure 1: A graphical sketch of the LeadingOnesTrailingZeros (LOTZ) [LTZ04] benchmark (Figure 1a) and its monotonically distorted version [LZZZ16] (Figure 1b). The details are given in Section 2. The orange points on the outmost line denote the Pareto front of the respective benchmark. The other blue points denote other objective values that are not Pareto optima.

2 Setting and Terminology

Let $\mathbb{N}$ denote the set of all natural numbers (with 0), and let $\mathbb{R}$ denote the set of all reals. For all $m, n \in \mathbb{R}$, we define $[m..n] := [m, n] \cap \mathbb{N}$ as well as $[n] := [1..n]$. For a vector $v \in \mathbb{R}^d$, we denote its components at index $i \in [d]$ by v_i.

We consider bi-objective pseudo-Boolean maximization, that is, given an integer $n \in \mathbb{N}_{\geq 1}$, we consider the maximization of a (bi-objective) *objective function* $f = (f_1, f_2) \colon \{0,1\}^n \to \mathbb{R}^2$. We call each $x \in \{0,1\}^n$ an *individual*, and $f(x)$ the *objective value of* x.

We compare objective values via the *weak domination* weak partial order $\succeq$. For all $u, v \in \mathbb{R}^2$, we say that u *weakly dominates* v (written as $u \succeq v$) if and only if $u_1 \geq v_1$ and $u_2 \geq v_2$. We extend this terminology to individuals of an objective function f and say, for all $x, y \in \{0,1\}^n$, that x *weakly dominates* y if and only if $f(x) \succeq f(y)$. We speak of *domination*, denoted by $\succ$, if we have weak domination and at least one of the inequalities is strict.

We are interested in the set of all non-dominated objective values, known as the *Pareto front (of f)*, that is, the set $F(f) := \{f(x) \mid x \in \{0,1\}^n \wedge \nexists y \in \{0,1\}^n \colon f(y) \succ f(x)\}$. We call each value in $F(f)$ a *Pareto optimum*.

Objective function. We consider a generalization of the LeadingOnesTrailingZeros (LOTZ) [LTZ04] benchmark, which is an established benchmark for the theoretical analysis of multi-objective heuristics. It aims to optimize the consecutive number of leading 1s as well the consecutive number of trailing 0s in a bit string. Formally, let $L, T \colon \{0,1\}^n \to \mathbb{R}$ be such that $L \colon x \mapsto \sum_{i \in [n]} \prod_{j \in [i]} x_j$ and $T \colon x \mapsto \sum_{i \in [n]} \prod_{j \in [n-i+1..n]} (1 - x_i)$, that is, L yields the number of leading 1s, and T the number of trailing 0s. Then LOTZ is the bi-objective function $\text{LOTZ} \colon \{0,1\}^n \to \mathbb{R}^2, x \mapsto \big(L(x), T(x)\big)$ (Figure 1a). We consider monotonically distorted versions of LOTZ, based on [LZZZ16] (Figure 1b). Let $u, v \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be monotonically increasing functions. Then we define

$$\text{LOTZ}_{u,v} \colon \{0,1\}^n \to \mathbb{R}^2, x \mapsto \big(u\big(L(x)\big), v\big(T(x)\big)\big).$$

Independent of u and v, we have $F(\text{LOTZ}_{u,v}) = \{(1^k, 0^{n-k}) \mid k \in [0..n]\}$, which has size $n + 1$. For all $x \in \{0,1\}^n$ and $k \in [0..n]$ with $x = 1^k 0^{n-k}$, we call x a $(k, n-k)$*-individual*.

Algorithm 1: The *decomposition-based multi-objective evolutionary algorithm* (MOEA/D) [ZL07] with the Chebyshev approach, maximizing a bi-objective function of size $n \in \mathbb{N}_{\geq 1}$.

Input: A bi-objective function $f : \{0,1\}^n \to \mathbb{R}^2$, a termination criterion, a mutation operator $\mathrm{mut}\colon \{0,1\}^n \to \{0,1\}^n$, a decomposition number $N \in N_{\geq 2}$, a weight vector $w \in [0,1]^N$, as well as a neighborhood size $T \in [N]$.

Output: A set P of incomparable best-so-far objective values.

Initialization: set $P = \emptyset$; for each $i \in [N]$, choose x_{w_i} uniformly at random from $\{0,1\}^n$; set $z_1^* = \max_{i \in [N]} f_1(x_{w_i})$, $z_2^* = \max_{i \in [N]} f_2(x_{w_i})$, and iteratively add $f(x_{w_i})$ to P if there is no $j \in [i-1]$ such that x_i is weakly dominated by x_j; for each $i \in [N]$, let B_i denote the indices of the T weights closest to w_i;

while stopping criterion is not met **do**

  for each subproblem $i \in [N]$ **do**

    Generate offspring: $y \leftarrow \mathrm{mut}(x_{w_i})$;

    Update z^*: set $z_1^* \leftarrow \max(z_1^*, f_1(y))$, $z_2^* \leftarrow \max(z_2^*, f_2(y))$;

    Update neighbors: for all $j \in B_i$, if $g_{w_j}(y, z^*) \leq g_{w_j}(x_{w_j}, z^*)$, then $x_{w_j} \leftarrow y$;

    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;

return P;

Throughout this section, we let $n \in \mathbb{N}_{\geq 1}$ always denote the *problem size* of $\mathrm{LOTZ}_{u,v}$, that is, the length of the bit strings, and we assume that n is given implicitly. Similarly, we assume that u and v are arbitrary and implicitly given. Moreover, big-O notation always refers to asymptotics in n.

Fitness level method. In parts of our analysis, we make use of the fact that the sMOEA/D can only improve on certain values, such as the fitness of a best-so-far individual for a subproblem. In order to estimate the expected waiting time until sufficiently many improvements occurred, we sum the expected times for each improvement. As each trial for an improvement is independent of the other trials, the probability for an improvement follows a geometric law. Bounding the improvement probability of such a law from below results in an upper bound of its expected waiting time, allowing us to bound the overall waiting time from above.

In the field of evolutionary computing, this argument is called *fitness level method*, and it was introduced by [Weg01]. The method works for any real-valued Markov chain $(X_t)_{t \in \mathbb{N}}$ over a finite space S when given a potential function $h\colon S \to \mathbb{R}$ and a target value $v \in \mathbb{R}$. We aim to bound the expected time $T = \inf\{t \in \mathbb{N} \mid h(X_t) = v\}$. It is important that the potential cannot decrease, that is, for all $t \in \mathbb{N}$, we have $h(X_t) \leq h(X_{t+1})$. We call the range of h the *fitness levels* (of h). Formally, we have the following theorem, which is a reformulation of a version by [Sud13].

Theorem 1 (Rephrased from [Sud13, Theorem 2])**.** *Let $(X_t)_{t \in \mathbb{N}}$ be a real-valued Markov chain over a finite space S, let $h\colon S \to \mathbb{R}$ be a potential function, let $v \in \mathbb{R}$ be a target value, and let $T = \inf\{t \in \mathbb{N} \mid X_t = v\}$. Furthermore, for all $i \in \{j \in \mathbb{R} \mid \exists s \in S\colon h(s) = j\} \setminus \{v\} =: I$, let q_i denote a lower bound on the probability of X transitioning from a state with potential i to a state with a strictly larger potential. Then $\mathrm{E}[T] \leq \sum_{i \in I} \frac{1}{q_i}$.*

3 The MOEA/D

The *decomposition-based multi-objective evolutionary algorithm* (MOEA/D) by Zhang and Li [ZL07] decomposes a given bi-objective function into several single-objective subproblems, each of which is optimized in a co-evolutionary manner, that is, they are optimized in parallel and exchange solutions. The decomposition is determined by 1) the subproblem type (the *approach*) 2) the number of subproblems $N \in \mathbb{N}_{\geq 2}$ (the *decomposition number*), and 3) the *weight vector* $w \in [0,1]^N$ that weights the objectives per subproblem. This results in a flexible algorithm but also in a great responsibility of the user, as different parameter choices can result in drastic performance differences.

First, we explain the classic MOEA/D (Algorithm 1) for the very commonly used [XXM20] Chebyshev approach. Afterward, we present our self-adjusting variant of the MOEA/D (Algorithm 2), which removes the choices of N and w and adjusts them on the fly.

3.1 Classic MOEA/D With the Chebyshev Approach

Given a decomposition number $N \in \mathbb{N}_{\geq 2}$ and a weight vector $w \in [0,1]^N$, the MOEA/D optimizes N single-objective subproblems $\{g_{w_i}\}_{i \in [N]}$, each of which is based on the original bi-objective function $f\colon \{0,1\}^n \to \mathbb{R}$, the weight vector w, and a *reference point* $z^* \in \mathbb{R}^2$, which reflects the currently best known values of both objectives. A very common way of modeling the subproblems, known as the *Chebyshev approach*, is to define for all $i \in [N]$ that

$$g_{w_i}\colon \{0,1\}^n \times \mathbb{R}^2 \to \mathbb{R}, \qquad (x, z^*) \mapsto \max\{w_i \cdot |z_1^* - f_1(x)|, (1-w_i) \cdot |z_2^* - f_2(x)|\}. \quad (1)$$

That is, subproblem $i \in [N]$ measures the w_i-weighted maximum difference between an individual and the reference point. Each subproblem is subject to *minimization*.

We call the reference point z^* *optimal* if and only if its components are the maximum values of each of the objectives in the Pareto front. For the optimal reference point and for each subproblem g, we call an individual *g-optimal* if and only if its g-function value is an optimum of g.

The MOEA/D aims to find the Pareto optima (of f) by searching for the g-optima and maintaining an *archive* of all currently seen objective values that are not weakly dominated. To this end, the algorithm optimizes each subproblem following an evolutionary search. That is, it maintains for each $i \in N$ a *best-so-far* individual x_{w_i} that has the best g_{w_i}-value among all so-far seen individuals.

Iteratively, the MOEA/D creates new individuals, based on x_{w_i}. These new individuals are called *offspring*, and they are generated via mutation. The input individuals for either operator are called *parent(s)*. In this paper, we consider two common mutation operators. *Standard bit mutation* creates an offspring $y \in \{0,1\}^n$ from $x \in \{0,1\}^n$ by choosing for each bit independently to flip it with probability $\frac{1}{n}$, that is, for all $k \in [n]$, we have $y_k = 1 - x_k$ with probability $\frac{1}{n}$ as well as $y_k = x_k$ otherwise. *1-bit mutation* creates an offspring $y \in \{0,1\}^n$ from $x \in \{0,1\}^n$ by choosing exactly one position $k \in [n]$ uniformly at random, flipping the bit at position k, and copying all other bits. That is, for all $k \in [n]$, we have $y_k = 1 - x_k$ as well as $y_k = x_k$ otherwise.

We note that the MOEA/D classically also applies an operation known as *crossover* in line 4 before performing mutation on the crossover result. Crossover takes two individuals and combines them into a new individual (in a way that needs to be specified by the user, similar to mutation). The parents for subproblem $i \in [N]$ are selected uniformly at random from the indices in B_i. Since our theoretical analysis focuses on mutation, we omit crossover in this paper, noting that it can be added to our self-adjusting variant analogously to how it is added to the classic MOEA/D.

An offspring y is compared to all best-so-far individuals within a certain neighborhood, and y replaces each such individual x with an equal or worse fitness. During this step, the reference point z^* is also updated, then the archive. We note that while the neighborhood size has a large range of possible values, it was recently shown that already a minimal size of 1, that is, a communication just among direct neighbors, can lead to best-possible asymptotic runtimes [DEKM26]. This suggests that the choice of this parameter may be less important as long as subproblems can exchange solutions at all.

3.2 Self-Adjusting MOEA/D

Due to the difficult and important decision of choosing a good decomposition when applying the MOEA/D, we propose a variant that adjusts the number of subproblems as well as the respective weights automatically during the optimization process. We call this variant the *self-adjusting MOEA/D* (sMOEA/D; Algorithm 2). We note that while we present the sMOEA/D with the Chebyshev approach (eq. (1)), it is applicable to any approach, although this may affect the complexity of some of the involved operations (such as the computation performed in line 5).

The sMOEA/D operates mainly in the same way as the classic MOEA/D. That is, it creates offspring, selects best-so-far individuals, and updates the archive in an almost identical way. The main difference is how the sMOEA/D chooses to adjust its number of subproblems and their weights. For simplicity, we assume the full neighborhood, that is, an improvement is communicated to all subproblems.

Algorithm 2: The self-adjusting decomposition-based multi-objective evolutionary algorithm (sMOEA/D), maximizing a pseudo-Boolean bi-objective function of size $n \in \mathbb{N}_{\geq 1}$.

Input: A bi-objective function $f\colon \{0,1\}^n \to \mathbb{R}^2$, a mutation operator $\mathrm{mut}\colon \{0,1\}^n \to \{0,1\}^n$, and a stopping criterion.

Output: A set P of incomparable best-so-far objective values.

1 *Initialization*: set $W = \{0,1\}$ and $P = \emptyset$; choose x_0 and x_1 uniformly at random from $\{0,1\}^n$; set $z_1^* = \max(f_1(x_0), f_1(x_1)), z_2^* = \max(f_2(x_0), f_2(x_1))$; add $f(x_0)$ and $f(x_1)$ to P and remove one of them from P if one is dominated by the other; for each $w \in \{0,1\}$, if $g_w(x_{1-w}, z^*) < g_w(x_w, z^*)$, replace x_w by x_{1-w}.

2 **while** stopping criterion is not met **do**

3 **while** there are consecutive weights $w_1, w_2, w_3 \in W$ with $f(x_{w_1}) = f(x_{w_2}) = f(x_{w_3})$ **do** remove w_2 from W;

4 **for** all consecutive weights $w_1, w_2 \in W$ with $f(x_{w_1}) \neq f(x_{w_2})$ **do**

5 find w such that $g_w(x_{w_1}, z^*) = g_w(x_{w_2}, z^*)$ (see eq. (2));

6 add w to W and set $x_w = \arg\min_{x \in P} g_w(x, z^*)$ once we are done with this loop;

7 **for** each subproblem $w \in W$ **do**

8 *Generate offspring*: $y \leftarrow \mathrm{mut}(x_w)$;

9 *Update* z^*: set $z_1^* \leftarrow \max(z_1^*, f_1(y))$, $z_2^* \leftarrow \max(z_2^*, f_2(y))$;

10 *Update* x_v: for all $v \in W$, if $g_v(y, z^*) \leq g_v(x_v, z^*)$, then $x_v \leftarrow y$;

11 *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;

12 return P;

The sMOEA/D maintains a set W of weights, each in $[0,1]$, with the weights 0 and 1 being initially in W and never leaving it. In the following, we assume that W is ordered increasingly and that its elements are indexed from 1 to $|W|$.[1] Whenever the algorithm finds a pair of *consecutive* weights, that is, for $i \in [|W| - 1]$, a pair (w_i, w_{i+1}), such that the objective values of the respective best-so-far individuals disagree, that is, such that $f(x_{w_i}) \neq f(x_{w_{i+1}})$, then the sMOEA/D adds another weight, aiming to find further Pareto optima of f. Moreover, if the objective values of the best-so-far individuals of more than two consecutive weights coincide, the algorithm removes all but the two extreme involved weights, in order to reduce the cardinality of W.

If there is an $i \in [|W| - 1]$ with $f(x_{w_i}) \neq f(x_{w_{i+1}})$, the sMOEA/D adds the weight $w \in [0,1]$ such that $g_w(x_{w_i}, z^*) = g_w(x_{w_{i+1}}, z^*)$, since this is an extremal point where the two g-functions coincide for x_{w_i} and $x_{w_{i+1}}$, making it a promising choice, given that the objective value of x_{w_i} and of $x_{w_{i+1}}$ differs. For the Chebyshev approach, we compute w directly as follows (and derive it below) as

$$w = \frac{|z_2^* - f_2(x_{w_{i+1}})|}{|z_1^* - f_1(x_{w_i})| + |z_2^* - f_2(x_{w_{i+1}})|}. \tag{2}$$

To see why eq. (2) holds, we first introduce for all $z \in \mathbb{R}$ and all $i \in \{1,2\}$ the notation $d_i\colon \mathbb{R}^2 \to \mathbb{R}, x \mapsto |z_i - f_i(x)|$. Thus, it holds for all $w \in [0,1]$ and all $x \in \mathbb{R}^2$ that $g_w(x,z) = \max\{w \cdot d_1(x), (1-w) \cdot d_2(x)\}$. Note that, for all $w \in [0,1]$, we have that the linear functions of the form $w \mapsto w \cdot d_1(x)$ all meet in $(0,0)$, and the linear functions of the form $w \mapsto (1-w) \cdot d_2(x)$ all meet in $(1,0)$. Last, for all $x \in \mathbb{R}^2$, let $w_x^* = d_2(x)/\big(d_1(x) + d_2(x)\big)$ denote the minimum of the function $w \mapsto g_w(x,z)$. We argue in the following that $g_w(x_{w_i}, z^*)$ and $g_w(x_{w_{i+1}}, z^*)$ meet at a point $w \in [0,1]$ such that $w \geq w^*_{x_{w_i}}$ and $w \leq w^*_{x_{w_{i+1}}}$, noting that eq. (2) is the intersection of $w \mapsto w \cdot d_1(x_{w_i})$ and $w \mapsto (1-w) \cdot d_2(x_{w_{i+1}})$, at $w = d_2(x_{w_{i+1}})/\big(d_1(x_{w_i}) + d_2(x_{w_{i+1}})\big)$.

An important observation is that, by line 10, we have $g_{w_i}(x_{w_i}, z^*) \leq g_{w_i}(x_{w_{i+1}}, z^*)$ as well as $g_{w_{i+1}}(x_{w_i}, z^*) \geq g_{w_{i+1}}(x_{w_{i+1}}, z^*)$. Thus, it cannot be that $w_i < w^*_{x_{w_i}}$ and $w_{i+1} > w^*_{x_{w_{i+1}}}$ at the same time, as x_{w_i} is otherwise replaced by $x_{w_{i+1}}$ or the other way around, contradicting that $x_{w_i} \neq x_{w_{i+1}}$, due to $f(x_{w_i}) \neq f(x_{w_{i+1}})$. Hence, the intersection occurs if $w_i \geq w^*_{x_{w_i}}$ or $w_{i+1} \leq w^*_{x_{w_{i+1}}}$. If $w_i \geq w^*_{x_{w_i}}$, then the intersection occurs for a $w \in [w_i, 1]$ with $w \leq w^*_{x_{w_{i+1}}}$, for an analogous

[1] This indexation is *not* reflected in Algorithm 2, as it is merely useful for explaining the algorithmic idea.

argument as before. Similarly, if $w_{i+1} \leq w^*_{x_{w_{i+1}}}$, then the intersection occurs for a $w \in [0, w_{i+1}]$ with $w \geq w^*_{x_{w_i}}$, concluding how to derive eq. (2).

Computational complexity. We note that the sMOEA/D remains very similar to the classic MOEA/D in terms of computational complexity. The classic MOEA/D looks for N improvements in each iteration, whereas the sMOEA/D looks for $|W|$ improvements each iteration, noting that $|W|$ can change during the run. Hence, the sMOEA/D only performs more computations if $|W| > N$.

Setting. We consider the sMOEA/D both with standard bit mutation as well as with 1-bit mutation as its mutation operator. Note that the probability to flip exactly one specific bit at position $i \in [n]$ is $\Omega(\frac{1}{n})$ for either operator, which follows from the definition of 1-bit mutation and follows for standard bit mutation by noting that the probability is $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$.

Runtime. We define the *runtime* of an MOEA/D variant A on a bi-objective function f to be the (random) number of objective-function evaluations until the archive P of A *covers* the Pareto front of f, that is, until $P \supseteq F(f)$, for the first time at the end of an iteration of the main loop. We note that it is typical to only count the function evaluations, as these are usually the most costly operations of a randomized search heuristic [DN20]. We assume that each created individual is evaluated exactly once, namely upon creation.

Changes for continuous problems. For continuous problems, the equality checks are not reasonable, as it is unlikely that equality occurs with uncountably many objective values. Hence, we propose in such settings to define a tolerance $\varepsilon \in \mathbb{R}_{>0}$ upfront and to treat all numbers with an absolute distance of at most ε as equal.

4 Mathematical Runtime Analysis on Generalized LOTZ Functions

Our main result is that the sMOEA/D (Algorithm 2) computes the complete Pareto front of *any* monotonically distorted LOTZ function in an expected number of $O(n^4)$ objective-function evaluations.

Theorem 2. *The sMOEA/D optimizes* $\mathrm{LOTZ}_{u,v}$ *in* $O(n^4)$ *expected function evaluations.*

This bound is only worse by a factor of n than the *best* known bound of the classic MOEA/D on standard LOTZ for a perfect decomposition, that is, one where the solutions to all subproblems cover the entire Pareto front. For some monotonically distorted versions of LOTZ, it is known that the common decomposition of the classic MOEA/D with evenly spread weights does not result in a perfect decomposition anymore, which is claimed as detrimental for the algorithm [LZZZ16]. The reason is that the uniform decomposition does not need to match the structure of the Pareto front of the distorted function very well. For example, consider Figure 1b, where the distance among different Pareto optima is far from uniform. Depending on the nature of the distortion, some Pareto optima may not be closest to any subproblem. The sMOEA/D does not have this problem, due to its dynamic decomposition, showing that it is far more robust.

Before we prove Theorem 2, we prove various useful statements that we utilize for the final proof.

Lemma 3 shows that there is a partition of the weights of the sMOEA/D such that each g-optimal solution is also a Pareto optimum of $\mathrm{LOTZ}_{u,v}$, given that the reference point is optimal, that is, $\big(u(n), v(n)\big)$. We note that the boundaries of the partition in Lemma 3 match eq. (2) (as well as the trivial weights 0 and 1 that are always part of the algorithm), which shows that the sMOEA/D is, in theory, capable of finding weights such that all Pareto optima are discerned.

Lemma 3. *Consider* $\mathrm{LOTZ}_{u,v}$*, with monotonically increasing functions* u *and* v *as well as eq.* (1) *with* $z^* = \big(u(n), v(n)\big)$*. Let* $w^*_0 \coloneqq 0$*, for all* $i \in [n]$*, let* $w^*_i \coloneqq \frac{v(n)-v(n-i)}{u(n)-u(i-1)+v(n)-v(n-i)}$*, and let* $w^*_{n+1} \coloneqq 1$*. Then* $\{[w^*_0, w^*_1]\} \cup \{(w^*_i, w^*_{i+1}]\}_{i\in[n]}$ *is a partition of* $[0,1]$ *such that for all* $i \in [0..n]$ *and all* $w \in (w^*_i, w^*_{i+1}]$ *(or all* $w \in [w^*_0, w^*_1]$ *in the case of* $i = 0$*), we have that each* $(i, n-i)$*-individual is the sole optimum of* g_w*.*

Proof. Let $i \in [0..n]$, let $w \in (w^*_i, w^*_{i+1}]$, but we exclude the cases $w \in \{0,1\}$ at the moment and prove them at the very end. Moreover, let $x \in \{0,1\}^n$ be the (unique) $(i, n-i)$-individual,

and let $y \in \{0,1\}^n \setminus \{x\}$. Ultimately, we show via a case distinction over the structure of y that $g_w(x) < g_w(y)$, which proves the claim. Recall that L and T denote, respectively, the number of leading ones and trailing zeros of an individual.

Before we conduct our case distinction, we define an important value that helps us reason for all $w' \in [0,1]$ more easily about $g_{w'}$. Following eq. (2), for all $z \in \{0,1\}^n$, we define $b_z \coloneqq \frac{v(n)-v(T(z))}{u(n)-u(L(z))+v(n)-v(T(z))}$ as the *breaking point* for z at which the maximum of $g_{w'}$ is taken with respect to the term over the first objective, that is, for all $w' \in [0,1]$, we have

$$g_{w'}(z) = \begin{cases} w'\big(u(n)-u(L(z))\big) & \text{if } w' > b_z, \\ (1-w')\big(v(n)-v(T(z))\big) & \text{otherwise.} \end{cases}$$

Case 1: y has strictly fewer leading ones and strictly fewer trailing zeros than x. Let $j = L(y) < i$ and $k < T(y) \leq n-i$. Each of the two terms of the $\max$ in $g_w(y)$ is strictly larger than the corresponding term in $g_w(x)$. That is, owing to the monotonicity of u and v and to the bound on j and k, we have $w\big(u(n)-u(i)\big) < w\big(u(n)-u(j)\big)$ and $(1-w)\big(v(n)-v(n-i)\big) < (1-w)\big(v(n)-v(k)\big)$. Hence, regardless of w, we have $g_w(y) > g_w(x)$.

Case 1': y has strictly more leading ones and strictly more trailing zeros than x. This is the same case as Case 1 when swapping x and y.

Case 2: y has strictly fewer leading ones than x and at least as many trailing zeros as x. Let $j = L(y) < i$ and $k = T(y) \geq n-i$. We show that $b_y \leq w_i^*$ via a case distinction with respect to whether the denominator of b_y is greater than that of w_i^*. If $u(n)-u(j)+v(n)-v(k) \geq u(n)-u(i-1)+v(n)-v(n-i)$, then by the monotonicity of v and by $k \geq n-i$, it follows that the numerator of b_y is at most that of w_i^*, that is, $v(n)-v(k) \leq v(n)-v(n-i)$.

If $u(n)-u(j)+v(n)-v(k) < u(n)-u(i-1)+v(n)-v(n-i)$, then by the monotonicty of u and by $j \leq i-1$, we have $b_y = 1 - \frac{u(n)-u(j)}{u(n)-u(j)+v(n)-v(k)} < 1 - \frac{u(n)-u(i-1)}{u(n)-u(i-1)+v(n)-v(n-i)} = w_i^*$.

We continue with a case distinction with respect to w, recalling that we only consider values $w > w_i^*$. Due to $w_i^* \geq b_y$, this implies that $w > b_y$ in the following and thus that $g_w(y) = w\big(u(n)-u(j)\big)$.

If $w > b_x$, then $g_w(x) = w\big(u(n)-u(i)\big)$. Hence, owing to the monotonicity of u and to $i > j$, it follows that $g_w(y) > g_w(x)$.

Last, if $w \leq b_x$, then $g_w(x) = (1-w)\big(v(n)-v(n-i)\big)$. Note that $g_w(y) > g_w(x)$ holds if and only if $w > \frac{v(n)-v(n-i)}{u(n)-u(j)+v(n)-v(n-i)} =: W$. Since $w_i^* \geq W$ and since $w > w_i^*$ by assumption, we obtain $w > W$. This concludes Case 2.

Case 2': y has strictly more leading ones than x and at most as many trailing zeros as x. This is the same case as Case 2 when swapping x and y.

Case 3: y has exactly as many leading ones as x. Since we assume that $x \neq y$ and since the last leading one of x must be followed by at least one zero and thereafter at least one one in order for y to be different from x, it follows that $k \coloneqq T(y) < n-i-1$.

We show that $b_y > w_{i+1}^*$ via a case distinction with respect to whether the denominator of b_y is greater than that of w_i^*. If $u(n)-u(i)+v(n)-v(k) \leq u(n)-u(i)+v(n)-v(n-i-1)$, then by the monotonicity of v and by $k \leq n-i-2$, it follows that the numerator of b_y is greater than that of w_{i+1}^*, that is, $v(n)-v(k) > v(n)-v(n-i-1)$.

If $u(n)-u(i)+v(n)-v(k) > u(n)-u(i)+v(n)-v(n-i-1)$, then $b_y = 1 - \frac{u(n)-u(i)}{u(n)-u(i)+v(n)-v(k)} > 1 - \frac{u(n)-u(i)}{u(n)-u(i)+v(n)-v(n-i-1)} = w_{i+1}^*$, as claimed.

We continue with a case distinction with respect to w, recalling that we only consider values $w \leq w_{i+1}^*$. Due to $w_{i+1}^* < b_y$, we have $w < b_y$ in the following and thus $g_w(y) = (1-w)\big(v(n)-v(k)\big)$.

If $w \leq b_x$, then $g_w(x) = (1-w)\big(v(n)-v(n-i)\big)$. Hence, owing to the monotonicity of v and to $n-i > k$, it follows that $g_w(y) > g_w(x)$.

Last, if $w > b_x$, then $g_w(x) = w\big(u(n)-u(i)\big)$. Note that $g_w(y) > g_w(x)$ holds if and only if $w < \frac{v(n)-v(k)}{u(n)-u(i)+v(n)-v(k)} = b_y$, which is true, as we discussed just above. This concludes Case 3.

The weights $w \in \{0, 1\}$. We start with $w = 0$ here, which implies that we consider x to be the $(0, n)$-individual. In this case, regardless of $z \in \{0, 1\}^n$, we have $g_w(z) = v(n) - v(T(z))$. Due to the monotonicity of v, this value is 0 if and only if $z = x$.

For $w = 1$, we get an analogous result for x being the $(n, 0)$-individual, as, for all $z \in \{0, 1\}^n$,we have that g_w simplifies to $u(n) - u(L(z))$, which is 0 if and only if $z = x$, concluding the proof. □

Lemma 4 bounds the maximum number of subproblems possible, providing us with an upper bound of the cost of each iteration.

Lemma 4. *Consider the sMOEA/D optimizing* $\text{LOTZ}_{u,v}$*. In each iteration, the number of subproblems is at most* $3n + 2$*.*

Proof. After the deletion step (line 3 of Algorithm 2), each $f(x_w)$ appears at most twice, since we delete one occurrence of $f(x_w)$ every time we see three consecutive ones.

Note that the set $\{f(x_w) \mid w \in W\}$ has no element that weakly dominates another because at each time, $f(x_w)$ is the best-possible value from the archive for each $w \in W$, which implies it cannot be dominated by another $f(x_{w'})$ because $f(x_{w'})$ is in the archive. Because $f_1(x)$ can take at most $n + 1$ values, we have $|\{f(x_w) \mid w \in W\}| \leq n + 1$. Hence after the deletion phase, $|W| \leq 2(n + 1)$.

In addition, there are at most n weights that are added to W because there are at most n consecutive weights in W that have different values of $f(x_w)$. This gives a population size of at most $3n + 2$ after adding the weights. □

We split the analysis of our main result into two phases. Phase 1 considers the number of iterations until the reference point is optimal. Phase 2 starts immediately once phase 1 ends and lasts until the archive covers the entire Pareto front.

Phase 1. Since solutions are exchanged among all subproblems, the g-optima for weights 0 and 1 are quickly found, which implies that the reference point is optimized.

Lemma 5. *Consider the sMOEA/D optimizing* $\text{LOTZ}_{u,v}$*. The optimal reference point is found in* $O(n^2)$ *expected iterations.*

We prove Lemma 5 by showing that the individuals corresponding to weights 0 and 1 optimize functions g_0 and g_1 until reaching their respective optimal individuals 1^n and 0^n. We measure progress, respectively, in the first or in the second objective, which is monotone. In expectation, improving such an objective takes $O(n)$ iterations, since one specific bit needs to flip. The result follows since we require at most n improvements. Details are deferred to the appendix.

Phase 2. In our analysis, we define fitness levels, which show how far away the algorithm is from covering the Pareto front. The higher the fitness level, the closer the algorithm is to the Pareto front. We then show that the fitness levels increase with sufficiently high probability.

Lemma 6 shows that as long as the Pareto front is not fully covered, there is always a weight $w \in W$ such that x_w is not optimal for g_w, leaving room for a fitness level improvement.

Lemma 6. *Consider the sMOEA/D optimizing* $\text{LOTZ}_{u,v}$*, and consider an iteration in which the Pareto front is not covered. Then before the mutation step (line 8 in Algorithm 2), there exists an individual* x_w *that is not* g_w*-optimal.*

We prove Lemma 6 via a case distinction. If a non-optimal x_w exists at the beginning of the iteration, weight w is only deleted if other individuals have matching objectives, which means that at least one other individual is non-optimal, by Lemma 3, and remains. If all current individuals are optimal, since the Pareto front is not covered, there exist at least two neighboring weights with differing objective. Hence, a new weight with a non-optimal individual is added. Details are deferred to the appendix.

Using a non-optimal individual from Lemma 6, the current fitness level quickly increases via mutation.

Lemma 7. *Consider the sMOEA/D optimizing* $\text{LOTZ}_{u,v}$*, and an iteration such that the Pareto front is not covered yet. Then the current fitness level increases by at least* 1 *in* $O(n)$ *expected iterations. Moreover, when starting with an optimal reference point, then the sMOEA/D finds the entire Pareto front in* $O(n^3)$ *expected iterations.*

Proof. We define the fitness level at the beginning of an iteration of the sMOEA/D as the number $k \in \mathbb{N}$ of values of the objective function f that are weakly dominated by the archive. Note that k is always positive and at most $(n+1)^2$ as the pair $(L(x), T(x))$ can take at most $(n+1)^2$ values, which is also an upper bound for $f(x)$. Moreover, k cannot decrease during the run of the algorithm. Last, if the fitness level is maximal, then the Pareto front has been found.

We show that it takes $O(n)$ expected iterations to increase each fitness level, corresponding to $O(n^3)$ expected iterations to reach a maximal fitness level, at which point the whole Pareto is covered.

If the Pareto front is not fully covered, by Lemma 6, at every mutation phase, there is at least one w such that x_w is not optimal for g_w. This means that with probability $\Omega(1/n)$, individual x_w is mutated into y such that $g_w(y) < g_w(x_w)$, as it suffices to either increase the number of leading ones or the number of trailing zeros of x_w by one (which can be done with probability $\Omega(1/n)$). Furthermore, y cannot be weakly dominated by an individual in the archive because otherwise we would have x_w equal to something better from the archive, which is impossible. Hence, we add y to the archive, and it weakly dominates itself, which was not the case before. Now if X is the random variable giving the number of iterations before the fitness level increases, X follows a geometric distribution of parameter $p = \Omega(1/n)$, resulting in $E[X] = O(n)$. Consequently, the number of weakly dominated elements increases by at least one in $O(n)$ expected iterations of the algorithm as required. □

Combining both phases. Combining the results of both phases and accounting for the maximum cost of an iteration by Lemma 4 yields immediately our main result.

Proof of Theorem 2. By Lemma 5, the first phase runs in $O(n^2)$ iterations, and from the population size being $O(n)$ (Lemma 4), we have a runtime of $O(n^3)$ function evaluations by multiplying the number of iterations by the upper bound on the number of individuals mutated per iteration.

By Lemma 7, the second phase runs in $O(n^3)$ iterations, and from the population size being $O(n)$ (Lemma 4), we have a runtime of $O(n^4)$ function evaluations.

We obtain a total runtime of $O(n^4)$ function evaluations by summing the runtime of both phases. □

5 Conclusion

We developed a new way how the MOEA/D can automatically adjust the number of subproblems and the weight vectors defining them. Compared to previous such algorithms, ours is simple enough to admit proven performance guarantees (which are, in fact, the first proven performance guarantees for a dynamic multi-objective evolutionary algorithm).

We proved that our sMOEA/D computes the Pareto front of any monotonically distorted LOTZ functions in an expected number of $O(n^4)$ function evaluations, clearly overcoming the difficulties detected in [LZZZ16].

In terms of generalization, we believe that a decomposition similar to the one presented in Lemma 3 is also possible for other established theory benchmarks, such as ONEMINMAX [GL10] and ONEJUMPZEROJUMP [ZD23]. Moreover, we ran experiments for 0/1-KNAPSACK instances, which we report in Section B.1. In these experiments, we compare the sMOEA/D to the classic MOEA/D with tuned parameters, finding that the former is at least as good if not better than the latter while requiring almost no parameter tuning. This suggests that the idea of the sMOEA/D translates well to settings that go beyond the theoretical guarantees present in this work.

In this first work, we restricted ourselves to bi-objective pseudo-Boolean optimization. Natural next steps, in order of increasing need for new ideas, are (1) the use of our algorithm for discrete problems on search spaces other than bit strings, (2) the extension of our algorithm to more than two objectives (this in particular needs the definition of a suitable way to insert new subproblems), and (3) possibly an extension to continuous problems, which usually operate with a fixed number of subproblems.

Acknowledgments

Martin S. Krejca is supported by the French National Agency for Research (ANR) via the JCJC grant titled MultiEDA (ANR-25-CE23-2418-01). This research also benefited from the support of the

FMJH Program Gaspard Monge for optimization and operations research and their interactions with data science.

References

[BNE07] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181:1653–1669, 2007.

[BZLQ23] Chao Bian, Yawen Zhou, Miqing Li, and Chao Qian. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5513–5521. ijcai.org, 2023.

[CDH+23] Sacha Cerf, Benjamin Doerr, Benjamin Hebras, Jakob Kahane, and Simon Wietheger. The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5522–5530. ijcai.org, 2023.

[DD20] Benjamin Doerr and Carola Doerr. Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 271–321. Springer, 2020. Also available at `https://arxiv.org/abs/1804.05650`.

[DEKM26] Benjamin Doerr, Moaad Elmoutassim, Martin S. Krejca, and Guy Mushiata Kibulu. Understanding the co-evolutionary solution exchange of the MOEA/D: Already the smallest communication radius gives the best-possible speed-up. In *Advances in Neural Information Processing Systems, NeurIPS 2026*, 2026. To appear.

[DJ14] Kalyanmoy Deb and Himanshu Jain. 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, 2014.

[DKW24] Benjamin Doerr, Martin S. Krejca, and Noé Weeks. Proven runtime guarantees for how the MOEA/D computes the Pareto front from the subproblem solutions. In *Parallel Problem Solving From Nature, PPSN 2024, Part III*, pages 197–212. Springer, 2024.

[DN20] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020. Also available at `http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html`.

[DN21] Anh Viet Do and Frank Neumann. Pareto optimization for subset selection with dynamic partition matroid constraints. In *Conference on Artificial Intelligence, AAAI 2021*, pages 12284–12292. AAAI Press, 2021.

[DNNS23] Anh Viet Do, Aneta Neumann, Frank Neumann, and Andrew M. Sutton. Rigorous runtime analysis of MOEA/D for solving multi-objective minimum weight base problems. In *Advances in Neural Information Processing Systems, NeurIPS 2023*, pages 36434–36448, 2023.

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[Gie03] Oliver Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2003*, pages 1918–1925. IEEE, 2003.

[GL10] Oliver Giel and Per Kristian Lehre. On the effect of populations in evolutionary multi-objective optimisation. *Evolutionary Computation*, 18:335–356, 2010.

[HJH19] Yicun Hua, Yaochu Jin, and Kuangrong Hao. A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular Pareto fronts. *IEEE Transactions on Cybernetics*, 49(7):2758–2770, 2019.

[HZ20] Zhengxin Huang and Yuren Zhou. Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework. In *Conference on Artificial Intelligence, AAAI 2020*, pages 2359–2366. AAAI Press, 2020.

[HZC+21] Zhengxin Huang, Yuren Zhou, Zefeng Chen, Xiaoyu He, Xinsheng Lai, and Xiaoyun Xia. Running time analysis of MOEA/D on pseudo-Boolean functions. *IEEE Transactions on Cybernetics*, 51:5130–5141, 2021.

[HZCH19] Zhengxin Huang, Yuren Zhou, Zefeng Chen, and Xiaoyu He. Running time analysis of MOEA/D with crossover on discrete optimization problem. In *Conference on Artificial Intelligence, AAAI 2019*, pages 2296–2303. AAAI Press, 2019.

[HZLL21] Zhengxin Huang, Yuren Zhou, Chuan Luo, and Qingwei Lin. A runtime analysis of typical decomposition approaches in MOEA/D framework for many-objective optimization problems. In *International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1682–1688, 2021.

[ISMN17] Hisao Ishibuchi, Yu Setoguchi, Hiroyuki Masuda, and Yusuke Nojima. Performance of decomposition-based many-objective algorithms strongly depends on Pareto front shapes. *IEEE Transactions on Evolutionary Computation*, 21(2):169–190, 2017.

[LL11] Hui Li and Dario Landa-Silva. An adaptive evolutionary multi-objective approach based on simulated annealing. *Evolutionary Computation*, 19(4):561–595, 2011.

[LTZ04] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8:170–182, 2004.

[LY20] Miqing Li and Xin Yao. What weights work for you? Adapting weights for any Pareto front shape in decomposition-based evolutionary multiobjective optimisation. *Evolutionary Computation*, 28(2):227–253, 2020.

[LZZZ16] Yuan-Long Li, Yu-Ren Zhou, Zhi-Hui Zhan, and Jun Zhang. A primary theoretical study on decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 20:563–576, 2016.

[MYL+20] Xiaoliang Ma, Yanan Yu, Xiaodong Li, Yutao Qi, and Zexuan Zhu. A survey of weight vector adjustment methods for decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 24:634–649, 2020.

[ODNS24] Andre Opris, Duc Cuong Dang, Frank Neumann, and Dirk Sudholt. Runtime analyses of NSGA-III on many-objective problems. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, pages 1596–1604. ACM, 2024.

[QML+14] Yutao Qi, Xiaoliang Ma, Fang Liu, Licheng Jiao, Jianyong Sun, and Jianshe Wu. MOEA/D with adaptive weight adjustment. *Evolutionary Computation*, 22(2):231–264, 2014.

[QYZ15] Chao Qian, Yang Yu, and Zhi-Hua Zhou. On constrained Boolean Pareto optimization. In *International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 389–395. AAAI Press, 2015.

[RBLQ24] Shengjie Ren, Chao Bian, Miqing Li, and Chao Qian. A first running time analysis of the Strength Pareto Evolutionary Algorithm 2 (SPEA2). In *Parallel Problem Solving from Nature, PPSN 2024, Part III*, pages 295–312. Springer, 2024.

[Sud13] Dirk Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435, 2013.

[Thi03] Dirk Thierens. Convergence time analysis for the multi-objective counting ones problem. In *Evolutionary Multi-Criterion Optimization, EMO 2003*, pages 355–364. Springer, 2003.

[WD23] Simon Wietheger and Benjamin Doerr. A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5657–5665. ijcai.org, 2023.

[Weg01] Ingo Wegener. Theoretical aspects of evolutionary algorithms. In *Automata, Languages and Programming, ICALP 2001*, pages 64–78. Springer, 2001.

[XXM20] Qian Xu, Zhanqi Xu, and Tao Ma. A survey of multiobjective evolutionary algorithms based on decomposition: variants, challenges and future directions. *IEEE Access*, 8:41588–41614, 2020.

[ZD23] Weijie Zheng and Benjamin Doerr. Theoretical analyses of multiobjective evolutionary algorithms on multimodal objectives. *Evolutionary Computation*, 31:337–373, 2023.

[ZD24] Weijie Zheng and Benjamin Doerr. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*, pages 20874–20882. AAAI Press, 2024.

[ZL07] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11:712–731, 2007.

[ZLD22] Weijie Zheng, Yufei Liu, and Benjamin Doerr. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Conference on Artificial Intelligence, AAAI 2022*, pages 10408–10416. AAAI Press, 2022.

[ZLT01] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK report*, 103, 2001.

[ZQL+11] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1:32–49, 2011.

NeurIPS Paper Checklist

(i) **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The contribution in the introduction, which is clearly highlighted, provides explicit references to all the results in the main body of the paper. Moreover, the entire introduction relates our results with the state of the art.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

(ii) **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We discuss limitations in the introduction, at the end of the discussion of the sMOEA/D (Section 3.2), directly after stating our main result (Theorem 2), and in the conclusion.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

(iii) **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Each mathematical claim is rigorously stated, and its proof is given in the paper. We note that the proofs of Lemmas 5 and 6 are, due to space constraints, deferred to the appendix, but we still provide proof sketches in the main paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

(iv) **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [N/A]

Justification: This paper does not contain any experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

(v) **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [N/A]

Justification: This paper does not contain any experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

(vi) **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [N/A]

Justification: This paper does not contain any experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

(vii) **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [N/A]

Justification: This paper does not contain any experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

(viii) **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [N/A]

Justification: This paper does not contain any experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

(ix) **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We reviewed the NeurIPS Code of Ethics carefully and concluded that none of our results or assumptions violate the code.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

(x) **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: Our main contribution provides a robust alternative to existing general-purpose optimization heuristics. As such, the efficiency with which certain results can be obtained may change, but our contribution does not allow approaching problems that could not be approached before.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

(xi) **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper does not deal with data or models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

(xii) **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [N/A]

Justification: This paper does not use existing assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

(xiii) **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: We do not provide new assets, but we provide pseudocode for our newly introduced algorithm variant (Algorithm 2), and we discuss its design and motivation (Section 3.2).

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

(xiv) **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper neither involves crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

(xv) **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper neither involves crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

(xvi) **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The research conducted in this paper does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

A Appendix for "A Dynamic Decomposition Strategy for the MOEA/D With Proven Performance Guarantees"

This appendix contains the proofs of Lemmas 5 and 6. For the sake of convenience, we repeat the lemmas before we state their proof. Moreover, Section B details empirical results.

Lemma 5. *Consider the sMOEA/D optimizing* $\mathrm{LOTZ}_{u,v}$*. The optimal reference point is found in* $O(n^2)$ *expected iterations.*

Proof. First, by *optimum of a weight* w, we mean an individual x that minimizes $g_w(x, z^*)$.

In the first phase, the individuals corresponding to weights 0 and 1 optimize functions g_0 and g_1 until reaching their respective optimal individuals 1^n and 0^n, thus finding the reference point $z^* = (n, n)$.

Notice that weights 0 and 1 are in W at the beginning and are always in W throughout the execution of the algorithm. Moreover, x_0 optimizes $g_0 \colon (x, z^*) \mapsto z_2^* - v(T(x_0))$ while x_1 optimizes $g_1 \colon (x, z^*) \mapsto z_1^* - u(L(x_1))$.

Notice that all mutations that increase $T(x_0)$ (resp. $L(x_1)$) decrease the objective function of x_0 (resp. x_1). Hence, the resulting offspring replaces x_0 (resp. x_1). All mutations that decrease $T(x_0)$ (resp. $L(x_1)$) have a negative impact on their respective objective functions and, thus, do not result in the offspring replacing x_0 (resp. x_1). It follows that $T(x_0)$ and $L(x_1)$ can only increase. They increase with probability $\Omega(1/n)$ because the number of leading ones or number of leading zeros is increased by at least one by flipping the first zero or the last one respectively, happening with probability $\Omega(1/n)$ with standard bit mutation. It follows from the fitness level method that in $O(n^2)$ expected iterations of the algorithm, z^* is found. □

Lemma 6. *Consider the sMOEA/D optimizing* $\mathrm{LOTZ}_{u,v}$*, and consider an iteration in which the Pareto front is not covered. Then before the mutation step (line 8 in Algorithm 2), there exists an individual* x_w *that is not* g_w*-optimal.*

Proof. After the first phase, for all $w \in W$, the best-so-far individual x_w optimizes the function $g_w \colon x \mapsto \max\{w\big(u(n) - u(L(x))\big), (1-w)\big(v(n) - v(T(x))\big)\}$ because $z^* = (u(n), v(n))$. We omit z^* from the parameter of the function, as it is optimal and thus does not change.

First, suppose at the beginning of an iteration there is such a sub-optimal x_w. Then there is a weight w such that x_w is not g_w-optimal. Now suppose w is deleted (which occurs before the mutation), then there are two weights $w_1 < w < w_2$ with $f(x_{w_1}) = f(x_{w_2}) = f(x_w)$ that are not deleted this iteration. Suppose for the sake of contradiction that x_{w_1} and x_{w_2} are both optimal, then from the interval structure of the g-optima we showed in Lemma 3, it follows that x_w is also optimal for g_w, which is a contradiction. Thus at least one of x_{w_1} and x_{w_2} is not optimal, showing the claim.

Now suppose that for all $w \in W$, x_w is g_w-optimal. Because not the whole Pareto front is covered and the population consists of weights 0 and 1, there exist two consecutive weights in the population with a number of leading ones that differ by at least 2. Let w_1 and w_2 be those weights. These weights fall each into one of the intervals of the partition of Lemma 3. That is, using the notation of Lemma 3, there exist $a, b \leq n$ with $b \geq a + 2$ such that $w_1 \in (w_a, w_{a+1}]$ (or $w_1 = 0$) and $w_2 \in (w_b, w_{b+1}]$. Thus, $f(w_1) \neq f(w_2)$, which means that the sMOEA/D adds a weight w between w_1 and w_2. This implies that $w \notin (w_a, w_{a+1}]$ (and $w \neq 0$) as x_{w_1} has a strictly smaller g_w-value than x_{w_2} in this interval, due to x_{w_1} being the unique optimum in this interval, according to Lemma 3, and as the sMOEA/D chooses w such that the values of $g_w(x_{w_1})$ and $g_w(x_{w_2})$ agree. Likewise, we have $w \notin (w_b, w_{b+1}]$. Hence, $w \in (w_{a+1}, w_b]$ which means g_w-optimal individuals are not in the archive as elements of the Pareto front (which is the case of g_w-optimal individuals). Consequently, the individual corresponding to this new weight w is not g_w-optimal before the mutation phase. □

B Empirical Results

This appendix contains empirical results for the sMOEA/D that the reviewers asked for. We note that these results were not peer-reviewed. The code is available under `https://doi.org/10.5281/zenodo.23570076`.

B.1 Empirical Analysis on 0/1-KNAPSACK

We complement our theoretical results from Section 4 by experiments that analyze the sMOEA/D in aspects not present in the theoretical analysis. To this end, we follow the experimental setup of Zhang and Li [ZL07] and consider random instances of the NP-hard bi-objective 0/1-KNAPSACK problem. Zhang and Li [ZL07] used this setup when introducing the classic MOEA/D, and they showed that it empirically outperforms another common multi-objective search heuristic known as GSEMO.

In total, we compare the performance of the sMOEA/D to that of the classic MOEA/D, and we analyze the number of subproblems of the former during the run. Our results show that the archive of the sMOEA/D is superior to that of the classic MOEA/D. Moreover, the number of the subproblems of the sMOEA/D grows over time, suggesting that less time is spent on improving bad solutions, and more time is spent on exploring the Pareto front. We detail our experimental setup in Section B.1.1, and we discuss our results in Section B.1.2.

We also conducted experiments for the sMOEA/D and the classic MOEA/D on random $\text{LOTZ}_{u,v}$ instances, but we detail this analysis only in Section B.2. Our results support our claims from Section 4. That is, they suggest that the sMOEA/D optimizes $\text{LOTZ}_{u,v}$ in reasonable time by finding new elements of the Pareto front one by one. The classic MOEA/D gets to the Pareto front quickly but does not find all Pareto optima within reasonable time.

B.1.1 Setup

Objective function. We consider the bi-objective 0/1-KNAPSACK, which is an NP-hard combinatorial maximization problem. The objective is to select a certain number of at most $n \in \mathbb{N}_{\geq 1}$ items and to place them into two bags, where the profit per item may differ per bag. Moreover, this problem is subject to capacity constraints, as each item has a weight and each bag has a certain capacity. More formally, for all $i \in [n]$, let $x_i \in \{0,1\}$ denote whether item i is chosen for both bags (value 1) or for none (value 0). In addition, for all $j \in \{1,2\}$ let $p_i^{(j)} \in \mathbb{R}_{\geq 0}$ denote the profit of item i in bag j, and let $\omega_i^{(j)} \in \mathbb{R}_{\geq 0}$ denote the weight of item i in bag j. Last, let $c^{(j)} \in \mathbb{R}_{\geq 0}$ denote the maximum capacity of bag j. The objective is then to maximize the function $f\colon \{0,1\}^n \to \mathbb{R}^2_{\geq 0}$ given for all $j \in \{1,2\}$ by $f_j\colon x \mapsto \sum_{i\in[n]} p_i^{(j)} \cdot x_i$, subject to $\sum_{i\in[n]} \omega_i^{(j)} \cdot x_i \leq c^{(j)}$.

During our runs, we use f as objective function, and we only generate valid instances by applying a repair mechanism when creating new solutions, as detailed in the *Algorithm parameters* paragraph.

Instances. We consider 10 random, independent instances for each value of $n \in \{250, 500, 750\}$. For each instance, we generate the profits and weights independently by drawing a value from $[10..100]$ uniformly at random for each profit and for each weight. For each bag $j \in \{1,2\}$, we set the capacity to half the total sum of the (random) weights, that is, $c^{(j)} = \lfloor \frac{1}{2} \sum_{i\in[n]} p_i^{(j)} \rfloor$.

Algorithm parameters. For both the classic MOEA/D and the sMOEA/D, we use 1-point crossover, standard bit mutation, and a neighborhood size of $T = 10$. Each algorithm has for each problem instance a budget of 100 000 FEs, after which it is stopped. For the classic MOEA/D, we choose a uniform weight vector, that is, for all $i \in [N]$, we choose $w_i = \frac{i-1}{N-1}$, and, based on preliminary tests, $N = 200$. This is close to the parameters chosen by Zhang and Li [ZL07] for the classic MOEA/D.

After mutation, we apply a **repair mechanism**, guaranteeing that the offspring $y \in \{0,1\}^n$ satisfies the capacity constraints. Let $\widehat{w} \in [0,1]$ be the weight of the subproblem $g_{\widehat{w}}$ for which y was generated, and let $z^* \in \mathbb{R}^2$ be the reference point. In order to satisfy for all $j \in \{0,1\}$ the constraint $\sum_{i\in[n]} \omega_i^{(j)} \cdot y_i \leq c^{(j)}$, we iteratively remove the item from y that has the best ratio of fitness loss to weight, that is, an item $i' \in [n]$ with $y_i = 1$ that minimizes $\big(g_{\widehat{w}}(y^{(i'\to 0)}, z^*) - g_{\widehat{w}}(y, z^*)\big)/(\omega_{i'}^{(1)} + \omega_{i'}^{(2)})$, where $y^{(i'\to 0)}$ is identical to y except that $y_{i'}^{(i'\to 0)} = 0$. We note that the repair mechanism makes use of additional FEs when evaluating $g_{\widehat{w}}$, which are taken into account for the termination criterion.

Methodology. We run the classic MOEA/D and the sMOEA/D each 5 times independently per instance and per value of n. During our experiments, we log 1) the archive of each algorithm after being terminated for each run, and we log 2) the number of subproblems of the sMOEA/D during

Table 1: The set coverage score for comparing the archives of the classic MOEA/D (c) and of the sMOEA/D (s), as well as the archive size of either algorithm at the end of each run. We report the average numbers of all 5 runs for all 10 instances for all 3 values of n, rounding numbers. The last column is the average of the averages reported in the 10 instances to the left, and it is computed using the original numbers and rounding afterward. We see that s weakly dominates far more points of c than the other way around. For all 3 values of n, these results are statistically significant, according to a one-sided Mann–Whitney U test on the 10 averages per instance before rounding. The final archive sizes are roughly the same between both algorithms. Please also refer to Section B.1.2. We report on the standard deviation of these values in Table 2.

n	metric	instance										avg
250	$C_c(s)$	11.0	11.6	6.4	15.2	17.1	12.2	2.9	16.0	10.7	16.8	11.5
	$C_s(c)$	60.0	64.4	79.4	55.5	59.3	65.2	82.9	66.0	78.2	65.2	67.9
	$\lvert P_c\rvert$	27.8	29.2	30.0	27.4	34.2	33.2	32.0	30.2	23.6	30.4	29.8
	$\lvert P_s\rvert$	27.2	24.0	27.8	25.2	26.4	32.4	26.6	30.6	23.8	28.2	27.2
500	$C_c(s)$	6.4	6.1	26.0	0.0	5.1	6.4	18.3	0.0	16.7	6.0	9.0
	$C_s(c)$	75.9	68.5	61.2	90.3	86.7	73.7	68.1	80.7	69.7	86.5	76.0
	$\lvert P_c\rvert$	24.6	23.6	29.6	26.0	22.4	25.0	21.0	32.4	24.8	27.6	25.7
	$\lvert P_s\rvert$	21.0	23.4	21.6	22.0	22.6	18.8	19.6	20.0	19.6	21.4	21.0
750	$C_c(s)$	17.8	0.0	6.7	0.0	13.8	2.3	13.6	10.9	22.2	1.1	8.8
	$C_s(c)$	68.0	89.7	83.5	90.1	86.1	87.6	70.3	82.4	72.5	88.6	81.9
	$\lvert P_c\rvert$	20.4	20.0	20.0	20.0	14.0	15.0	22.4	16.2	20.0	18.8	18.7
	$\lvert P_s\rvert$	16.2	18.2	15.4	17.4	16.2	17.2	14.2	17.4	14.0	16.8	16.3

each iteration. Afterward, we compare the archives of the classic and the sMOEA/D on the same instances, using the following *set coverage* metric, defined as follows. Given two finite archives $A, B \subseteq \mathbb{R}^2$, the set coverage $C_A(B)$ of A and B is the percentage of solutions of B that are weakly dominated by A, that is, $C_A(B) = 100 \cdot |\{u \in B \mid \exists v \in A\colon y \succeq x\}|/|B|$.

Code. All of our experiments (excluding generating the data) runs in roughly 90 seconds on a computer with a 10-core CPU, using a RAM of about 200 MB. The data is generated in less than a second. We use deterministic seeds for generating random numbers. The code itself is available under `https://doi.org/10.5281/zenodo.23570076`.

B.1.2 Results

We report two results. First, we compare the archive of the sMOEA/D to that of the classic MOEA/D (Table 1). Afterward, we analyze the number of subproblems of the sMOEA/D over time (Figure 2).

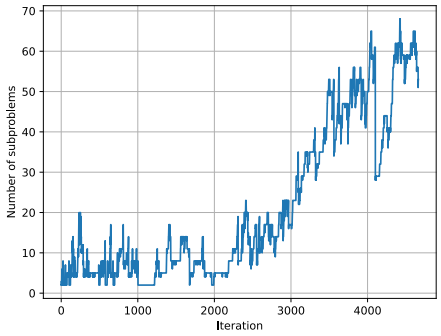
Comparing archives. Table 1 reports on the average of the 5 independent runs per instance and per value of n. In terms of the set coverage measure C, that is, the relative number of individuals in one archive weakly dominating that of the other, we see that the archive of the sMOEA/D always weakly dominates far more points of the archive of the classic MOEA/D than the other way around. This difference is statistically significant for a p-value of 0.01 for all three values of n. This suggests that the sMOEA/D generally finds better solutions in terms of weak domination.

Moreover, the table reports on the average size of the final archive per instance. Here, we see that the classic MOEA/D has a slightly larger archive on average than the sMOEA/D, but the sizes are often comparable between both algorithms. The difference is most likely a result of the random creation of the offspring, as the differences are within one standard deviation of the data (reported in Table 2). In combination with the set coverage above, these results suggest that the sMOEA/D outperforms the classic MOEA/D in parts of the search space that they both explore.

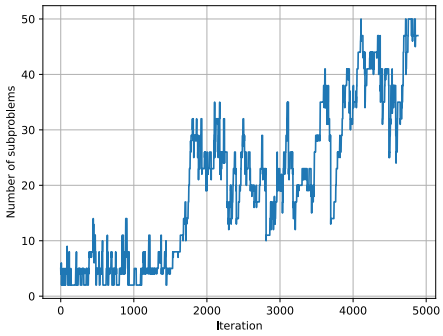
Number of subproblems. Figure 2 shows how the number of subproblems of the sMOEA/D evolves over time. Although the figure only shows two examples, we note that the figures for all instances follow the same qualitative trend. Initially, the sMOEA/D operates with not so many

Table 2: The standard deviation of the values from Table 1. Each value below is the average of the 5 standard deviations of the 5 runs per instance. We only report rounded values. We see that the average difference of the archive sizes seen in Table 1 is within one (average) standard deviation, which means that they are likely due to random variances.

n	metric	instance										avg
250	$C_c(s)$	13.6	19.3	9.4	13.6	19.9	13.2	2.9	17.6	11.6	10.8	13.2
	$C_s(c)$	19.2	23.7	17.3	17.6	21.5	17.7	10.9	28.6	16.8	20.6	19.4
	$\lvert P_c\rvert$	3.1	2.3	5.2	5.0	5.7	7.2	2.1	5.3	2.9	5.0	4.4
	$\lvert P_s\rvert$	6.9	3.0	2.9	7.4	5.5	2.1	4.5	3.0	3.8	7.2	4.6
500	$C_c(s)$	10.9	8.1	32.1	0.0	6.3	12.7	28.4	0.0	31.2	12.0	14.2
	$C_s(c)$	16.3	19.5	27.6	8.0	12.2	18.6	21.4	16.3	34.5	17.5	19.2
	$\lvert P_c\rvert$	2.7	6.9	8.8	5.1	3.9	2.5	5.1	2.1	4.2	4.6	4.6
	$\lvert P_s\rvert$	4.2	2.6	5.9	6.4	2.1	4.0	7.0	2.7	2.6	2.3	4.0
750	$C_c(s)$	28.0	0.0	13.3	0.0	21.8	4.6	19.5	15.6	27.8	2.1	13.3
	$C_s(c)$	32.5	9.8	30.7	12.2	17.5	10.2	31.4	15.9	27.3	10.1	19.8
	$\lvert P_c\rvert$	3.4	3.4	2.6	4.9	5.7	1.9	2.3	5.3	6.1	3.4	3.9
	$\lvert P_s\rvert$	1.9	3.5	3.1	3.4	2.6	5.2	5.7	4.6	3.6	3.8	3.7



(a) Instance 0 of size $n = 500$.



(b) Instance 0 of size $n = 750$.

Figure 2: The number of subproblems per iteration of the sMOEA/D on two random instances of the stated size. The number is initially small, resulting only in a few FEs, and it grows in later iterations, allowing the algorithm to explore the Pareto front more thoroughly. Please also refer to Section B.1.2.

subproblems. This means that many solutions weakly dominate each other, not requiring more subproblems. This strategy works well until it gets harder to find improving solutions, most likely because the Pareto fron (or local optima) are found. From this point on, the number of subproblems grows until progress is hard to make and the algorithm runs out of budget of FEs.

This behavior is good as it allows the algorithm to only use up few FEs early on and to use most of its budget once it gets closer to the Pareto front. This allows it to explore the Pareto front more thoroughly, which is potentially why we see in Table 1 that the archive of the sMOEA/D dominates that of the classic MOEA/D to a large degree. In contrast, the classic MOEA/D uses the same larger number of subproblems, namely $N = 200$, in each iteration, providing it with fewer opportunities to explore the Pareto front once it gets there.

Conclusion. The sMOEA/D competes with or even outperforms a good parametrization of the classic MOEA/D for a problem class on which the classic MOEA/D outperformed another common multi-objective search heuristic (the GSEMO) [ZL07]. The sMOEA/D does so while requiring almost no parameter choices of the user (just the neighborhood size T, coupled to the choice of the crossover operator). For parts of the Pareto front that both algorithms explore, the sMOEA/D outperforms the classic MOEA/D.

Table 2 reports the standard deviation of the values from Table 1. Below, we go into detail about the additional experiments we ran.

B.2 Performance on $\mathbf{LOTZ}_{u,v}$

We run the classic MOEA/D and the sMOEA/D in the same setting as mentioned in Section 3.2 on random $\mathrm{LOTZ}_{u,v}$ instances of size $n = 300$. For the classic MOEA/D, we choose $N = 4n$. We create both $u, v\colon [0..n] \to [0, 1]$ by choosing $u(0) = v(0) = 0$, $u(1) = v(1) = 1$, and $n - 1$ independent random numbers from $[0, 1]$ for either function and then sort them ascendingly. The resulting sorted ranges of length $n + 1$ represent the ranges of u and v, respectively.

Results. We run both algorithms on the same 5 random instances.

We observe that the sMOEA/D always covers the entire Pareto front within reasonable time, that is, within about 4 minutes (using the same computational resources as mentioned in the *Code* paragraph of Section B.1.1). In contrast, the classic MOEA/D does not cover the entire Pareto front even after 5 hours of execution.

For either algorithm, we log how quickly it finds solutions on the Pareto front. Both algorithms quickly find first solutions on the Pareto front and are then slowed down by finding the remaining solutions. The sMOEA/D finds all solutions eventually but does so one by one. Its number of subproblems grows steadily and proportionally to the size of the Pareto front covered. The classic MOEA/D finds the solutions to the subproblems quickly but does not manage to find all of the remaining optima within reasonable time.