Theoretical Analyses of Multiobjective Evolutionary Algorithms on Multimodal Objectives

Weijie Zheng, Benjamin Doerr

Theoretical Analyses of Multiobjective Evolutionary Algorithms on Multimodal Objectives*

Weijie Zheng
School of Computer Science and Technology
International Research Institute for Artificial Intelligence
Harbin Institute of Technology
Shenzhen, China
Southern University of Science and Technology
Shenzhen, China

Benjamin Doerr†
Laboratoire d'Informatique (LIX)
École Polytechnique, CNRS
Institut Polytechnique de Paris
Palaiseau, France

March 28, 2023

Abstract

Multiobjective evolutionary algorithms are successfully applied in many real-world multiobjective optimization problems. As for many other AI methods, the theoretical understanding of these algorithms is lagging far behind their success in practice. In particular, previous theory work considers mostly easy problems that are composed of unimodal objectives.

As a first step towards a deeper understanding of how evolutionary algorithms solve multimodal multiobjective problems, we propose the OneJumpZeroJump problem, a bi-objective problem composed of two objectives isomorphic to the classic jump function benchmark. We prove that

*A preliminary short version was published in the proceeding of AAAI 2021 [DZ21].
†Corresponding author.

the simple evolutionary multiobjective optimizer (SEMO) with probability one does not compute the full Pareto front, regardless of the runtime. In contrast, for all problem sizes n and all jump sizes $k \in [4..\frac{n}{2}-1]$, the global SEMO (GSEMO) covers the Pareto front in an expected number of $\Theta((n-2k)n^k)$ iterations. For $k = o(n)$, we also show the tighter bound $\frac{3}{2}en^{k+1} \pm o(n^{k+1})$, which might be the first runtime bound for an MOEA that is tight apart from lower-order terms. We also combine the GSEMO with two approaches that showed advantages in single-objective multimodal problems. When using the GSEMO with a heavy-tailed mutation operator, the expected runtime improves by a factor of at least $k^{\Omega(k)}$. When adapting the recent stagnation-detection strategy of Rajabi and Witt [RW22] to the GSEMO, the expected runtime also improves by a factor of at least $k^{\Omega(k)}$ and surpasses the heavy-tailed GSEMO by a small polynomial factor in k. Via an experimental analysis, we show that these asymptotic differences are visible already for small problem sizes: A factor-5 speed-up from heavy-tailed mutation and a factor-10 speed-up from stagnation detection can be observed already for jump size 4 and problem sizes between 10 and 50. Overall, our results show that the ideas recently developed to aid single-objective evolutionary algorithms to cope with local optima can be effectively employed also in multiobjective optimization.

1 Introduction

Real-world problems often contain multiple conflicting objectives. For such problems, a single best solution cannot be determined without additional information. One solution concept for such problems is to compute a set of solutions each of which cannot be improved without worsening in at least one objective (Pareto optima) and then let a decision maker choose one of these solutions. With their population-based nature, multiobjective evolutionary algorithms (MOEAs) are a natural choice for this approach and have indeed been very successfully applied here [ZQL+11].

Similar to the situation for single-objective evolutionary algorithms, the rigorous theoretical understanding of MOEAs falls far behind the success of these algorithms in practical applications. The classic works in this area have defined multiobjective, especially bi-objective, counterparts of well-analyzed single-objective benchmark functions used in evolutionary computation theory and have analyzed the performance of mostly very simple MOEAs on these benchmarks.

For example, in the problems COCZ [LTZ+02] and OneMinMax [GL10], the two objectives are both (conflicting) variants of the classic OneMax benchmark. The classic benchmark LeadingOnes was used to construct the LOTZ [LTZ04b] and WLPTNO [QYZ13] problems. These multiobjective benchmark problems

are among the most intensively studied [Gie03, BFN08, DKV13, DGN16, BQT18, HZCH19, HZ20, COGNS20, ZLD22, BQ22, ZD22]. We note that these problems are all very easy. They are composed of unimodal objectives and they have the further property that from each set of solutions P a set P' witnessing the Pareto front can be computed by repeatedly selecting a solution from P, flipping a single bit in it, adding it to P, and removing dominated solutions from P. They are thus relatively easy to solve, as witnessed by typical runtimes such as $O(n^2 \log n)$ on the OneMax-type problems or $O(n^3)$ on the LeadingOnes-inspired benchmarks. These runtimes naturally are higher than for the single-objective counterparts due to the fact the Pareto front of the multiobjective versions has size $\Theta(n)$, hence $\Theta(n)$ times more solutions have to be computed compared to the single-objective setting.

We defer a detailed discussion of the state of the art of rigorous analyses of MOEAs to Section 3 and state here only that, to the best of our knowledge, there is not a single runtime analysis work discussing in detail how MOEAs cope with problems composed of multimodal objectives.

Our contributions. This paper aims at a deeper understanding of how MOEAs cope with multiobjective problems consisting of natural, well-analyzed, multimodal objectives. In the theory of single-objective evolutionary computation, the class of Jump functions proposed by Droste et al. [DJW02] is a natural and intensively used multimodal function class that has inspired many ground-breaking results. Hence, in this paper, we design a bi-objective counterpart of the Jump function with problem size n and jump size k, called $\text{OneJumpZeroJump}_{n,k}$. It consists of one Jump function w.r.t. the number of ones and one Jump function w.r.t. the number of zeros, hence both objectives are isomorphic to classic jump functions. We compute its Pareto front in Theorem 7 and, as intended, observe that different from the easy multimodal benchmarks the Pareto front is not connected, that is, one cannot transform any solution on the Pareto front into any other solution on the Pareto front via one-bit flips. From this observation, we easily show that for all $n \in \mathbb{N}$ and all $k \in [2..\frac{n}{2}]$, the simple evolutionary multiobjective optimizer (SEMO) cannot find the full Pareto front of the OneJumpZeroJump benchmark (Theorem 9).

The heart of this work are mathematical runtime analyses for the global SEMO (GSEMO). We first show that the classic version of this algorithm for all n and $k \in [2..\lfloor \frac{n}{2} \rfloor]$ finds the Pareto front of $\text{OneJumpZeroJump}_{n,k}$ in $O((n-2k+3)n^k)$ iterations (and fitness evaluations) in expectation (Theorem 12). We show a matching lower bound of $\Omega((n-2k)n^k)$ for $k \in [4..\frac{n}{2}-1]$ (Theorem 13). Here and in the remainder, the asymptotic notation only hides constants independent of n and k. We note that our actual bounds are more precise. In particular, for $4 \leq k = o(n)$, the expected runtime is $\frac{3}{2}en^{k+1}$ apart from lower-order terms. This

result might be the first runtime analysis of an MOEA that is tight apart from lower order terms.

We then try to reuse two ideas that led to performance gains for multimodal problems in single-objective optimization. Doerr et al. [DLMN17] proposed the *fast mutation* operator in which the number of flipped bits follows a power-law distribution with (negative) exponent $\beta > 1$. They showed that the $(1+1)$ EA with this operator optimizes jump functions with jump size k by a factor of $k^{\Omega(k)}$ faster than the standard $(1+1)$ EA. We show that the GSEMO with fast mutation also witnesses such an improvement over the standard GSEMO. More precisely, we prove an upper bound of $O((n-2k)(en)^k/k^{k+0.5-\beta})$ for this algorithm (Theorem 20). We also provide a result with explicit leading constant, for the first time for an algorithm using fast mutation. We are optimistic that our non-asymptotic estimates for the number of flipped bits by this operator will be useful also in other works.

The stagnation-detection strategy of Rajabi and Witt [RW22] is a second way to cope with multimodality (in single-objective optimization). So far only used in conjunction with the $(1^+,\lambda)$ EA (and there mostly with the $(1+1)$ EA), it consists of counting for how long no improvement has been found and using this information to set the mutation rate (higher mutation rates when no improvement was made for longer time). With suitable choices of the parameters of this strategy, the $(1+1)$ EA can optimize jump functions with jump size k faster than the standard $(1+1)$ EA by a factor of $\Omega(k^{\Omega(k)})$. This is the same rough estimate as for the $(1+1)$ EA with fast mutation. A more detailed calculation, however, shows that the stagnation-detection $(1+1)$ EA is faster than the fast $(1+1)$ EA by a factor of $\Theta(k^{\beta-0.5})$. We note that this difference usually is not extremely large since β is usually chosen small ($\beta = 1.5$ was recommended by [DLMN17]) and k has to be small to admit reasonable runtimes (recall that the runtime dependence on n is $\Omega(n^k)$). Nevertheless, in experiments conducted by [RW22] the stagnation-detection $(1+1)$ EA was faster than the fast $(1+1)$ EA by around a factor of three. Different from fast mutation, it is not immediately clear how to incorporate stagnation detection into the GSEMO. Being an algorithm with non-trivial parent population, one question is whether one should count unsuccessful iterations globally or individually for each parent individual. Also, clearly the parameters of the algorithm need to be adjusted to the longer waiting times for an improvement. We succeeded in defining a stagnation-detection version of the GSEMO that effectively solves the OneJumpZeroJump problem, more precisely, that computes the full Pareto front in an expected runtime of $O((n-2k)(en)^k/k^k)$, again a $k^{\Omega(k)}$ factor improvement over the classic GSEMO and reducing the runtime guarantee for the heavy-tailed GSEMO by a small factor of $\Omega(k^{\beta-0.5})$, see Theorem 25.

Via a small set of experiments, we demonstrate that these are not only asymptotic differences. Compared to the standard GSEMO, we observe roughly a factor-5 speed-up with heavy-tailed mutation and a factor-10 speed-up with our stagnation-detection GSEMO, and this already for jump size $k = 4$ and problem sizes n between 10 and 50.

We note that this work is an extended version of the 7-page (excluding acknowledgments and references) preliminary paper [DZ21] and differs in the following ways. We estimate the leading constant of the asymptotic upper bound of the runtime of the GSEMO with heavy-tailed mutation for not too large jump size, which has not been shown in our preliminary version, and also not been shown before for the theoretical works with the heavy-tailed mutation. Besides, this version adds a formal discussion on the multimodality in multiobjective problems in Section 3, and contains all mathematical proofs that had to be omitted in [DZ21] for reasons of space.

The remainder of this paper is organized as follows. Section 2 introduces the basic definitions that will be used throughout this work. In Section 3, we discuss the relevant previous works. The $\textsc{OneJumpZeroJump}_{n,k}$ function class is defined in Section 4. Our mathematical runtime analyses are presented in Sections 5 to 8. Section 9 shows our experimental results. A conclusion is given in Section 10.

2 Basic Definitions

A multiobjective optimization problem consists of optimizing multiple objectives simultaneously. In this paper, we restrict ourselves to the maximization of bi-objective pseudo-Boolean problems $f = (f_1, f_2) : \{0,1\}^n \to \mathbb{R}^2$. We note that only sporadic theoretical works exist that discuss the optimization of more than two objectives [LTZ04b, BQT18, HZLL21].

We briefly review the standard notation in multiobjective optimization. For any two search points $x, y \in \{0,1\}^n$, we say that

- x *weakly dominates* y, denoted by $x \succeq y$, if and only if $f_1(x) \geq f_1(y)$ and $f_2(x) \geq f_2(y)$;
- x *dominates* y, denoted by $x \succ y$, if and only if $f_1(x) \geq f_1(y)$ and $f_2(x) \geq f_2(y)$ and at least one of the inequalities is strict.

We call $x \in \{0,1\}^n$ *Pareto optimal* if and only if there is no $y \in \{0,1\}^n$ such that $y \succ x$. All Pareto optimal solutions form the *Pareto set* S^* of the problem. The set $F^* = f(S^*) = \{(f_1(x), f_2(x)) \mid x \in S^*\}$ of the function values of the Pareto set is called the *Pareto front*.

For most multiobjective problems, the objectives are at least partially conflicting and thus there is usually not a single Pareto optimum. Since a priori it is not

clear which of several incomparable Pareto optima to prefer, the most common target is to compute the Pareto front, that is, compute a set P of solutions such that $f(P) := \{(f_1(x), f_2(x)) \mid x \in P\}$ equals the Pareto front F^*. This is our objective in this work as well. We note that if the Pareto front is excessively large, then one has to resort to approximating it in a suitable manner, but this will not be our problem here.

We will use $|x|_1$ and $|x|_0$ to denote the number of ones and the number of zeros of the search point $x \in \{0,1\}^n$. We use $[a..b]$ to denote the set $\{a, a+1, \dots, b\}$ for $a, b \in \mathbb{Z}$ and $a \leq b$.

3 Previous Works and A Discussion of Multimodality in Multiobjective Problems

Since this work conducts a mathematical runtime analysis of an evolutionary algorithm for discrete search spaces, we primarily discuss the state of the art inside this subarea of evolutionary computation. The mathematical runtime analysis as a rigorous means to understand evolutionary algorithms was started in the 1990s with works like [Müh92, Bäc93, Rud97]. The first analysis of MOEAs were conducted by Laumanns et al. [LTZ+02, LTZ04b], Giel [Gie03], and Thierens [Thi03]. The theory of MOEAs here often followed the successful examples given by single-objective works. For example, the multiobjective benchmarks COCZ [LTZ04b], and OneMinMax [GL10] are composed of two contradicting copies of the classic OneMax problem, the problems LOTZ [LTZ04b] and WLPTNO [QYZ13] follow the main idea of the LeadingOnes problem [Rud97].

Due to the higher complexity of analyzing MOEAs, many topics with interesting results on single-objective EAs received almost no attention for MOEAs. For example, the first runtime analysis of a single-objective algorithm using crossover was conducted by Jansen and Wegener [JW01] and was followed up by a long sequence of important works. In contrast, only sporadic works discuss crossover in MOEAs [NT10, QYZ13, HZCH19, BQ22]. Dynamic parameter settings for single-objective EAs were first discussed by Jansen and Wegener [JW05] and then received a huge amount of attention (see, e.g., the bookchapter [DD20]), whereas the only such result for MOEAs appears to be one result by Doerr et al. [DHP22]. How single-objective evolutionary algorithms cope with noise was discussed already by Droste [Dro04], but despite numerous follow-up works in single-objective optimization, there is only a single mathematical runtime analysis of an MOEA in the presence of noise [Gut12].

In this work, we regard another topic that is fairly well-understood in the evolutionary theory community on the single-objective side, but where essentially

nothing is known on the multiobjective side. It is well-known in general that local optima are challenging for evolutionary algorithms. Already one of the first runtime analysis works, the ground-breaking paper by Droste et al. [DJW02], proposes a multimodal (that is, having local optima different from the global optimum) benchmark of scalable difficulty. The jump function with difficulty parameter k, later called *gap size*, resembles the easy OneMax function except that it has a valley of very low fitness around the optimum. This valley consists of the $k-1$ first Hamming levels around the optimum, hence the search points in distance k form a local optimum that is easy to reach, but hard to leave. Mutation-based algorithms need to flip the right k bits to go from any point on the local optimum to a search point of better fitness (which here is the global optimum). While other multimodal benchmarks have been proposed and have received some attention, most notably trap functions [JS07] and the DeceptiveLeadingBlocks problem [LN19], it is safe to say that the Jump benchmark is the by far most studied one and that these studies have led to fundamental insights on how evolutionary algorithms cope with local optima.

For reasons of space, we only discuss some of the most significant works on jump functions. The first work [DJW02] determined the runtime of the $(1+1)$ EA on Jump functions with gap size $k \geq 2$ to be $\Theta(n^k)$, showing that local optima can lead to substantial performance losses. This bound was tightened and extended to arbitrary mutation rates by Doerr et al. [DLMN17]. This analysis led to the discovery of a heavy-tailed mutation operator and more generally, established the use of power-law distributed parameters, an idea that led to many interesting results subsequently [**?**, e.g.,]]AntipovBD21gecco,CorusOY21foga,AntipovBD22,DangELQ22. Again an analysis of how single-trajectory heuristics optimize jump functions led to the definition of a stagnation-detection mechanism, which in a very natural manner leads to a reduction of the time an EA is stuck on a local optimum [RW22]. Jump functions are among the first and most convincing examples that show that crossover can lead to significant speed-ups over only using mutation as variation operator [JW02, DFK+18, ADK22]. That estimation-of-distribution algorithms and ant-colony optimizers may find it easier to cope with local optima was also shown via jump functions [HS18, Doe21, BBD21, Wit23].

In contrast to this intensive discussion of multimodal functions in single-objective evolutionary optimization, almost no such results exist in multiobjective optimization. Before discussing this aspect in detail, let us agree (for this work) on the following terminology. We say that a function $f : \{0,1\}^n \to \mathbb{R}$ is *unimodal* if it has a unique maximum and if all other search points have a Hamming neighbor with strictly larger fitness. We say that f is *multimodal* if it has local optima different from the global optimum. Here we call *local optimum* a set S of search points

such that $f(x) = f(y)$ for all $x, y \in S$ and such that all neighbors of S, that is, all $z \notin S$ having a Hamming neighbor in S, have a smaller f-value. Note that a global optimum is also a local optimum according to this definition. With these definitions, unimodal functions are those where a hillclimber can reach the optimum via one-bit flips regardless of the initial solution. In contrast, multimodal functions definitely need a single-trajectory heuristic to flip more than one bit (when started in a suitable solution) or need the acceptance of truly inferior solutions. With our definition, plateaus of constant fitness render a function not unimodal, but they do not imply multimodality. This is convenient for our purposes as we shall not deal with such plateaus – for the simple reason that they behave again differently from unimodal functions and true local optima. We note that the role of plateaus in multiobjective evolutionary optimization has been extensively studied, among others, in [BFH+07, FHN10, FHN11, QTZ16, LZZZ16].

We extend the definitions of unimodality and multimodality to multiobjective problems in the natural way: A multiobjective problem is *unimodal* if all its objectives are unimodal functions. Many such problems are easy and even simple MOEAs employing one-bit flips as only variation operator, e.g., the SEMO, can find the full Pareto front. Examples of such problems include the well-studied benchmarks inspired by the unimodal single-objective benchmarks OneMax and LeadingOnes. However, somewhat surprisingly, it is not true that the SEMO can cover the full Pareto front of all unimodal multiobjective problems as we now show.

Lemma 1. *There exists a unimodal multiobjective problem whose full Pareto front cannot be covered by the SEMO in an arbitrarily long time with a positive probability.*

Proof. We consider the following bi-objective problem defined on $\{0,1\}^3$.

$$f(000) = (0,1), f(001) = (2,0), f(010) = (0,2), f(011) = (1,5)$$
$$f(100) = (4,0), f(101) = (5,0), f(110) = (3,3), f(111) = (0,4).$$

Then its Pareto front is $\{(1,5),(5,0),(3,3)\}$. We know that f_1 is unimodal as

$$f_1(000) < f_1(001), f_1(001) < f_1(101), f_1(010) < f_1(011), f_1(011) < f_1(001),$$
$$f_1(100) < f_1(101), f_1(110) < f_1(100), f_1(111) < f_1(110),$$

and $f_1(101)$ is the unique maximal value. We also know that f_2 is unimodal as

$$f_2(000) < f_2(010), f_2(001) < f_2(000), f_2(010) < f_2(011), f_2(100) < f_2(000),$$
$$f_2(101) < f_2(111), f_2(110) < f_3(111), f_2(111) < f_2(011),$$

and $f_2(011)$ is the unique maximal value.

Consider the run of the SEMO described in Table 1. It is easy to see that this process happens with probability of

$$\frac{1}{8} \cdot \frac{1}{3} \cdot \left(\frac{1}{2}\frac{1}{3}\right) \cdot \left(\frac{1}{2}\frac{1}{3}\right) \cdot \left(\frac{1}{2}\frac{1}{3}\right) = \frac{1}{5184} > 0.$$

Generation	Population
0	(100, (4,0))
1	(100, (4,0)), (000, (0,1))
2	(100, (4,0)), (010, (0,2))
3	(101, (5,0)), (010, (0,2))
4	(101, (5,0)), (011, (1,5))

Table 1: An example process of the SEMO. Given is the population at the end of each iteration (which for iteration 0 is the initial population). To ease understanding the process, with each individual we also state its objective value.

We now show that once the optimization process has reached the state described in the last line of Table 1, the Pareto front point $(3, 3)$ cannot be reached anymore. Note that 110 is the only Pareto optimum for $(3, 3)$, and its Hamming neighbors are $010, 100$, and 111. Since $f(010) = (0, 2), f(100) = (4, 0)$, and $f(111) = (0, 4)$, we know that 010 and 111 are strictly dominated by 011, and that 100 is strictly dominated by 101. Hence, even if any of them is generated, it cannot survive to the next generation when the current population is $\{101, 011\}$. Therefore, 110 cannot be generated (and $(3, 3)$ cannot be reached) via the SEMO with the one-bit mutation. □

We say that a multiobjective problem is *multimodal* if at least one objective is multimodal. We note that this does not automatically imply that the SEMO cannot find the full Pareto front, in fact, as the following proof of Lemma 2 shows, there are multi-objective problems consisting only of multimodal objectives such that the SEMO regardless of the initial solution computes the full Pareto front very efficiently. Clearly, our interest in this work are multimodal multiobjective problems that are harder to optimize. To avoid misunderstandings, we note that a second notion of multimodality exists. In *multimodal optimization*, the target is to find all or a diverse set of solutions having some optimality property [Pre15, LYQ16, TI20]. This notion is unrelated to our notion of multimodality.

Lemma 2. *There is a bi-objective problem $f = (f_1, f_2) : \{0, 1\}^n \to \mathbb{R}^2$ such that both objectives f_1 and f_2 have several local optima, but the SEMO algorithm, regardless of its initial solution, finds the full Pareto front in time $O(n^2 \log n)$.*

Proof. Let $k, \ell \in \mathbb{N}_{\geq 2}$ and $n = k\ell$. Define f_1 via

$$f_1(x) = \begin{cases} |x|_1, & \text{if } |x|_1 \equiv 0 \pmod{\ell}, \\ \lfloor |x|_1/\ell \rfloor \ell + (\ell - i), & \text{if } |x|_1 \equiv i \pmod{\ell}, \end{cases}$$

for all $x \in \{0,1\}^n$. Hence f_1 agrees with the classic OneMax benchmark when $|x|_1$ is a multiple of ℓ. Solutions x with $|x|_1 \equiv 1 \pmod{\ell}$ are local optima different from the global optimum, which is $x^* = 1^n$. The function f_1 is deceptive in each interval $[j\ell + 1..j\ell + \ell - 1]$, $j = 0, \ldots, k-1$.

We define f_2 analogously, but with the roles of zeroes and ones interchanged, that is,

$$f_2(x) = \begin{cases} |x|_0, & \text{if } |x|_0 \equiv 0 \pmod{\ell}, \\ \lfloor |x|_0/\ell \rfloor \ell + (\ell - i), & \text{if } |x|_0 \equiv i \pmod{\ell}, \end{cases}$$

for all $x \in \{0,1\}^n$. Clearly, f_2 has the same characteristics as f_1, in particular, it is highly multimodal with its k local optima.

Figure 1 plots these two functions with $(n, \ell) = (50, 10)$. We clearly to see the $n/\ell + 1 = 6$ local optima for both objectives, namely the points with $|x|_1 \equiv 1 \pmod{\ell}$ and $x = 1^n$ for f_1 and the points with $|x|_0 \equiv 1 \pmod{\ell}$ and $x = 0^n$ for f_2.

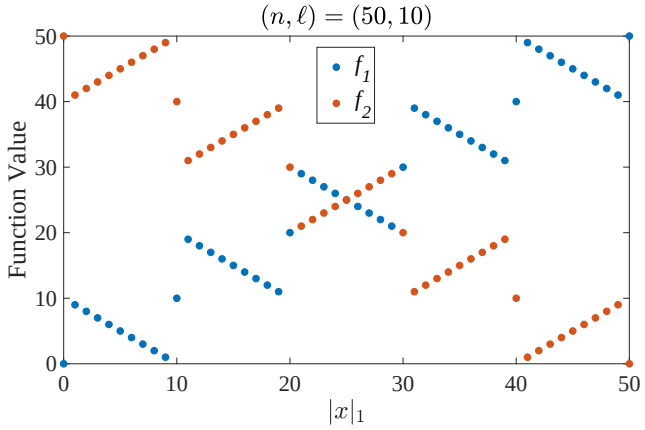

Figure 1: The values of f_1 and f_2 with respect to $|x|_1$, the number of ones in the search point x.

We note that any solution is a Pareto optimum of $f = (f_1, f_2)$. Indeed, we have $f_1(x) + f_2(x) = n$ for all $x \in \{0,1\}^n$. Consequently, if $f_1(x) > f_1(y)$ for some $x, y \in \{0,1\}^n$, then necessarily $f_1(x) < f_2(y)$ and thus x does not dominate y.

Since all solutions are Pareto optima and since f has the same classes of search points of equal fitness, the optimization process of the SEMO on f is identical

to the one on OneMax. With [GL10, Theorem 3] and its proof, our claim is proven. □

Having made precise what we understand as multimodal multiobjective problem, we review the known results on such problems.

To the best of our knowledge, only very few runtime analysis works on multimodal multiobjective problems exist and they all are not focussed on the possible difficulties arising from the multimodality. Roughly speaking, these can be divided into works that construct artificial problems to demonstrate, often in an extreme way, a particular strength or weakness of an algorithm, and works on classic combinatorial optimization problems that happen to be multimodal. The first set of works gave rise to the multimodal problems ZPLG, SPG, and Dec-obj-MOP. The first two of these were proposed by Qian et al. [QTZ16] and are defined as follows.

Definition 3 (ZPLG [QTZ16])**.** *The function* $ZPLG : \{0,1\}^n \to \mathbb{R} \times \{0,1,2\}$ *is defined by*

$$ZPLG(x) = \begin{cases} (n+1, 1), & \text{if } x = 1^i 0^{n-i}, i \in [1..\frac{3}{4}n - 1]; \\ (n+2+i, 0), & \text{if } x = 1^{\frac{3}{4}n+2i} 0^{\frac{1}{4}n-2i}, i \in [0..\frac{1}{8}n]; \\ (|x|_0, 2), & \text{else.} \end{cases}$$

Definition 4 (SPG [QTZ16])**.** *The function* $SPG : \{0,1\}^n \to \mathbb{R} \times \{0,1\}$ *is defined by*

$$SPG(x) = \begin{cases} (-1, 0), & \text{if } x = 1^i 0^{n-i}, i \bmod 3 = 1, i \in [1..n]; \\ (in, 0), & \text{if } x = 1^i 0^{n-i}, i \bmod 3 \in \{0,2\}, i \in [1..n]; \\ (|x|_0, 1), & \text{else.} \end{cases}$$

In order to investigate the effect of mixing low-level heuristics, Qian et al. [QTZ16] showed that mixing fair selection w.r.t. the decision space and the objective space is beneficial for ZPLG, and that mixing 1-bit and 2-bit mutation is efficient for SPG.

In the first theoretical study of decomposition-based MOEAs, Li et al. [LZZZ16] analyzed how the MOEA/D solves several multiobjective problems, among them the following multimodal one.

Definition 5 (Dec-obj-MOP [LZZZ16])**.** *The function* Dec-obj-MOP : $\{0,1\}^n \to \mathbb{R}^2$ *is defined by*

$$\textsc{Dec-obj-MOP}(x) = (n + 1 - |x|_0 \bmod n + 1, n + |x|_0 \bmod n + 1).$$

We note that the three problems ZPLG, SPG, and DEC-OBJ-MOP all appear slightly artificial. Also, they have only one multimodal objective. The first objective of ZPLG has local optima on $x = 1^{\frac{3}{4}n+2i}0^{\frac{1}{4}n-2i}, i \in [0..\frac{1}{8}n]$, the first objective of SPG has local optima on $x = 0^n$ and $x = 1^i0^{n-i}, i \bmod 3 = 0, i \in [1..n]$, and the second objective of DEC-OBJ-MOP has two local optima on $x \in \{0^n, 1^n\}$.

The second type of works dealing with multimodal multiobjective problems are those which regard combinatorial optimization problems, for example, [LTZ04a, KB06, Neu07, NR08, Gre09, Hor09, QYZ13, QYZ15, QZTY18, FQT19, QBF20, RBN20, RNNF22]. Combinatorial optimization problems almost always are multimodal, simply because already a simple cardinality constraint suffices to render a problem multimodal. We note for some such problems the multimodality is not very strong, e.g., for minimum spanning tree problems flipping two bits suffices to leave a local optimum. Overall, all these works are relatively problem-specific and we could not distill from these any general insights on how MOEAs cope with local optima.

4 The OneJumpZeroJump Problem

To study via mathematical means how MOEAs cope with multimodality, we now define and analyze a class of bi-objective functions of scalable difficulty. As mentioned above, this is strongly influenced by the single-objective JUMP function class proposed in [DJW02], which is intensively used in the theory of single-objective evolutionary computation and which gave rise to many interesting results including that larger mutation rates help in the optimization of multimodal functions, like [DLMN17], that crossover can help to cope with multimodality, like [JW02, DFK+18], and that estimation-of-distribution algorithms and the $(1 + (\lambda, \lambda))$ GA can significantly outperform classic evolutionary algorithms on multimodal problems, like [HS18, Doe21, AD20, ADK22].

We recall that for all $n \in \mathbb{N}$ and $k \in [1..n]$, the jump function $\text{JUMP}_{n,k} : \{0,1\}^n \to \mathbb{R}$ with problem size n and gap size k is defined by $\text{JUMP}_{n,k}(x) = k+|x|_1$, if $|x|_1 \in [0..n-k] \cup \{n\}$ and $\text{JUMP}_{n,k}(x) = n - |x|_1$ otherwise. Hence for $k \geq 2$, this function has a valley of low fitness around its optimum $x^* = 1^n$, which can be crossed only by flipping k specific bits (or accepting solutions with very low fitness). We define the $\text{ONEJUMPZEROJUMP}_{n,k}$ function as a bi-objective counterpart of the function $\text{JUMP}_{n,k}$.

Definition 6 (OneJumpZeroJump$_{n,k}$). *Let $n \in \mathbb{N}$ and $k = [1..n]$. The function OneJumpZeroJump$_{n,k} = (f_1, f_2) : \{0,1\}^n \to \mathbb{R}^2$ is defined by*

$$f_1(x) = \begin{cases} k + |x|_1, & \text{if } |x|_1 \leq n-k \text{ or } x = 1^n, \\ n - |x|_1, & \text{else;} \end{cases}$$

$$f_2(x) = \begin{cases} k + |x|_0, & \text{if } |x|_0 \leq n-k \text{ or } x = 0^n, \\ n - |x|_0, & \text{else.} \end{cases}$$

Hence the first objective of OneJumpZeroJump$_{n,k}$ is just the classic Jump$_{n,k}$ function. The second objective has a fitness landscape isomorphic to this function, but the roles of zeros and ones are interchanged, that it, $f_2(x) = \text{Jump}_{n,k}(\bar{x})$, where $\bar{x}$ is the bitwise complement of x. Figure 2 displays these two functions and in particular the two local optima on $|x|_1 = n-k$ and $x = 1^n$ for the first objective and two local optima on $|x|_1 = k$ and $x = 0^n$ for the second objective.

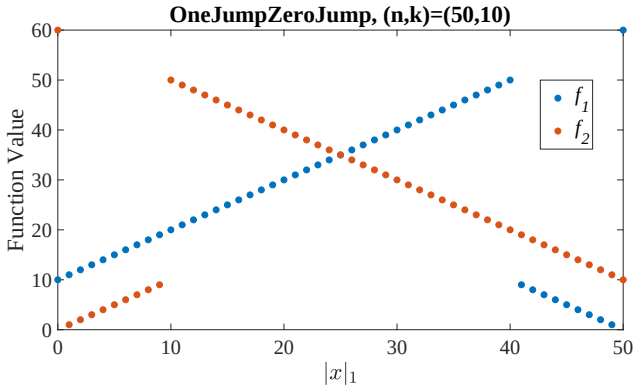

Figure 2: The values of two objectives in OneJumpZeroJump$_{n,k}$ with respect to $|x|_1$, the number of ones in the search point x.

In the following theorem we determine the Pareto set and front of the OneJumpZeroJump$_{n,k}$ function. As Figure 2 suggests, for $k \geq 2$ the Pareto set consists of an inner region of all search points x with $|x|_1 \in [k..n-k]$ and the two extremal points 1^n and 0^n, as visualized in Figure 3. We shall call the Pareto front values $\{(a, 2k+n-a) \mid a \in [2k..n]\}$ the *inner part/region* of the Pareto front in this paper.

Theorem 7. *The Pareto set of the OneJumpZeroJump$_{n,k}$ function is $S^* = \{x \mid |x|_1 \in [k..n-k] \cup \{0,n\}\}$, and the Pareto front is $F^* = \{(a, 2k+n-a) \mid a \in [2k..n] \cup \{k, n+k\}\}$.*

Proof. We firstly prove that for any given $x \in S^*$, $y \not\succ x$ holds for all $y \in \{0,1\}^n$. If $y \succ x$, then $f_1(y) \geq f_1(x)$ and $f_2(y) \geq f_2(x)$, and at least one of two inequalities

is strict. W.l.o.g., let $f_1(y) > f_1(x)$. From $x \in S^*$, we have $f_1(x) \geq k$, hence $y \neq 0^n$. Since $f_1(y) \leq n+k$, we know $x \neq 1^n$. Then we have $|x|_1 \in [k..n-k] \cup \{0\}$ since $x \in S^*$.

- If $|x|_1 \in [k..n-k]$, then $k \leq |x|_1 < |y|_1 \leq n-k$ or $|y|_1 = n$. For $k \leq |y|_1 \leq n-k$, we have $k \leq |y|_0 < |x|_0 \leq n-k$. Then $f_2(y) = k + |y|_0 < k + |x|_0 = f_2(x)$. For $|y|_1 = n$, we have $f_2(y) = k < k + k \leq f_2(x)$ since $|x|_0 \in [k..n-k]$ from $|x|_1 \in [k..n-k]$.
- If $|x|_1 = 0$, then $f_2(x) = k + n > f_2(y)$ since $y \neq 0^n$.

That is, we obtain $f_2(y) < f_2(x)$ which is contradictory to $y \succ x$. Hence, S^* is the Pareto set.

We secondly prove that for any given $y \in \{0,1\}^n \setminus S^*$, there is $x \in S^*$ such that $x \succ y$. Since $y \in \{0,1\}^n \setminus S^*$, we have $|y|_1 \in [1..k-1] \cup [n-k+1..n-1]$.

- If $|y|_1 \in [1..k-1]$, we have $f_1(y) = k + |y|_1$ and $f_2(y) = |y|_1$. Then we could take x with $|x|_1 = k$, and have $f_1(x) = k+k > f_1(y)$ and $f_2(x) = k+n-k > f_2(y)$. Then $x \succ y$.
- If $|y|_1 \in [n-k+1..n-1]$, we have $f_1(y) = n-|y|_1$ and $f_2(y) = k+|y|_0 \leq 2k-1$. Then we could take x with $|x|_1 = n-k$, and have $f_1(x) = n > f_1(y)$ and $f_2(x) = k+k > f_2(y)$. Then $x \succ y$.

Hence, F^* is the Pareto front and this theorem is proven. □

Theorem 7 implies that for $k > n/2$, the Pareto front consists only of the two extremal solutions 0^n and 1^n. This appears to be a relatively special situation that is not very representative for multimodal multiobjective problems. For this reason, in the remainder, we shall only regard the case that $k \leq n/2$. In this case, again from Theorem 7, we easily obtain in the following corollary a general upper bound on the size of any set of solutions without pair-wise weak domination, and thus also on the size of the population in the algorithms discussed in this work.

Corollary 8. *Let $k \leq n/2$. Consider any set of solutions P such that $x \not\preceq y$ w.r.t. $\textsc{OneJumpZeroJump}_{n,k}$ for all $x, y \in P$ with $x \neq y$. Then $|P| \leq n - 2k + 3$.*

Proof. We first note that any solution that lies on the Pareto front dominates any solution not on the Pareto front. Hence either P is a subset of the Pareto front or it contains no point on the Pareto front. In the first case, Theorem 7 immediately shows our claim. Hence assume that P contains no point of the Pareto front. Let $x, y \in P$ with $|x|_1, |y|_1 \in [1..k-1]$. Assume without loss of generality that $|x|_1 \leq |y|_1$. Then $x \preceq y$. Hence P contains at most one solution x with $|x|_1 \in [1..k-1]$. A symmetric argument shows that P contains at most one solution x with $|x|_1 \in [n-k+1..n-1]$. Hence $|P| \leq 2 \leq n - 2k + 3$. □

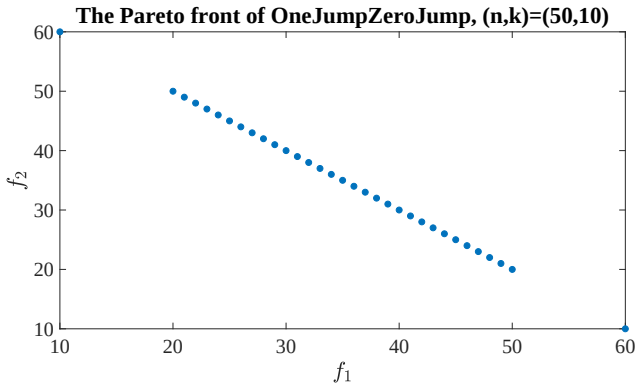

Figure 3: The Pareto front for the OneJumpZeroJump$_{n,k}$ function with $(n, k) = (50, 10)$.

5 SEMO Cannot Optimize OneJumpZeroJump Functions

The simple evolutionary multiobjective optimizer (SEMO), proposed by [LTZ+02], is a well-studied basic benchmark algorithm in multiobjective evolutionary theory [QYZ13, LZZZ16]. It is a multiobjective analogue of the randomized local search (RLS) algorithm, which starts with a random individual and tries to improve it by repeatedly flipping a single random bit and accepting the better of parent and this offspring. As a multiobjective algorithm trying to compute the full Pareto front, the SEMO naturally has to work with a non-trivial population. This is initialized with a single random individual. In each iteration, a random parent is chosen from the population. It generates an offspring by flipping a random bit. The offspring enters the population if it is not weakly dominated by some individual already in the population. In this case, any individual dominated by it is removed from the population. The details of SEMO are shown in Algorithm 1. We note that more recent works such as [FHN10] define the (G)SEMO minimally different, namely so that in case of equal objective values the offspring enters the population and replaces the existing individual with this objective value. Preferring offspring in case of equal objective values allows EAs to traverse plateaus of equal fitness and this is generally preferred over keeping the first solution with a certain objective value. For our purposes, this difference is not important since all points with equal fitness behave identically.

In the following Theorem 9, we show that the SEMO cannot cope with the multimodality of the OneJumpZeroJump$_{n,k}$ function. Even with infinite time, it does not find the full Pareto front of any OneJumpZeroJump function. We note that this result is not very surprising and we prove it rather for reasons of

Algorithm 1 SEMO

1: Generate $x \in \{0,1\}^n$ uniformly at random and $P \leftarrow \{x\}$
2: **loop**
3: Uniformly at random select one individual x from P
4: Generate x' via flipping one bit chosen uniformly at random
5: **if** there is no $y \in P$ such that $x' \preceq y$ **then**
6: $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
7: **end if**
8: **end loop**

completeness. It is well-known in single-objective optimization that the *randomized local search* heuristic with positive (and often very high) probability fails on multimodal problems (the only published reference we found is [DJK08, Theorem 9], but surely this was known before). It is for the same reason that the SEMO algorithm cannot optimize the OneJumpZeroJump function.

Theorem 9. *For all $n, k \in \mathbb{N}$ with $k \in [2..\lfloor \frac{n}{2} \rfloor]$, the SEMO cannot optimize the $\textsc{OneJumpZeroJump}_{n,k}$ function.*

Proof. Note that any individual with $m' \in [k..n-k]$ ones has the function value $(m'+k, n-m'+k) \geq (2k, 2k)$. Since a search point with $m \in [1..k-1]$ ones has the function value $(m+k, m) \leq (2k-1, k-1)$ and a search point with $m \in [1..k-1]$ zeros has the function value $(m, m+k) \leq (k-1, 2k-1)$, we know that any newly-generated search point with number of ones in $[k..n-k]$ will lead to the removal from the current population of all individuals with number of zeros or ones in $[1..k-1]$ and will also prevent any such individual from entering the population in the future.

W.l.o.g., suppose that the initial individual has at most $\frac{n}{2}$ ones. We show that 1^n can never enter the population. Assume first that never in this run of the SEMO a search point with $n-k$ ones is generated. Since each offspring has a Hamming distance of one from its parent, the statement "Never an individual with $n-k$ or more ones is generated" holds in each generation. In particular, the search point 1^n is never generated.

Hence assume now that at some time t for the first time an individual x with $n-k$ ones is generated. As in the previous paragraph, up to this point no search point with more than $n-k$ ones is generated. In particular, the point 1^n is not generated so far. Since x lies on the Pareto front, it enters the population. From this point on, by our initial considerations, no individual with $n-k+1$ to $n-1$ ones can enter the population. In particular, at no time a parent with $n-1$ ones can be selected, which is a necessary precondition for generating the point 1^n. Consequently, the point 1^n will never be generated. □

6 Runtime Analysis for the GSEMO

As we have shown in the previous section, to deal with the OneJumpZeroJump benchmark a mutation-based algorithm needs to be able to flip more than one bit. The global SEMO (GSEMO), proposed by [Gie03], is a well-analyzed MOEA that has this ability. Generalized from the $(1+1)$ EA algorithm, it uses the standard bit-wise mutation, that is, each bit is flipped independently with the same probability of, usually, $1/n$. The details are shown in Algorithm 2.

Algorithm 2 GSEMO

1: Generate $x \in \{0,1\}^n$ uniformly at random and $P \leftarrow \{x\}$
2: **loop**
3: Uniformly at random select one individual x from P
4: Generate x' via independently flipping each bit value of x with probability $1/n$
5: **if** there is no $y \in P$ such that $x' \preceq y$ **then**
6: $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
7: **end if**
8: **end loop**

Theorem 12 will show that GSEMO can find the Pareto front. To make the main proof clearer, some propositions are extracted as the following lemmas. Lemma 10 shows that once a solution in the inner part of the Pareto front is part of the population, it takes $O(n^2 \log n)$ iterations until the whole inner part of the Pareto front is covered.

Lemma 10. *Consider optimizing the function $f :=$ OneJumpZeroJump$_{n,k}$ via the GSEMO. Let $a_0 \in [2k..n]$. Assume that at some time the population contains an individual x with $f(x) = (a_0, 2k+n-a_0)$. Then the expected number of interations until all $(a, 2k+n-a)$, $a \in [2k..n]$, are covered by the population is at most $2en(n-2k+3)(\ln\lceil\frac{n}{2}\rceil + 1)$ iterations.*

Proof. Note that any solution corresponding to $(a, 2k+n-a)$ in the Pareto front contains exactly $a-k$ ones. If the population contains such an individual, then the probability to find $(a+1, 2k+n-a-1)$ in the Pareto front is at least

$$\frac{1}{n-2k+3}\frac{n-a+k}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{n-a+k}{en(n-2k+3)},$$

where we use that the population size is at most $n-2k+3$, see Corollary 8. Hence, by a simple Markov chain argument (similar to Wegener's fitness level method [Weg01]), the expected time to find all $(a, 2k+n-a)$, $a \in [a_0+1..n]$, is

at most

$$\sum_{a=a_0}^{n-1} \frac{en(n-2k+3)}{n-a+k}. \tag{1}$$

Similarly, the probability to find $(a-1, 2k+n-a+1)$ in the Pareto front once the population contains a solution with function value $(a, 2k+n-a)$ is at least

$$\frac{1}{n-2k+3}\frac{a-k}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{a-k}{en(n-2k+3)},$$

and the expected time to find all $(a, 2k+n-a)$, $a \in [2k..a_0-1]$, is at most

$$\sum_{a=a_0}^{2k+1} \frac{en(n-2k+3)}{a-k}. \tag{2}$$

By comparing the individual summands, the sum of (1) and (2) is at most

$$2\sum_{a=\lfloor n/2 \rfloor + k}^{n-1} \frac{en(n-2k+3)}{n-a+k} \leq 2en(n-2k+3)\left(\ln\left\lceil\frac{n}{2}\right\rceil + 1\right). \qquad \square$$

From a population covering the whole inner part of the Pareto front, the two missing extremal points of the front can be found in roughly $O(n^{k+1})$ iterations, as the following Lemma 11 shows.

Lemma 11. *Consider the GSEMO optimizing the* $\textsc{OneJumpZeroJump}_{n,k}$ *function. Assume that at some time, all* $(a, 2k+n-a)$, $a \in [2k..n]$, *are covered by the current population. Then the two missing points* $(a, 2k+n-a)$, $a \in \{k, n+k\}$, *of the Pareto front will be found in an expected number of at most* $\frac{3}{2}en^k(n-2k+3)$ *additional generations.*

Proof. We pessimistically calculate the time to generate two elements $(a, 2k+n-a)$, $a \in \{k, n+k\}$, in the Pareto front ignoring the fact that the current population may contain the corresponding solutions 0^n and 1^n already. Similar to the proof of Lemma 10, we note that the probability to find $(k, k+n)$ in the Pareto front from a solution with function value $(2k, n)$ is at least

$$\frac{1}{n-2k+3}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{n-k} \geq \frac{1}{en^k(n-2k+3)}.$$

The same estimate holds for the probability to find $(k+n, k)$ in the Pareto front from one solution with function value $(n, 2k)$. Hence, the expected time to find one of $(k, n+k)$ and $(k+n, k)$ is at most $\frac{e}{2}n^k(n-2k+3)$, and the expected time to find the remaining element of the Pareto front is at most $en^k(n-2k+3)$ additional iterations. $\square$

Now we establish our main result for the GSEMO.

Theorem 12. *Let $n \in \mathbb{N}_{\geq 2}$ and $k \in [1..\lfloor \frac{n}{2} \rfloor]$. The expected runtime of the GSEMO optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function is at most*

$$e(n-2k+3)(\tfrac{3}{2}n^k + 2n\ln\lceil \tfrac{n}{2} \rceil + 3).$$

For $2 \leq k = o(n)$, this bound is $\frac{3}{2}en^{k+1} \pm o(n^{k+1})$.

Proof. We first consider the time until the population P contains an x such that $(f_1(x), f_2(x)) = (a, 2k+n-a)$ for some $a \in [2k..n]$. If the initial search point has a number of ones in $[k..n-k]$, then such an x is already found. Otherwise, the initial search point has at most $k-1$ ones or at most $k-1$ zeros. W.l.o.g., we consider an initial point with $m \leq k-1$ zeros. If $m = 0$, that is, the initial point is 1^n, then its function value is $(n+k, k)$. Since any point $\tilde{x}$ with $\tilde{m} \in [1..k-1]$ zeros has function value $(\tilde{m}, k+\tilde{m})$, such a point is not dominated by 1^n, and thus will be kept in the population. After that, since $(\tilde{m}, k+\tilde{m})$ increases with respect to $\tilde{m}$, one individual with larger $\tilde{m} \in [1..k-1]$ will replace the one with smaller $\tilde{m} \in [1..k-1]$. Finally, any individual with $m' \in [k..n-k]$ zeros has function value $(n-m'+k, m'+k) > (k-1, 2k-1)$ and thus dominates all search points with $\tilde{m} \in [1..k-1]$ zeros. Hence, we pessimistically add up the expected waiting times for increasing the number of zeros one by one until one individual with k zeros is found, which need an expected number of iterations of at most

$$\begin{aligned} &\sum_{m=0}^{k-1} \left(\frac{1}{n-2k+3} \frac{n-m}{n} \left(1-\frac{1}{n}\right)^{n-1} \right)^{-1} \leq \sum_{m=0}^{k-1} \frac{en(n-2k+3)}{n-m} \\ &\leq en(n-2k+3)\frac{k}{n-k+1} \leq en(n-2k+3). \end{aligned} \tag{3}$$

Then from Lemma 10, we know that all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front will be found in an expected number of at most $2en(n-2k+3)(\ln\lceil \frac{n}{2} \rceil + 1)$ iterations. After that, from Lemma 11, the points $(k, n+k)$ and $(n+k, k)$ in the Pareto front will be found in an expected number of at most $\frac{3}{2}en^k(n-2k+3)$ additional iterations. Hence, the expected iterations to find the Pareto front is at most $en(n-2k+3) + 2en(n-2k+3)(\ln\lceil \frac{n}{2} \rceil + 1) + \frac{3}{2}en^k(n-2k+3) = e(n-2k+3)(\frac{3}{2}n^k + 2n\ln\lceil \frac{n}{2} \rceil + 3)$. □

We finally show that this bound is very tight. For convenience, we only consider the case $k \geq 4$. In this case, it is sufficiently hard to generate an extremal point of the Pareto front from the inner part that we do not need a careful analysis of the population dynamics in the short initial phase in which the inner part of the Pareto front is detected, which is not the main theme of this work. That said, we are very optimistic that it would pose no greater difficulties to show that with

high probability at the first time when a solution with less than $\frac{1}{4}n$ or more than $\frac{3}{4}n$ ones enters the population, the population already contains $\Theta(n)$ individuals. Hence from this point on, only every $\Theta(n)$ iterations an outer-most point will be chosen as parent, and this should suffice to prove an $\Omega(n^{k+1})$ lower bound also for $k = 2$ and $k = 3$.

Theorem 13. *Let $n \in \mathbb{N}_{\geq 8}$ and $k \in [4..\frac{n}{2} - 1]$. The expected runtime of the GSEMO optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function is at least*

$$\frac{3}{2}e(n-2k+1)n^k \frac{n^k}{(n-1)^k}\left(1 - \frac{1}{n^{n/4-2}} - \frac{2}{n^{n-2k}} - \frac{e+3}{n} - \frac{4e(\ln n + 1)}{n^{k-3}}\right).$$

If $k = o(n)$, this bound becomes $\frac{3}{2}en^{k+1} - o(n^{k+1})$ and matches the upper bound of Theorem 12 apart from lower-order terms.

Proof. We first prove that with high probability the inner region of the Pareto front will be fully covered before 0^n or 1^n is generated and then use this to bound the time until the two extremal points of the Pareto front are found.

For the initial individual x, we have $E[|x|_1] = \frac{1}{2}n$. With the additive Chernoff bound, see, e.g., [Doe20, Theorem 1.10.7], we know that with probability at least

$$1 - 2\exp\left(-\frac{2}{n}\left(\frac{n}{4}\right)^2\right) = 1 - 2\exp\left(-\frac{1}{8}n\right),$$

the initial individual has $\frac{1}{4}n < |x|_1 < \frac{3}{4}n$. Then $|x|_1 < k$ or $|x|_1 > n - k$ only if $k > \frac{1}{4}n$. If $k > \frac{1}{4}n$, w.l.o.g., let $|x|_1 < k$, then we will show that with probability at least $1 - n^{-n/4+2} - 2n^{-n+2k} - en^{-1}$, an individual x' with $k \leq |x'|_1 \leq n - k$ is generated earlier than 1^n or 0^n. Note that before such an x' appears, the population consists of one individual with $\frac{1}{4}n < |x|_1 < k$ except if 1^n, 0^n, or a y with $|y|_1 > n - k$ is generated. We note that the probability to generate one of the exceptions from one individual with $\frac{1}{4}n < |x|_1 < k$ is at most

$$\frac{1}{n^{n-|x|_1}} + \frac{1}{n^{|x|_1}} + \frac{1}{n^{n-k+1-|x|_1}} < \frac{1}{n^{n-k+1}} + \frac{1}{n^{\frac{1}{4}n}} + \frac{1}{n^{n-2k+2}} \leq \frac{1}{n^{\frac{1}{4}n}} + \frac{2}{n^{n-2k+2}},$$

where the last inequality uses $k \geq 1$. Then with probability at least

$$\left(1 - n^{-n/4} - 2n^{-(n-2k+2)}\right)^{n^2} \geq 1 - n^{-n/4+2} - 2n^{-n+2k},$$

the population consists of only one individual with $\frac{1}{4}n < |x|_1 < k$ in n^2 iterations. Then from the similar calculation in (3) except changing $n - 2k + 3$ to 1, we know that it takes an expected time of at most en iterations to generate such an x'. Via Markov's inequality, we know that with probability at least $1 - \frac{en}{n^2} = 1 - \frac{e}{n}$,

such an x' will be generated in n^2 iterations. Hence, with probability at least $1 - n^{-n/4+2} - 2n^{-n+2k} - en^{-1}$, an individual x' with $k \leq |x'|_1 \leq n-k$ is generated earlier than 1^n or 0^n.

From this point on, we will first show that with probability at least $1 - \frac{8e(\ln n+1)}{n^{k-3}}$, all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front are found before 0^n or 1^n is generated. It is not difficult to see that any search point y in either of the two gaps, that it, with $|y|_1 \in [1..k-1] \cup [n-k+1..n-1]$, has both objective values less than any search point z with $|z|_1 \in [k..n-k]$, and thus cannot enter into the population. Therefore, the probability to generate 0^n or 1^n from some parent x with $|x|_1 \in [k..n-k]$ is at most

$$\frac{1}{n^{|x|_1}}\left(1-\frac{1}{n}\right)^{n-|x|_1} + \frac{1}{n^{n-|x|_1}}\left(1-\frac{1}{n}\right)^{|x|_1} \leq \frac{2}{n^k}. \tag{4}$$

From Lemma 10, we know that all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front will be found in an expected number of at most $2en(n-2k+3)(\ln\lceil\frac{n}{2}\rceil+1)$ iterations. Via Markov's inequality, we know that with probability at least $1 - \frac{2en(n-2k+3)(\ln\lceil\frac{n}{2}\rceil+1)}{2en^3(\ln n+1)} \geq 1 - \frac{1}{n}$, all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front will be found in $2en^3(\ln n+1)$ iterations. Hence, with (4), we know that the event that during all $2en^3(\ln n+1)$ iterations, 0^n or 1^n is not generated, has probability at least

$$\left(1-\frac{2}{n^k}\right)^{2en^3(\ln n+1)} \geq 1 - \frac{4en^3(\ln n+1)}{n^k} = 1 - \frac{4e(\ln n+1)}{n^{k-3}}.$$

Now all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front are found and the current population has size $n-2k+1$. From this time on, we compute the probability to generate the remaining Pareto optimal solution 0^n and 1^n as

$$\begin{aligned}
&\sum_{i=k}^{n-k} \frac{1}{n-2k+1}\left(\frac{1}{n^i}\left(1-\frac{1}{n}\right)^{n-i} + \frac{1}{n^{n-i}}\left(1-\frac{1}{n}\right)^{i}\right)\\
&= \frac{2}{(n-2k+1)n^k}\left(1-\frac{1}{n}\right)^{n-k}\left(1+\sum_{j=1}^{n-2k}\frac{1}{n^j}\left(1-\frac{1}{n}\right)^{-j}\right)\\
&\leq \frac{2}{(n-2k+1)n^k}\left(1-\frac{1}{n}\right)^{n-k}\left(1+\frac{\frac{1}{n-1}}{1-\frac{1}{n-1}}\right)\\
&= \left(1+\frac{1}{n-2}\right)\frac{2}{(n-2k+1)n^k}\left(1-\frac{1}{n}\right)^{n-k} = \frac{n-1}{n-2}\frac{2}{(n-2k+1)n^k}\left(1-\frac{1}{n}\right)^{n-k}.
\end{aligned}$$

Then the expected time to find 0^n or 1^n is at least $\frac{1}{2}(n-2k+1)n^k\left(1-\frac{1}{n}\right)^{k-n}\frac{n-2}{n-1}$. W.l.o.g., 0^n is found. Now the current population size is $n-2k+2$, and the

probability to find the remaining 1^n is

$$\begin{aligned}
&\frac{1}{n-2k+2}\cdot\frac{1}{n^n}+\sum_{i=k}^{n-k}\frac{1}{n-2k+2}\cdot\frac{1}{n^i}\left(1-\frac{1}{n}\right)^{n-i}\\
&\qquad\le\frac{1}{(n-2k+2)n^k}\left(1-\frac{1}{n}\right)^{n-k}\left(\frac{1}{n^{n-k}}\left(1-\frac{1}{n}\right)^{-n+k}+1+\frac{1}{n-2}\right)\\
&\qquad\le\frac{1}{(n-2k+2)n^k}\left(1-\frac{1}{n}\right)^{n-k}\left(1+\frac{2}{n-2}\right)\\
&\qquad=\frac{1}{(n-2k+2)n^k}\left(1-\frac{1}{n}\right)^{n-k}\frac{n}{n-2}
\end{aligned}$$

Then we need at least $(n-2k+2)n^k\left(1-\frac{1}{n}\right)^{k-n}\frac{n-2}{n}$ additional iterations in expectation to cover the whole Pareto front.

In summary, together with the above discussed probability for the initial individual and the probability that all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front are found before 0^n or 1^n is generated, we obtain that the lower bound to cover the Pareto front is

$$\begin{aligned}
&\left(1-\frac{1}{n^{n/4-2}}-\frac{2}{n^{n-2k}}-\frac{e}{n}-\frac{1}{n}-\frac{4e(\ln n+1)}{n^{k-3}}\right)\cdot\frac{3}{2}(n-2k+1)n^k\left(1-\frac{1}{n}\right)^{k-n}\frac{n-2}{n}\\
&\ge\frac{3}{2}(n-2k+1)n^k\left(1-\frac{1}{n}\right)^{k-n}\left(1-\frac{1}{n^{n/4-2}}-\frac{2}{n^{n-2k}}-\frac{e+3}{n}-\frac{4e(\ln n+1)}{n^{k-3}}\right)\\
&\ge\frac{3}{2}e(n-2k+1)n^k\frac{n^k}{(n-1)^k}\left(1-\frac{1}{n^{n/4-2}}-\frac{2}{n^{n-2k}}-\frac{e+3}{n}-\frac{4e(\ln n+1)}{n^{k-3}}\right).
\end{aligned}$$

□

7 GSEMO with Heavy-Tailed Mutation

In the previous section, we have shown that the GSEMO can optimize our multimodal optimization problem, but similar to the single-objective world (say, the optimization of Jump functions via simple evolutionary algorithms [DJW02]), the runtime increases significantly with the distance k that a solution on the Pareto front can have from all other solutions on the front. As has been discussed in [DLMN17], increasing the mutation rate can improve the time it takes to jump over such gaps. However, this work also showed that a deviation from the optimal mutation rate can be costly: A small constant-factor deviation from the optimal rate k/n leads to a performance loss exponential in k. For this reason, a heavy-tailed mutation operator was proposed. Compared to using the optimal (usually unknown) rate, it only loses a small polynomial factor (in k) in performance.

We now equip the GSEMO with the heavy-tailed mutation operator from [DLMN17] and observe similar advantages. We start by introducing the

heavy-tailed mutation operator and giving a first non-asymptotic estimate for the probabilities to flip a certain number of bits.

7.1 Heavy-Tailed Mutation Operator

The following capped power-law distribution will be used in the definition of the heavy-tailed mutation operator.

Definition 14 (Power-law distribution $D^\beta_{n/2}$)**.** *Let $n \in \mathbb{N}_{\geq 2}$ and $\beta > 1$. Let $C^\beta_{n/2} := \sum_{i=1}^{n/2} i^{-\beta}$. We say a random variable ξ follows the power-low $D^\beta_{n/2}$, written as $\xi \sim D^\beta_{n/2}$, if for all $\alpha \in [1..n/2]$, we have*

$$\Pr[\xi = \alpha] = \left(C^\beta_{n/2}\right)^{-1} \alpha^{-\beta}.$$

The heavy-tailed mutation operator proposed by [DLMN17], in the remainder denoted by $\textsc{mut}^\beta(\cdot)$, in each application independently samples a number α from the power-law distribution $D^\beta_{n/2}$ and then uses standard bit-wise mutation with mutation rate α/n, that is, flips each bit independently with probability α/n. [DLMN17] shows the two following properties of $\textsc{mut}^\beta$.

Lemma 15 ([DLMN17])**.** *Let $x \in \{0,1\}^n$ and $y \sim \textsc{mut}^\beta(x)$. Let $H(x,y)$ denote the Hamming distance between x and y. Then we have*

$$P^\beta_i := \Pr[H(x,y) = i] = \begin{cases} \left(C^\beta_{n/2}\right)^{-1} \Theta(1), & \text{for } i = 1; \\ \left(C^\beta_{n/2}\right)^{-1} \Omega(i^{-\beta}), & \text{for } i = [2..n/2]. \end{cases}$$

In order to obtain an understanding of the leading constants when generating a solution with a not too large Hamming distance, we prove the following estimations, which have not been shown before, and can also be applied to other previous works about the theoretical analysis of the heavy-tailed mutation. We suspect tighter estimates of the leading constants exist but leave it as interesting future work.

Lemma 16. *Let $x \in \{0,1\}^n$ and $y \sim \textsc{mut}^\beta(x)$. Let $H(x,y)$ denote the Hamming distance between x and y. Then we have*

$$P^\beta_i := \Pr[H(x,y) = i] \geq \begin{cases} \frac{\beta-1}{e\beta}, & \text{for } i = 1; \\ \frac{\beta-1}{2\sqrt{2\pi}e^{8\sqrt{2}+13}\beta} i^{-\beta}, & \text{for } i = [2..\lfloor\sqrt{n}\rfloor]. \end{cases}$$

The proof of Lemma 16 will use an elementary estimate for $\binom{n}{k}$ when $k \leq \sqrt{n}$. Since we have not found a good reference for it and since we believe that it might be useful for other runtime analyses, we formulate it as a separate lemma and give a formal proof.

Lemma 17. *Let $n \in \mathbb{N}$ and $i = [0..\lfloor\sqrt{n}\rfloor]$. Then $\binom{n}{i} \geq \frac{n^i}{2i!}$.*

Proof. Our claim holds trivially for $i = 0$ and 1. Thus, for any $n \in [1..3]$, this lemma holds. We now discuss $n \geq 4$ and $i \in [2..\lfloor\sqrt{n}\rfloor]$. By Weierstrass product inequality [**?**, see, e.g.,]Lemma 1.4.8(b)]Doerr20bookchapter, we know

$$\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\dots\left(1-\frac{i-1}{n}\right) \geq 1-\sum_{j=1}^{i-1}\frac{j}{n} = 1-\frac{\frac{(i-1)(i-1+1)}{2}}{n} \geq 1-\frac{i^2}{2n} \geq \frac{1}{2}.$$

Then we have

$$\begin{aligned}\binom{n}{i} &= \frac{n!}{(n-i)!i!} = \frac{n(n-1)\dots(n-i+1)}{i!} = \frac{n^i}{i!}\frac{n(n-1)\dots(n-i+1)}{n^i} \\ &= \frac{n^i}{i!}\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\dots\left(1-\frac{i-1}{n}\right) \geq \frac{n^i}{2i!}.\end{aligned}$$

□

Now we prove Lemma 16.

Proof of Lemma 16. We calculate

$$P_1^\beta = \Pr[H(x,y)=1] \geq \left(C_{n/2}^\beta\right)^{-1}\frac{1}{1^\beta}\binom{n}{1}\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{1}{eC_{n/2}^\beta}.$$

Since $\beta > 1$, we know

$$C_{n/2}^\beta = \sum_{i=1}^{n/2} i^{-\beta} \leq 1+\int_1^{n/2} x^{-\beta}\,dx = 1+\frac{(n/2)^{-\beta+1}-1}{-\beta+1} \leq 1+\frac{1}{\beta-1} = \frac{\beta}{\beta-1}. \tag{5}$$

Hence, $P_1^\beta \geq \frac{\beta-1}{e\beta}$.

For $i \in [2..\lfloor\sqrt{n}\rfloor]$, we have

$$\binom{n}{i} \geq \frac{n^i}{2i!} \geq \frac{n^i}{2\sqrt{2\pi i}(i/e)^i e^{1/(12i)}} = \frac{1}{2\sqrt{2\pi i}}\left(\frac{n}{i}\right)^i e^{i-1/(12i)},$$

where the second inequality uses Stirling's approximation. Using that $j \to j^i(1-\frac{j}{n})^{n-i}$ is unimodal and has its maximum at $j=i$, we compute

$$\begin{aligned}
P_i^\beta &= \left(C_{n/2}^\beta\right)^{-1} \sum_{j=1}^{n/2} \frac{1}{j^\beta} \binom{n}{i} \left(\frac{j}{n}\right)^i \left(1-\frac{j}{n}\right)^{n-i} \\
&\geq \frac{e^{i-1/(12i)}}{2\sqrt{2\pi i} C_{n/2}^\beta} \sum_{j=1}^{n/2} \frac{1}{j^\beta} \left(\frac{j}{i}\right)^i \left(1-\frac{j}{n}\right)^{n-i} \\
&\geq \frac{e^{i-1/(12i)}}{2\sqrt{2\pi i} C_{n/2}^\beta} \sum_{j=i-\lfloor\sqrt{i}\rfloor}^{i} \frac{1}{j^\beta} \left(\frac{j}{i}\right)^i \left(1-\frac{j}{n}\right)^{n-i} \\
&\geq \frac{e^{i-1/(12i)}}{2\sqrt{2\pi i} C_{n/2}^\beta \cdot i^\beta} \sum_{j=i-\lfloor\sqrt{i}\rfloor}^{i} \left(\frac{j}{i}\right)^i \left(1-\frac{j}{n}\right)^{n-i} \\
&\geq \frac{e^{i-1/(12i)}(\lfloor\sqrt{i}\rfloor+1)}{2\sqrt{2\pi i} C_{n/2}^\beta \cdot i^\beta} \left(\frac{i-\sqrt{i}}{i}\right)^i \left(1-\frac{i-\sqrt{i}}{n}\right)^{n-i} \\
&\geq \frac{1}{2\sqrt{2\pi} C_{n/2}^\beta i^\beta} \exp\left(i-\frac{1}{12i}\right) \exp\left(-\frac{i}{\sqrt{i}-1}\right) \exp\left(-\frac{n-i}{\frac{n}{i-\sqrt{i}}-1}\right) \\
&= \frac{1}{2\sqrt{2\pi} C_{n/2}^\beta i^\beta} \exp\left(i-\frac{1}{12i}-\frac{i}{\sqrt{i}-1}-\frac{n-i}{\frac{n}{i-\sqrt{i}}-1}\right).
\end{aligned}$$

Now we calculate

$$\begin{aligned}
i-\frac{1}{12i}-\frac{i}{\sqrt{i}-1}-\frac{n-i}{\frac{n}{i-\sqrt{i}}-1} &= -\frac{1}{12i}-\frac{i}{\sqrt{i}-1}+\frac{n\sqrt{i}}{n-i+\sqrt{i}} \\
&= -\frac{1}{12i}-\frac{n\sqrt{i}-i^2+i\sqrt{i}}{(n-i+\sqrt{i})(\sqrt{i}-1)} \geq -\frac{1}{24}-\frac{n\sqrt{i}+i\sqrt{i}}{(n-i)\sqrt{i}/4} \\
&= -\tfrac{1}{24}-4\left(\tfrac{2n}{n-i}-1\right) \geq -\tfrac{1}{24}-4\left(\tfrac{2n}{n-\sqrt{n}}-1\right) \\
&= -\tfrac{1}{24}-4\left(\tfrac{2}{\sqrt{n}-1}+1\right) \geq -\tfrac{1}{24}-4\left(\tfrac{2}{\sqrt{2}-1}+1\right) \\
&\geq -8\sqrt{2}-13,
\end{aligned}$$

where the first inequality uses that $n\sqrt{i}-i^2+i\sqrt{i} \geq 0$ for $i < \sqrt{n}$ and $\sqrt{i}-1 \geq \sqrt{i}/4$ for $i \geq 2$, the second inequality uses $i < \sqrt{n}$, and the third inequality uses $n \geq 2$. Then we know

$$P_i^\beta \geq \frac{\left(C_{n/2}^\beta\right)^{-1}}{2\sqrt{2\pi} i^\beta} e^{-8\sqrt{2}-13} \geq \frac{\beta-1}{2\sqrt{2\pi} e^{8\sqrt{2}+13}\beta} i^{-\beta},$$

where the last inequality uses (5). Hence, this lemma is proven. □

Equipping the standard GSEMO with this mutation operator $\textsc{mut}^\beta$ gives Algorithm 3, which we call GSEMO-HTM.

Algorithm 3 The GSEMO-HTM algorithm with power-law exponent $\beta > 1$

```
Generate x ∈ {0,1}^n uniformly at random, and P ← {x}
loop
    Uniformly at random select one individual x from P
    Sample α from D^β_{n/2} and generate x' via independently flipping each bit value of x with probability α/n
    if there is no y ∈ P such that x' ⪯ y then
        P = {z ∈ P | z ⋠ x'} ∪ {x'}
    end if
end loop
```

7.2 GSEMO-HTM on OneJumpZeroJump$_{n,k}$

We now analyze the runtime of the GSEMO-HTM on $\textsc{OneJumpZeroJump}_{n,k}$. We start with an estimate for the time it takes to find the inner part of the Pareto front.

Lemma 18. *Consider the GSEMO-HTM optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function. Let $a_0 \in [2k..n]$. If in a certain iteration, the point $(a_0, 2k+n-a_0)$ is covered by the current population, then all $(a, 2k+n-a)$, $a \in [2k..n]$, in the Pareto front will be found in an expected number of at most $\frac{2}{P_1^\beta} n(n-2k+3)(\ln\lceil\frac{n}{2}\rceil + 1)$ iterations.*

Proof. The proof is very similar to the one of Lemma 10. As there, we ignore any positive effect of mutation flipping more than one bit. Noting that the corresponding solution for $(a, 2k+n-a)$ in the Pareto front has $a-k$ ones, the probability to find $(a+1, 2k+n-a-1)$ in the Pareto front conditional on the population has one solution with function value $(a, 2k+n-a)$ is at least

$$\frac{1}{n-2k+3}\frac{n-a+k}{n}P_1^\beta = \frac{(n-a+k)P_1^\beta}{n(n-2k+3)}.$$

Hence, the expected time to find all $(a, 2k+n-a)$ for $a \in [a_0..n]$ is at most

$$\sum_{a=a_0}^{n-1} \frac{n(n-2k+3)}{(n-a+k)P_1^\beta}. \tag{6}$$

Similarly, the probability to find $(a-1, 2k+n-a+1)$ in the Pareto front conditional on the population containing a solution with function value $(a, 2k+n-a)$ is at least

$$\frac{1}{n-2k+3}\frac{a-k}{n}P_1^\beta = \frac{(a-k)P_1^\beta}{n(n-2k+3)},$$

and the expected time to find all $(a, 2k+n-a)$ for $a \in [2k..a_0]$ is at most

$$\sum_{a=2k+1}^{a_0} \frac{n(n-2k+3)}{(a-k)P_1^\beta}. \tag{7}$$

By comparing the individual summands, the sum of (6) and (7) is at most

$$2\sum_{a=\lfloor n/2\rfloor+k}^{n-1} \frac{n(n-2k+3)}{(n-a+k)P_1^\beta} \leq \frac{2n(n-2k+3)(\ln\lceil\frac{n}{2}\rceil+1)}{P_1^\beta}. \qquad \square$$

We now estimate the time to find the two extreme Pareto front points after the coverage of the inner part of the Pareto front.

Lemma 19. *Consider the GSEMO-HTM optimizing the* $\textsc{OneJumpZeroJump}_{n,k}$ *function. Assume that at some time, all* $(a, 2k+n-a)$, $a \in [2k..n]$, *are covered by the current population. Then the two missing* $(a, 2k+n-a)$, $a \in \{k, n+k\}$, *in the Pareto front will be found in an expected number of at most* $\frac{3}{2P_k^\beta}\binom{n}{k}(n-2k+3)$ *additional generations.*

Proof. We note that the probability to find $(k, k+n)$ in the Pareto front from one solution with function value $(2k, n)$, or to find $(k+n, k)$ in the Pareto front from one solution with function value $(n, 2k)$ is at least

$$\frac{2}{n-2k+3}\binom{n}{k}^{-1} P_k^\beta.$$

Hence, the expected time to find $(k, n+k)$ or $(k+n, k)$ is at most $\binom{n}{k}(n-2k+3)/(2P_k^\beta)$.

W.l.o.g., $(k, n+k)$ is found first. Then the probability to find $(k+n, k)$ in the Pareto front from one solution with function value $(n, 2k)$ is at least

$$\frac{1}{n-2k+3}\binom{n}{k}^{-1} P_k^\beta.$$

Hence, the expected additional time to find the last missing element in the Pareto front is at most $\binom{n}{k}(n-2k+3)/P_k^\beta$. Then this lemma is proven. $\square$

Now we are ready to show the runtime for the GSEMO-HTM on $\textsc{OneJumpZeroJump}_{n,k}$.

Theorem 20. *The expected runtime of the GSEMO-HTM optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function is at most $(n-2k+3)O(k^{\beta-0.5})C^\beta_{n/2}\frac{n^n}{k^k(n-k)^{n-k}}$.*

Proof. We first consider the time to find one $(a, 2k+n-a)$ for some $a \in [2k..n]$ in the Pareto front. If the initial search point has a number of ones in $[k, n-k]$, then $(a, 2k+n-a)$ for some $a \in [2k..n]$ in the Pareto front is already found. Otherwise, we have the initial search point with at most $k-1$ ones or zeros. W.l.o.g., we consider the initial point with $m \leq k-1$ zeros. Similar to the discussion in the proof of Theorem 12, we pessimistically add the waiting time to increase the number of zeros one by one until one individual with k zeros is found. This gives the expected number of iterations of at most

$$\sum_{m=0}^{k-1}\left(\frac{1}{n-2k+3}\frac{n-m}{n}P_1^\beta\right)^{-1} \leq \sum_{m=0}^{k-1}\frac{n(n-2k+3)}{(n-m)P_1^\beta} \leq \frac{n(n-2k+3)(\ln n+1)}{P_1^\beta}.$$

Then from Lemma 18, we know that all $(a, 2k+n-a)$ for $a \in [2k..n]$ in the Pareto front will be found in at most $\frac{2}{P_1^\beta}n(n-2k+3)(\ln(n-k)+1)$ iterations in expectation. After that, from Lemma 19, the points $(k, n+k)$ and $(n+k, k)$ enter the Pareto front in at most $\frac{2}{P_k^\beta}\binom{n}{k}(n-2k+3)$ iterations in expectation.

Hence, the expected time to cover the Pareto front is at most

$$\begin{aligned}
&(n-2k+3)\left(\frac{2n(\ln\lceil\frac{n}{2}\rceil+1)}{P_1^\beta}+\frac{2}{P_k^\beta}\binom{n}{k}+\frac{n(\ln n+1)}{P_1^\beta}\right)\\
&\quad=(n-2k+3)\left(O(1)C^\beta_{n/2}n(\ln n+1)+O(k^{\beta-0.5})C^\beta_{n/2}\frac{n^n}{k^k(n-k)^{n-k}}\right)\\
&\quad=(n-2k+3)O(k^{\beta-0.5})C^\beta_{n/2}\frac{n^n}{k^k(n-k)^{n-k}},
\end{aligned} \tag{8}$$

where the first equality uses Lemma 15. □

To give a non-asymptotic runtime bound, by Lemma 16, for $k \in [2..\lfloor\sqrt{n}\rfloor]$, we know that the first line of (8) can be calculated as

$$\begin{aligned}
&(n-2k+3)\left(\frac{2n(\ln\lceil\frac{n}{2}\rceil+1)}{P_1^\beta}+\frac{2}{P_k^\beta}\binom{n}{k}+\frac{n(\ln n+1)}{P_1^\beta}\right)\\
&\quad\leq(n-2k+3)\left(\frac{e\beta}{\beta-1}3n(\ln n+1)+\frac{4\sqrt{2\pi}e^{8\sqrt{2}+13}\beta}{\beta-1}k^\beta\frac{n!}{k!(n-k)!}\right).
\end{aligned}$$

Via the Stirling's approximation $\sqrt{2\pi}n^{n+0.5}e^{-n} \leq n! \leq en^{n+0.5}e^{-n}$ and due to the fact that $n/(n-k) \leq 2$, we know that

$$\begin{aligned}\frac{n!}{k!(n-k)!} &\leq \frac{en^{n+0.5}e^{-n}}{\left(\sqrt{2\pi}k^{k+0.5}e^{-k}\right)\left(\sqrt{2\pi}(n-k)^{n-k+0.5}e^{-(n-k)}\right)} \\ &\leq \frac{en^n}{\sqrt{2}\pi k^{k+0.5}(n-k)^{n-k}}.\end{aligned}$$

Hence, we easily obtain the following runtime for $k \in [2..\lfloor\sqrt{n}\rfloor]$.

Theorem 21. *Let $n \in \mathbb{N}$ and $k \in [2..\lfloor\sqrt{n}\rfloor]$. Then the expected runtime of the GSEMO-HTM optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function is at most $(n - 2k + 3)(\frac{4e^{8\sqrt{2}+14}\beta}{(\beta-1)\sqrt{\pi}}k^{\beta-0.5}\frac{n^n}{k^k(n-k)^{n-k}} + \frac{3e\beta}{\beta-1}n(\ln n + 1))$.*

Comparing Theorem 12 and Theorem 20 (Theorem 21), we see that the asymptotic expected runtime of the GSEMO-HTM on $\textsc{OneJumpZeroJump}_{n,k}$ is smaller than that of the GSEMO by a factor of around $k^{k+0.5-\beta}/e^k$.

8 GSEMO with Stagnation Detection

In this section, we discuss the non-trivial question of how to adapt the stagnation-detection strategy proposed by [RW22] to multiobjective optimization, which increases the mutation rate with the time that no improvement has been found. We define a stagnation-detection variant of the GSEMO that has a slightly better asymptotic performance on OneJumpZeroJump than the GSEMO with heavy-tailed mutation. In contrast to this positive result on OneJumpZeroJump, we speculate that this algorithm may have difficulties with plateaus of constant fitness.

8.1 The Stagnation-Detection Strategy of Rajabi and Witt

[RW22] proposed the following strategy to adjust the mutation rate during the run of the $(1+1)$ EA. We recall that the $(1+1)$ EA is a very elementary EA working with a parent and offspring population size of one, that is, it generates in each iteration one offspring from the unique parent via mutation and accepts it if it is at least as good as the parent. The classic mutation operator for this algorithm is standard bit-wise mutation with mutation rate $1/n$, that is, the offspring is generated by flipping each bit of the parent independently with probability $1/n$.

The main idea of the stagnation-detection approach is as follows. Assume that the $(1+1)$ EA for a longer time, say at least $10n \ln n$ iterations, has not accepted any new solution. Then, with high probability, it has generated (and rejected) all

Hamming neighbors of the parent. Consequently, there is no use to generate these solutions again and the algorithm should better concentrate on solutions further away from the parent. This can be achieved by increasing the mutation rate. For example, with a mutation rate of $\frac{2}{n}$ the rate of Hamming neighbors produced is already significantly reduced; however, Hamming neighbors can still be generated, which is important in case we were unlucky so far and missed one of them.

More generally and more precisely, to implement this stagnation-detection approach the $(1+1)$ EA maintains a counter ("failure counter") that keeps track of how long the parent individual has not given rise to a better offspring. This counter determines the current mutation rate. This dependency is governed by a safety parameter R which is recommended to be at least n. Then for $r = 1, 2, \dots$ in this order the mutation rate r/n is used for

$$T_r := \lceil 2(\tfrac{en}{r})^r \ln(nR) \rceil \tag{9}$$

iterations; the rate $\lfloor n/2 \rfloor$ is used until an improvement is found, that is, the mutation rate is never increased above $1/2$. When a strictly improving solution is found, the counter is reset to zero, and consequently, the mutation rate starts again at $\frac{1}{n}$.

[RW22] showed that the $(1+1)$ EA with this strategy optimizes $\textsc{Jump}_{n,k}$ with $k = o(n)$ in time $\Omega((\frac{en}{k})^k(1 - \frac{k^2}{n-k}))$ and $O((\frac{en}{k})^k)$. In particular, for $k = o(\sqrt{n})$, a tight (apart from constant factors independent of k and n) bound of $\Theta((\frac{en}{k})^k)$ is obtained. This is faster than the runtime of $\Theta(k^{\beta-0.5}(\frac{en}{k})^k)$ proven by [DLMN17] for the $(1+1)$ EA with heavy-tailed mutation with power-law exponent $\beta > 1$ by a factor of $k^{\beta-0.5}$. For the recommended choice $\beta = 1.5$, this factor is $\Theta(k)$.

8.2 Adaptation of the Stagnation-Detection Strategy to Multiobjective Optimization

The $(1+1)$ EA being an algorithm without a real population, it is clear that certain adaptations are necessary to use the stagnation-detection strategy in multiobjective optimization.

8.2.1 Global or Individual Failure Counters

The first question is how to count the number of unsuccessful iterations. The following two obvious alternatives exist.

Individual counters: From the basic idea of the stagnation-detection strategy, the most natural solution is to equip each individual with its own counter. Whenever an individual is chosen as parent in the GSEMO, its counter is increased by one. New solutions (but see the following subsection for an important technicality

of what "new" shall mean) entering the population (as well as the random initial individual) start with a counter value of zero.

A global counter: Algorithmically simpler is the approach to use only one global counter. This counter is increased in each iteration. When a new solution enters the population, the global counter is reset to zero.

We suspect that for many problems, both ways of counting give similar results. The global counter appears to be wasteful in the sense that when a new individual enters the population, also parents that are contained in the population for a long time re-start generating offspring with mutation rate $\frac{1}{n}$ despite the fact that they have, with very high probability, already generated as offspring all solutions close by. On the other hand, often these "old individuals" do not generate solutions that enter the population anyway, so that an optimized choice of the mutation rate is less important.

For the OneJumpZeroJump problem, it is quite clear that this second effect is dominant. A typical run starts with some individual in the inner region of the Pareto front. In a relatively short time, the whole middle region is covered, and for this it suffices that relatively recent solutions generate a suitable Hamming neighbor as offspring. The runtime is dominated by the time to find the two extremal solutions and this will almost always happen from the closest parent in the middle region of the front (regardless of whether individual counters or a global counter is used). For this reason, we analyze in the following the simpler approach using a global counter.

8.2.2 Dealing with Indifferent Solutions

One question that becomes critical when using stagnation detection is how to deal with indifferent solutions, that is, which solution to put or keep in the population in the case that an offspring y has the same (multiobjective) fitness as an individual x already in the population. Since $f(x) = f(y)$, we have $x \preceq y$ and $y \preceq x$, that is, both solutions do an equally good job in dominating others and thus in approximating the Pareto front. In early works, e.g. [LTZ+02] proposing the SEMO algorithm, such later generated indifferent solutions do not enter the population. This is partially justified by the fact that in many of the problems regarded in these works, search points with equal fitness are fully equivalent for the future run of the algorithm. We note that our OneJumpZeroJump problem also has this property, hence all results presented so far are valid regardless of how indifferent solutions are treated.

When non-equivalent search points with equal fitness exist, it is less obvious how to deal with indifferent solutions. In particular, it is clear that larger plateaus of constant fitness can be traversed much easier when a new indifferent solution is accepted as this allows to imitate a random walk behavior on the

plateau [BFH+07]. For that reason, and in analogy to single-objective optimization [JW01], it seems generally more appropriate to let a new indifferent solution enter the population, and this is also what most of the later works on the SEMO and GSEMO algorithm do [FHN10, FHN11, QTZ16, LZZZ16, BQT18, COGNS20].

Unfortunately, as mentioned in Section 1, it is not so clear how to handle indifferent solutions together with stagnation detection. In principle, when a new solution enters the population, the failure counter has to be reset to zero to reset the mutation rate to $1/n$. Otherwise, the continued use of a high mutation rate would prohibit finding good solutions in the direct neighborhood of the new solution. However, the acceptance of indifferent solutions can also lead to unwanted resets. For the OneJumpZeroJump problem, for example, it is easy to see by mathematical means that in a typical run, it will happen very frequently that an indifferent solution is generated. If this enters the population with a reset of a global failure counter (or an individual counter), then the regular resets will prevent the counters to reach interesting values. In a quick experiment for $n = 50$, $k = 4$, and a global counter, the largest counter value ever reached in this run of over 500,000,000 iterations was 5. Consequently, this SD-GSEMO was far from ever increasing the mutation rate and just imitated the classic GSEMO.

For this reason, in this work we regard the GSEMO with stagnation detection only in the variant that does not accept indifferent solutions, and we take note of the fact that thus our positive results on the stagnation-detection mechanism will not take over to problems with non-trivial plateaus of constant fitness.

8.2.3 Adjusting the Self-Adjustment

In the $(1 + 1)$ EA with stagnation detection, [RW22] increased the mutation rate from $\frac{r}{n}$ to $\frac{r+1}{n}$ once the rate $\frac{r}{n}$ has been used for T_r iterations with T_r as defined in (9). This choice ensured that any particular target solution in Hamming distance r is found in this phase with probability at least $1 - (nR)^{-2}$, see the proof of Lemma 2 in [RW22]. Since in a run of the GSEMO with current population size $|P|$ each member of the population is chosen as parent only an expected number of $T_r/|P|$ times in a time interval of length T_r, we need to adjust the parameter T_r. Not surprisingly, by taking

$$\tilde{T}_r = \lceil 2\,|P|\,(\tfrac{en}{r})^r \ln(nR) \rceil, \tag{10}$$

that is, roughly $|P|\,T_r$, the probability to generate any particular solution in Hamming distance r in phase r is at least

$$1 - \left(1 - \frac{1}{|P|}\left(\frac{r}{n}\right)^r \left(1 - \frac{r}{n}\right)^{n-r}\right)^{2|P|(en)^r \ln(nR)/r^r} \geq 1 - \frac{1}{(nR)^2},$$

which is sufficient for this purpose. Note that the population size $|P|$ changes only if a new solution enters the population and in this case the mutation rate is reset to $\frac{1}{n}$. Hence the definition of $\tilde{T}_r$, with the convention that we suppress $|P|$ in the notation to ease reading, is unambiguous.

8.2.4 The GSEMO with Stagnation Detection: SD-GSEMO

Putting the design choices discussed so far together, we obtain the following variant of the GSEMO, called SD-GSEMO. Its pseudocode is shown in Algorithm 4.

Algorithm 4 SD-GSEMO with safety parameter R

Generate $x \in \{0,1\}^n$ uniformly at random, and $P \leftarrow \{x\}$
$r \leftarrow 1$ and $u \leftarrow 0$
loop
  Uniformly and randomly select one individual x from P
  Generate x' via independently flipping each bit value of x with probability r/n
  $u \leftarrow u + 1$
  if there is no $y \in P$ such that $x' \preceq y$ **then**
    $P = \{z \in P \mid z \not\preceq x'\} \cup \{x'\}$
    $r \leftarrow 1$ and $u \leftarrow 0$
  end if
  if $u > 2|P|(\frac{en}{r})^r \ln(nR)$ **then**
    $r \leftarrow \min\{r+1, \frac{n}{2}\}$ and $u \leftarrow 0$
  end if
end loop

8.3 Runtime Analysis of the SD-GSEMO on $\textbf{OneJumpZeroJump}_{n,k}$

We now analyze the runtime of the SD-GSEMO on the OneJumpZeroJump function class. This will show that its expected runtime is by at least a factor of $\Omega(k)$ smaller than the one of the heavy-tailed GSEMO (which was a factor of $k^{\Omega(k)}$ smaller than the one of the standard GSEMO).

We extract one frequently used calculation into Lemma 23 to make our main proof clearer. The proof of Lemma 23 uses the following lemma.

Lemma 22 (Lemma 2.1 [RW20])**.** *Let $m, n \in \mathbb{N}$ and $m < n$. Then $\sum_{i=1}^{m} (\frac{en}{i})^i < \frac{n}{n-m}(\frac{en}{m})^m$.*

Lemma 23. *Let $c \geq 0$, $a, n \in \mathbb{N}$, and $a \leq \frac{n}{2}$. Then*

$$\sum_{r=a+1}^{n/2} \frac{1}{n^{c(r-a)}} \sum_{i=1}^{r} \left(\frac{en}{i}\right)^i \leq 2\left(\frac{en}{a}\right)^a \left(\frac{1}{n^{c-1}a} + \frac{1}{2n^{2c-3}a^2}\right).$$

Proof. We compute

$$\begin{aligned}
\sum_{r=a+1}^{n/2} \frac{1}{n^{c(r-a)}} \sum_{i=1}^{r} \left(\frac{en}{i}\right)^i &\leq \sum_{r=a+1}^{n/2} \frac{1}{n^{c(r-a)}} \frac{n}{n-r} \left(\frac{en}{r}\right)^r \leq 2 \sum_{r=a+1}^{n/2} \left(\frac{e}{r}\right)^r \frac{n^r}{n^{c(r-a)}} \\
&\leq 2\left(\frac{e}{a+1}\right)^{a+1} \frac{n^{a+1}}{n^c} + 2\left(\frac{n}{2} - a\right)\left(\frac{e}{a+2}\right)^{a+2} \frac{n^{a+2}}{n^{2c}} \\
&\leq 2\left(\frac{en}{a}\right)^a \left(\frac{e}{n^{c-1}a}\left(1 - \frac{1}{a+1}\right)^{a+1} + \frac{e^2}{2n^{2c-3}a^2}\left(1 - \frac{2}{a+2}\right)^{a+2}\right) \\
&\leq 2\left(\frac{en}{a}\right)^a \left(\frac{1}{n^{c-1}a} + \frac{1}{2n^{2c-3}a^2}\right),
\end{aligned}$$

where the first inequality uses Lemma 22. □

As for all algorithms discussed in this work, the most costly part of the optimization process is finding the two extremal points of the Pareto front. Borrowing many ideas from the proof of [RW20, Theorem 3.2], we show the following estimate for the time to find these extremal points when the inner part of the Pareto front is already covered.

Lemma 24. *Consider the SD-GSEMO optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function. Consider the first time T_0 that all $(a, 2k+n-a)$, $a \in [2k..n]$, are covered by the current population. Then the two possibly missing elements $(a, 2k+n-a)$, $a \in \{k, n+k\}$, of the Pareto front will be found in an expected number of at most $(n-2k+3)(\frac{en}{k})^k(\frac{3}{2} + (\frac{4k}{n} + \frac{12}{nk})\ln(nR)) + 2k$ additional generations.*

Proof. Let T' denote the first time (additional number of iterations) that one of 0^n and 1^n is found. Let phase r be the period when the SD-GSEMO uses the bit-wise mutation rate $\frac{r}{n}$, and let A_r denote the event that either 0^n or 1^n is found in phase

r. With the pessimistic assumption that neither 0^n nor 1^n is found during phase r for all $r < k$, we have

$$E[T'] = \sum_{r=1}^{n/2} E[T' \mid A_r] \Pr[A_r] \leq \sum_{r=k}^{n/2} E[T' \mid A_r] \Pr[A_r \mid r \geq k].$$

Conditional on the event A_r, the time T' to find the first extremal point includes all time spent in phases 1 to $r-1$, that is, at most $\tilde{T}_1 + \cdots + \tilde{T}_{r-1}$ with $\tilde{T}_i$ defined as in (10) for $|P| = n - 2k + 3$. In phase r, each iteration has a probability of at least $p_r = \frac{2}{n-2k+3} \left(\frac{r}{n}\right)^k \left(1 - \frac{r}{n}\right)^{n-k}$ to find an extremal point (via choosing one of the boundary points of the inner region as parent and then flipping exactly the right k bits in it). Consequently, the time spent in phase r follows a geometric law with success probability p conditional on being at most $\tilde{T}_r$. Since for any random variable X and any number T such that $\Pr[X \leq T] > 0$ we have $E[X \mid X \leq T] \leq E[X]$, the expected number of iterations spend in phase r conditional on A_r is at most $\frac{1}{p_r}$.

From this, and recalling from Corollary 8 that the population size is at most $n - 2k + 3$, we compute

$$\begin{aligned}
E[T' \mid A_k] &\leq \tilde{T}_1 + \cdots + \tilde{T}_{k-1} + \tfrac{1}{p_k} \\
&\leq \sum_{r=1}^{k-1} \left\lceil 2(n-2k+3) \left(\frac{en}{r}\right)^r \ln(nR) \right\rceil + \left(\frac{2}{n-2k+3} \left(\frac{k}{n}\right)^k \left(1 - \frac{k}{n}\right)^{n-k} \right)^{-1} \\
&\leq 2(n-2k+3)\ln(nR) \frac{n}{n-k+1} \left(\frac{en}{k-1}\right)^{k-1} + k - 1 + \frac{n-2k+3}{2} \left(\frac{en}{k}\right)^k \\
&\leq \frac{n-2k+3}{2} \left(\frac{en}{k}\right)^k \left(\frac{4k\ln(nR)}{en} \left(1 + \frac{1}{k-1}\right)^{k-1} + 1 \right) + k - 1 \\
&\leq \frac{n-2k+3}{2} \left(\frac{en}{k}\right)^k \left(\frac{4k\ln(nR)}{n} + 1 \right) + k - 1, \qquad (11)
\end{aligned}$$

where the second inequality uses Lemma 22.

We observe that for $r \geq k$, the probability that during phase r neither 0^n nor 1^n is found, is at most

$$\left(1 - \frac{2}{|P|} \left(\frac{r}{n}\right)^k \left(1 - \frac{r}{n}\right)^{n-k}\right)^{\lceil 2|P|(en)^r \ln(nR)/r^r \rceil} \leq \frac{1}{(nR)^4}. \qquad (12)$$

Hence we have $\Pr[A_r] \leq \prod_{i=k}^{r-1} \frac{1}{(nR)^4} \leq (nR)^{-4(r-k)} \leq n^{-4(r-k)}$ for $r > k$. Noting that the estimate $E[T' \mid A_r] \leq \sum_{i=1}^{r} \tilde{T}_i$ holds trivially for $r < \frac{n}{2}$, and that it also

holds for $r = \frac{n}{2}$ due to $\frac{1}{p_r} \leq \lceil 2(n-2k+3)(2e)^{n/2}\ln(nR)\rceil = \tilde{T}_{\frac{n}{2}}$, we compute

$$\begin{aligned}
\sum_{r=k+1}^{n/2} E[T' \mid A_r]\Pr[A_r] &\leq \sum_{r=k+1}^{n/2} \frac{1}{n^{4(r-k)}} \left(\sum_{i=1}^{r} \left\lceil 2(n-2k+3)\left(\frac{en}{i}\right)^i \ln(nR)\right\rceil\right) \\
&\leq 4(n-2k+3)\ln(nR)\left(\frac{en}{k}\right)^k \left(\frac{1}{n^3k} + \frac{1}{2n^5k^2}\right) + \sum_{r=k+1}^{n/2} \frac{r}{n^{4(r-k)}} \\
&\leq 4(n-2k+3)\ln(nR)\left(\frac{en}{k}\right)^k \left(\frac{1}{n^3k} + \frac{1}{2n^5k^2}\right) + \sum_{r=k+1}^{n/2} \frac{n}{2n^4} \\
&\leq 4(n-2k+3)\ln(nR)\left(\frac{en}{k}\right)^k \left(\frac{1}{n^3k} + \frac{1}{2n^5k^2}\right) + 1, \qquad (13)
\end{aligned}$$

where we use Lemma 23 for the second inequality.

Noting $\Pr[A_k] \leq 1$, together with (11) and (13), we have

$$E[T'] \leq \frac{n-2k+3}{2}\left(\frac{en}{k}\right)^k \left(\frac{4k\ln(nR)}{n} + 1 + \left(\frac{1}{n^3k} + \frac{1}{2n^5k^2}\right) 8\ln(nR)\right) + k.$$

We omit a precise statement of the very similar calculation for the time to generate the second extremal point. The only differences are that the expression $\frac{2}{n-2k+3}$ in the definition of p has to be replaced by $\frac{1}{n-2k+3}$ and that the expression $\frac{2}{|P|}$ in (12) has to be replaced by $\frac{1}{|P|}$, both due to the fact that now only one parent individual can generate the desired solution via a k-bit flip. With this, we obtain that the expected additional number of iterations to generate the second extremal point is at most

$$(n-2k+3)\left(\frac{en}{k}\right)^k \left(\frac{2k\ln(nR)}{n} + 1 + \left(\frac{1}{nk} + \frac{1}{2nk^2}\right) 4\ln(nR)\right) + k.$$

With $4(\frac{1}{nk} + \frac{1}{2nk^2}) + 8(\frac{1}{2n^3k} + \frac{1}{4n^5k^2}) \leq \frac{12}{nk}$, this lemma is proven. □

Now we are ready to show the runtime for the SD-GSEMO on the $\textsc{OneJumpZeroJump}_{n,k}$ function.

Theorem 25. *The expected runtime of the SD-GSEMO optimizing the $\textsc{OneJumpZeroJump}_{n,k}$ function is at most*

$$(n-2k+3)(\tfrac{en}{k})^k(\tfrac{3}{2} + (\tfrac{4k}{n} + \tfrac{12}{nk})\ln(nR)) + 3e(n-2k+3)(n\ln n + 2(n-2)\ln(nR)).$$

For $k = o(n/\ln(nR))$, this bound is at most

$$(n-2k+3)\Big((\tfrac{3}{2} + o(1))(\tfrac{en}{k})^k + 3en(\ln n + 2\ln(nR))\Big).$$

Proof. We first consider the expected time until we have all $(a, 2k+n-a)$, $a \in [2k..n]$, in the current Pareto front. Our main argument is the following adaptation of the analysis of this period for the standard GSEMO in the proof of Theorem 12. The main argument there was that the time spent in this period can be estimated by a sum of at most $n-2$ waiting times for the generation of a particular Hamming neighbor of a particular member of the population. The same argument, naturally, is valid for the SD-GSEMO, and fortunately, we can also reuse the computation of these waiting times. Consider the time such a subperiod now takes with the SD-GSEMO. If we condition on the subperiod ending before the rate is increased above the initial value of $1/n$, then the expected time is at most the expected time of this subperiod for the standard GSEMO (this is the same argument as used in the proof of Lemma 24). Consequently, the total time of this period for the SD-GSEMO is at most the corresponding time for the standard GSEMO $(en(n-2k+3)(\ln n+1) + 2en(n-2k+3)(\ln(n-k)+1)$ as determined in the proof of Theorem 12) plus the additional time caused by subperiods using rates r/n with $r \geq 2$.

We prove an upper bound for this time valid uniformly for all subperiods. Let phase r be the time interval of this subperiod when the SD-GSEMO uses the bit-wise mutation rate $\frac{r}{n}$. We recall that at all times the population contains an individual x such that there is a search point y that can dominate at least one individual in the current population and that y is a Hamming neighbor of x, that is, can be generated from x by flipping a single bit. Hence for all $r \geq 1$, the probability that this y is not found in phase r is at most

$$\left(1 - \frac{1}{|P|}\left(1-\frac{r}{n}\right)^{n-1}\frac{r}{n}\right)^{\lceil 2|P|(en)^r \ln(nR)/r^r \rceil} \leq \frac{1}{(nR)^2}.$$

Thus with the probability at least $1 - \frac{1}{(nR)^2}$, the desired search point y will be found in phase r. Hence, if y is not found in phase $r = 1$, then analogous to (13) in the proof of Lemma 24, also noting that the estimate $E[T' \mid A_r] \leq \sum_{i=1}^{r} \tilde{T}_i$ holds trivially for $r < \frac{n}{2}$ and it also holds for $r = \frac{n}{2}$ since $|P|2^n \leq \lceil 2|P|(2e)^{n/2}\ln(nR)\rceil = \tilde{T}_{\frac{n}{2}}$, the expected time spent in phases from 2 to $n/2$ is at most

$$\sum_{r=2}^{n/2} \frac{1}{(nR)^{2(r-1)}} \sum_{i=1}^{r} \left(2|P|\left(\frac{en}{i}\right)^i \ln(nR)\right) \leq 4en|P|\ln(nR)\frac{3}{2n} = 6e|P|\ln(nR),$$

where we use Lemma 23 for the first inequality.

With this the total additional runtime caused by the stagnation-detection strategy compared with the GSEMO to find all $(a, 2k+n-a)$, $a \in [2k..n]$ in the Pareto front is at most $(n-2)6e|P|\ln(nR)$. Therefore together with the runtime for GSEMO to find all $(a, 2k+n-a)$, $a \in [2k..n]$ in the Pareto front in the proof of

Theorem 12, we know the expected time when all $(a, 2k+n-a)$, $a \in [2k..n]$ are contained in the current Pareto front is at most

$$\begin{aligned} en(n-2k+3)(\ln n+1) + 2en(n-2k+3)(\ln(n-k)+1) \\ +(n-2)6e(n-2k+3)\ln(nR)) \\ = 3e(n-2k+3)(n\ln n + 2(n-2)\ln(nR)). \end{aligned}$$

Together with the time to find the remaining two elements in the Pareto front in Lemma 24, this theorem is proven. □

Assume that, as suggested in [RW20], the control parameter R is set to n. Then the dominating element of the upper bound in Theorem 25 becomes $(n-2k+3)(\frac{en}{k})^k(\frac{3}{2}+8(\frac{k}{n}+\frac{3}{nk})\ln n)$. Hence if $k = O(\frac{n}{\ln n})$, then the runtime of SD-GSEMO on $\textsc{OneJumpZeroJump}_{n,k}$ is $O((n-2k)(\frac{en}{k})^k)$.

9 Experiments

To understand the performance of the algorithms discussed in this work for concrete problem sizes (for which an asymptotic mathematical analysis cannot give definite answers), we now conduct a simple experimental analysis. Since the SEMO cannot find the Pareto front, we did not include it in this investigation. We did include the variant of the SD-GSEMO, denoted by SD-GSEMO-Ind, in which each individual has its own failure counter (see the discussion in Section 8). Our experimental settings are the same for all algorithms.

- $\textsc{OneJumpZeroJump}_{n,k}$: jump size $k = 4$ and problem size $n = 10, 14, \ldots, 50$.
- Maximal iterations for the termination: n^{k+3}. This number of iterations was reached in none of the experiments, i.e., the results reported are the true runtimes until the full Pareto front was found.
- $\beta = 1.5$ as suggested in [DLMN17] for the power-law distribution in GSEMO-HTM.
- $R = n$ for SD-GSEMO and SD-GSEMO-Ind as suggested in [RW20].
- 20 independent runs for each setting.

Figure 4 shows the median number of function evaluations of GSEMO, GSEMO-HTM, SD-GSEMO, and SD-GSEMO-Ind on the $\textsc{OneJumpZeroJump}_{n,k}$ function. To see how the experimental results compare

with our bounds, we also plot (i) the curve $1.5e(n-2k)n^k$ corresponding to the bounds for the GSEMO in Theorems 12 and 13, (ii) the curve $(n-2k)(en)^k/k^{k-1}$ for the GSEMO-HTM with $\beta = 1.5$; since the leading constant in Theorem 20 is implicit, we chose a constant such that the curve matches the experimental data. We observe that for $n \geq 18$, Theorem 21 have given an upper bound of $\frac{4e^{8\sqrt{2}+14}\beta}{(\beta-1)\sqrt{\pi}} \geq 6 \times 10^{11}$ for the leading constant, which is far larger than 1 and indicates that further improvements of the leading constant estimate are possible, and (iii) the curve $1.5(n-2k)(en)^k/k^k$ corresponding to the upper bound of SD-GSEMO with $R = n$ in Theorem 25.

We clearly see that these curves, in terms of shape and, where known, in terms of leading constants, match well the estimates of our theoretical runtime results. We also see, as predicted by informal considerations, the similarity of the performance of the SD-GSEMO and the SD-GSEMO-Ind. Finally, our experiments show that the different runtime behaviors are already visible for moderate (and thus realistic) problem sizes and not only in the asymptotic sense in which they were proven. In particular, we observe a performance improvement by a factor of (roughly) 5 through the use heavy-tailed mutation and by a factor of (roughly) 10 with the stagnation-detection strategy.

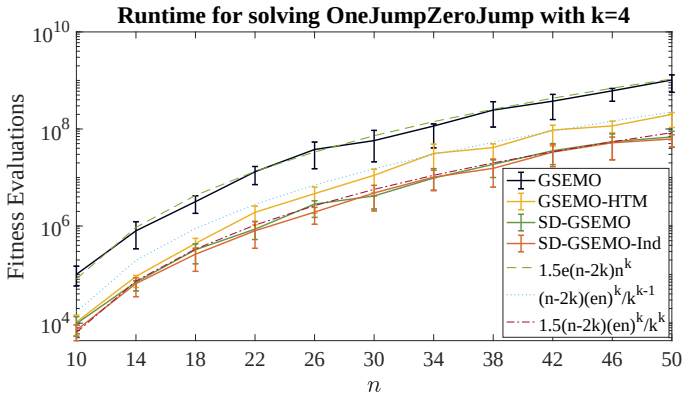

Figure 4: The median number of function evaluations (with the first and third quartiles) of GSEMO, GSEMO-HTM, SD-GSEMO, and SD-GSEMO-Ind on $\textsc{OneJumpZeroJump}_{n,k}$ with $k = 4$ and $n = 10 : 4 : 50$ in 20 independent runs.

10 Conclusion and Outlook

Noting that, different from single-objective evolutionary computation, no broadly accepted multimodal benchmarks exist in the theory of MOEAs, we proposed

the OneJumpZeroJump benchmark, which is a multiobjective analogue of the single-objective jump problem.

We use this benchmark to show that several insights from single-objective EA theory extend to the multiobjective world. Naturally, algorithms using 1-bit mutation such as the SEMO cannot optimize our benchmark, that is, cannot compute a population that covers the full Pareto front. In contrast, for all problem sizes n and jump sizes $k \in [4..\frac{n}{2}-1]$, the GSEMO covers the Pareto front in $\Theta((n-2k)n^k)$ iterations in expectation. Heavy-tailed mutation rates and the stagnation-detection mechanism of [RW22] give a runtime improvement over the standard GSEMO by a factor of $k^{\Omega(k)}$, with the stagnation-detection GSEMO slightly ahead by a small polynomial factor in k. These are the same gains as observed for the single-objective Jump benchmark with gap size k. Our experiments confirm this performance ranking already for moderate problem sizes, with the stagnation-detection GSEMO slightly more ahead than what the asymptotically small advantage suggests. On the downside, adapting the stagnation-detection mechanism to MOEAs needs taking several design choices, among which the question how to treat indifferent solutions could be difficult for problems having larger plateaus of constant fitness.

Overall, this work suggests that the recently developed ideas to cope with multimodality in single-objective evolutionary optimization can be effective in multiobjective optimization as well. In this first work in this direction, we only concentrated on mutation-based algorithms. The theory of evolutionary computation has also observed that crossover and estimation-of-distribution algorithms can be helpful in multimodal optimization. Investigating to what degree these results extend into multiobjective optimization is clearly an interesting direction for future research.

Also, we only covered very simple MOEAs in this work. Analyzing more complex MOEAs amenable to mathematical runtime anlyses, such as the decomposition-based MOEA/D [ZL07, LZZZ16] or the NSGA-II [DPAM02, ZLD22] would be highly interesting. For the MOEA/D, this would most likely require an adaptation of our benchmark problem. Since the difficult-to-find extremal points of the front are just the solutions of the single-objective sub-problems, and thus the two problems that naturally are part of the set of subproblems regarded by the MOEA/D, this algorithm might have an unfair advantage on the OneJumpZeroJump problem.

Work conducted after this research: After the publication of [DZ21], the following works have used the OneJumpZeroJump benchmark. In [DQ22], the performance of the NSGA-II, both with standard-bit mutation and with heavy-tailed mutation, on the OneJumpZeroJump benchmark was analyzed. When the population size N is at least four times the size $n-2k+3$ of the Pareto front and

$2 < k \leq n/4$, then the standard NSGA-II finds the Pareto front in time $O(Nn^k)$. Hence the NSGA-II and the GSEMO satisfy the same asymptotic runtime guarantees when N is chosen asymptotically optimal (that is, $N = \Theta(n - 2k)$). With heavy-tailed mutation, again a $k^{\Omega(k)}$ improvement is obtained. In this work, an NSGA-II was regarded that does not use crossover, but it is clear that the same upper bounds could have been shown when crossover with rate less than one was used. Doerr and Qu[DQ23b] show that the upper bound of $O(Nn^k)$ for the NSGA-II with standard-bit mutation is asymptotically tight in many cases: When $N = o(n^2/k^2)$, the runtime is $\Omega(Nn^k)$. Consequently, in this regime the NSGA-II does not profit from larger population sizes (not even when regarding the parallel runtime, that is, the number of iterations). Doerr and Qu [DQ23a] prove that uniform crossover can give super-constant speed-ups for the NSGA-II optimizing OneJumpZeroJump functions. This result is not totally surprising given that Dang et al. [DFK+18] have shown that crossover speeds up the $(\mu + 1)$ EA on single-objective jump functions, but the reason for the speed-ups shown in [DQ23a] is different (and, in fact, can be translated to different, and sometimes stronger, speed-ups for the $(\mu + 1)$ EA on jump functions). These recent results show that a jump-like benchmark, as proposed in this work, is useful in multiobjective evolutionary computation.

Acknowledgments

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

This work was also supported by Science, Technology and Innovation Commission of Shenzhen Municipality (Grant No. GXWD20220818191018001), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019A1515110177), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

References

[AD20] Denis Antipov and Benjamin Doerr. Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda))$ genetic algorithm on jump functions. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*, pages 545–559. Springer, 2020.

[ADK22] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. A rigorous runtime analysis of the $(1 + (\lambda, \lambda))$ GA on jump functions. *Algorithmica*, 84:1573–1602, 2022.

[Bäc93] Thomas Bäck. Optimal mutation rates in genetic search. In *International Conference on Genetic Algorithms, ICGA 1993*, pages 2–8. Morgan Kaufmann, 1993.

[BBD21] Riade Benbaki, Ziyad Benomar, and Benjamin Doerr. A rigorous runtime analysis of the 2-MMAS_{ib} on jump functions: ant colony optimizers can cope well with local optima. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, pages 4–13. ACM, 2021.

[BFH+07] Dimo Brockhoff, Tobias Friedrich, Nils Hebbinghaus, Christian Klein, Frank Neumann, and Eckart Zitzler. Do additional objectives make a problem harder? In *Genetic and Evolutionary Computation Conference, GECCO 2007*, pages 765–772. ACM, 2007.

[BFN08] Dimo Brockhoff, Tobias Friedrich, and Frank Neumann. Analyzing hypervolume indicator based algorithms. In *Parallel Problem Solving from Nature, PPSN 2008*, pages 651–660. Springer, 2008.

[BQ22] Chao Bian and Chao Qian. Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In *Parallel Problem Solving From Nature, PPSN 2022*, pages 428–441. Springer, 2022.

[BQT18] Chao Bian, Chao Qian, and Ke Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 1405–1411. IJCAI, 2018.

[COGNS20] Edgar Covantes Osuna, Wanru Gao, Frank Neumann, and Dirk Sudholt. Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation. *Theoretical Computer Science*, 832:123–142, 2020.

[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`.

[DFK+18] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22:484–497, 2018.

[DGN16] Benjamin Doerr, Wanru Gao, and Frank Neumann. Runtime analysis of evolutionary diversity maximization for OneMinMax. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 557–564. ACM, 2016.

[DHP22] Benjamin Doerr, Omar El Hadri, and Adrien Pinard. The $(1+(\lambda,\lambda))$ global SEMO algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2022*, pages 520–528. ACM, 2022.

[DJK08] Benjamin Doerr, Thomas Jansen, and Christian Klein. Comparing global and local mutations on bit strings. In *Genetic and Evolutionary Computation Conference, GECCO 2008*, pages 929–936. ACM, 2008.

[DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[DKV13] Benjamin Doerr, Bojana Kodric, and Marco Voigt. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2013*, pages 432–439. IEEE, 2013.

[DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 777–784. ACM, 2017.

[Doe20] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 1–87. Springer, 2020. Also available at `https://arxiv.org/abs/1801.06733`.

[Doe21] Benjamin Doerr. The runtime of the compact genetic algorithm on Jump functions. *Algorithmica*, 83:3059–3107, 2021.

[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.

[DQ22] Benjamin Doerr and Zhongdi Qu. A first runtime analysis of the NSGA-II on a multimodal problem. In *Parallel Problem Solving From Nature, PPSN 2022*, pages 399–412. Springer, 2022.

[DQ23a] Benjamin Doerr and Zhongdi Qu. The first mathematical proof that crossover gives super-constant performance gains for the NSGA-II. In *Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, 2023. To appear.

[DQ23b] Benjamin Doerr and Zhongdi Qu. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, 2023. To appear.

[Dro04] Stefan Droste. Analysis of the (1+1) EA for a noisy OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2004*, pages 1088–1099. Springer, 2004.

[DZ21] Benjamin Doerr and Weijie Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, pages 12293–12301. AAAI Press, 2021.

[FHN10] Tobias Friedrich, Nils Hebbinghaus, and Frank Neumann. Plateaus can be harder in multi-objective optimization. *Theoretical Computer Science*, 411:854–864, 2010.

[FHN11] Tobias Friedrich, Christian Horoba, and Frank Neumann. Illustration of fairness in evolutionary multi-objective optimization. *Theoretical Computer Science*, 412:1546–1556, 2011.

[FQT19] Chao Feng, Chao Qian, and Ke Tang. Unsupervised feature selection by pareto optimization. In *AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 3534–3541. AAAI, 2019.

[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.

[Gre09] Gero Greiner. Single- and multi-objective evolutionary algorithms for graph bisectioning. In *Foundations of Genetic Algorithms, FOGA 2009*, pages 29–38. ACM, 2009.

[Gut12] Walter J. Gutjahr. Runtime analysis of an evolutionary algorithm for stochastic multi-objective combinatorial optimization. *Evolutionary Computation*, 20:395–421, 2012.

[Hor09] Christian Horoba. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In *Foundations of Genetic Algorithms, FOGA 2009*, pages 113–120. ACM, 2009.

[HS18] Václav Hasenöhrl and Andrew M. Sutton. On the runtime dynamics of the compact genetic algorithm on jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 967–974. ACM, 2018.

[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.

[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.

[JS07] Jens Jägersküpper and Tobias Storch. When the plus strategy outperforms the comma strategy and when not. In *Foundations of Computational Intelligence, FOCI 2007*, pages 25–32. IEEE, 2007.

[JW01] Thomas Jansen and Ingo Wegener. Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5:589–599, 2001.

[JW02] Thomas Jansen and Ingo Wegener. The analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica*, 34:47–66, 2002.

[JW05] Thomas Jansen and Ingo Wegener. Real royal road functions – where crossover provably is essential. *Discrete Applied Mathematics*, 149:111–125, 2005.

[KB06] Rajeev Kumar and Nilanjan Banerjee. Analysis of a multiobjective evolutionary algorithm on the 0-1 knapsack problem. *Theoretical Computer Science*, 358:104–120, 2006.

[LN19] Per Kristian Lehre and Phan Trung Hai Nguyen. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 154–168. ACM, 2019.

[LTZ+02] Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, and Kalyanmoy Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving from Nature, PPSN 2002*, pages 44–53. Springer, 2002.

[LTZ04a] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing*, 3:37–51, 2004.

[LTZ04b] 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.

[LYQ16] Jing J. Liang, Caitong Yue, and Bo-Yang Qu. Multimodal multiobjective optimization: A preliminary study. In *Congress on Evolutionary Computation, CEC 2016*, pages 2454–2461. IEEE, 2016.

[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.

[Müh92] Heinz Mühlenbein. How genetic algorithms really work: mutation and hillclimbing. In *Parallel Problem Solving from Nature, PPSN 1992*, pages 15–26. Elsevier, 1992.

[Neu07] Frank Neumann. Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem. *European Journal of Operational Research*, 181:1620–1629, 2007.

[NR08] Frank Neumann and Joachim Reichel. Approximating minimum multicuts by evolutionary multi-objective algorithms. In *Parallel Problem Solving from Nature, PPSN 2008*, pages 72–81. Springer, 2008.

[NT10] Frank Neumann and Madeleine Theile. How crossover speeds up evolutionary algorithms for the multi-criteria all-pairs-shortest-path problem. In *Parallel Problem Solving from Nature, PPSN 2010, Part I*, pages 667–676. Springer, 2010.

[Pre15] Mike Preuss. *Multimodal Optimization by Means of Evolutionary Algorithms*. Springer, 2015.

[QBF20] Chao Qian, Chao Bian, and Chao Feng. Subset selection by Pareto optimization with recombination. In *AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 2408–2415. AAAI, 2020.

[QTZ16] Chao Qian, Ke Tang, and Zhi-Hua Zhou. Selection hyper-heuristics can provably be helpful in evolutionary multi-objective optimization. In *Parallel Problem Solving from Nature, PPSN 2016*, pages 835–846. Springer, 2016.

[QYZ13] Chao Qian, Yang Yu, and Zhi-Hua Zhou. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 204:99–119, 2013.

[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.

[QZTY18] Chao Qian, Yibo Zhang, Ke Tang, and Xin Yao. On multiset selection with size constraints. In *AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 1395–1402. AAAI, 2018.

[RBN20] Vahid Roostapour, Jakob Bossek, and Frank Neumann. Runtime analysis of evolutionary algorithms with biased mutation for the multi-objective minimum spanning tree problem. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 551–559. ACM, 2020.

[RNNF22] Vahid Roostapour, Aneta Neumann, Frank Neumann, and Tobias Friedrich. Pareto optimization for subset selection with dynamic cost constraints. *Artificial Intelligence*, 302:103597, 2022.

[Rud97] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovăc, 1997.

[RW20] Amirhossein Rajabi and Carsten Witt. Self-adjusting evolutionary algorithms for multimodal optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1314–1322. ACM, 2020.

[RW22] Amirhossein Rajabi and Carsten Witt. Self-adjusting evolutionary algorithms for multimodal optimization. *Algorithmica*, 84:1694–1723, 2022.

[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.

[TI20] Ryoji Tanabe and Hisao Ishibuchi. A review of evolutionary multimodal multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 24:193–200, 2020.

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

[Wit23] Carsten Witt. How majority-vote crossover and estimation-of-distribution algorithms cope with fitness valleys. *Theoretical Computer Science*, 940:18–42, 2023.

[ZD22] Weijie Zheng and Benjamin Doerr. Better approximation guarantees for the NSGA-II by using the current crowding distance. In *Genetic and Evolutionary Computation Conference, GECCO 2022*, pages 611–619. ACM, 2022.

[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.

[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.