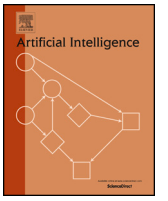
Artificial Intelligence 330 (2024) 104098

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation

Duc-Cuong Dang, Andre Opris, Dirk Sudholt *

Chair of Algorithms for Intelligent Systems, University of Passau, Passau, Germany

ARTICLE INFO

Keywords:
Evolutionary computation
Runtime analysis
Recombination
Multi-objective optimisation
Unbiased black-box algorithms
Hypermutation

ABSTRACT

Evolutionary algorithms are popular algorithms for multi-objective optimisation (also called Pareto optimisation) as they use a population to store trade-offs between different objectives. Despite their popularity, the theoretical foundation of multi-objective evolutionary optimisation (EMO) is still in its early development. Fundamental questions such as the benefits of the crossover operator are still not fully understood. We provide a theoretical analysis of the well-known EMO algorithms GSEMO and NSGA-II to showcase the possible advantages of crossover: we propose classes of "royal road" functions on which these algorithms cover the whole Pareto front in expected polynomial time if crossover is being used. But when disabling crossover, they require exponential time in expectation to cover the Pareto front. The latter even holds for a large class of black-box algorithms using any elitist selection and any unbiased mutation operator. Moreover, even the expected time to create a single Pareto-optimal search point is exponential. We provide two different function classes, one tailored for one-point crossover and another one tailored for uniform crossover, and we show that some immune-inspired hypermutations cannot avoid exponential optimisation times. Our work shows the first example of an exponential performance gap through the use of crossover for the widely used NSGA-II algorithm and contributes to a deeper understanding of its limitations and capabilities.

1. Introduction

Many optimisation problems have multiple conflicting objectives, and the aim is to find a set of Pareto-optimal solutions. Evolutionary algorithms (EAs) are general-purpose optimisers that use principles from natural evolution such as mutation, crossover (recombination) and selection to evolve a population (multi-set) of candidate solutions. EAs, such as the popular algorithm NSGA-II [18], well suited for this task as they are able to use their population to store multiple trade-offs between objectives. However, the theoretical understanding of evolutionary multi-objective optimisation (EMO) is lagging far behind its success in practice [66]. There is little understanding on how the choice of search operators and parameters affects performance in multi-objective settings.

In single-objective evolutionary optimisation, a rigorous theory has emerged over the past 25 years. It led to a better understanding of the working principles of EAs via performance guarantees and it inspired the design of novel EAs with better performance guarantees, e. g., choosing mutation rates from a heavy-tailed distribution to enable large changes [25], changing the order of crossover and mutation and amplifying the probability of improving mutations [24], parent selection preferring worse search points [11] or adapting mutation rates during the run [26].

* Corresponding author.
E-mail addresses: duccuong.dang@uni-passau.de (D.-C. Dang), andre.opris@uni-passau.de (A. Opris), dirk.sudholt@uni-passau.de (D. Sudholt).

https://doi.org/10.1016/j.artint.2024.104098
Received 27 November 2023; Received in revised form 14 February 2024; Accepted 16 February 2024

Available online 27 February 2024

In particular, the importance of the crossover operator is not well understood, despite being a topic of intensive, ongoing research in evolutionary computation and in population genetics [55]. In single-objective optimisation there is a body of works on the usefulness of crossover [62, Section 8.4] on illustrative pseudo-Boolean example problems [37,59,41,14,61,7,24] and problems from combinatorial optimisation such as shortest paths [23], graph colouring problems [30,60] and the closest string problem [63]. These works provide some explanations on how crossover can speed up optimisation, even though we are still far away from seeing a complete picture and many open questions remain.

In EMO the dynamic behaviour of EAs is even less well understood. Rigorously analysing the dynamic behaviour of EAs is hard enough in single-objective optimisation. EMO brings about additional challenges as there is no total order between search points. Search points may be incomparable due to trade-offs between different objectives. The most widely used EMO algorithm NSGA-II [18] imposes a total order by using non-dominated sorting (sorting the population according to ranks based on dominance) and a diversity score called crowding distance to break ties between equal ranks. Understanding this ranking is non-trivial, and the first rigorous runtime analyses of NSGA-II were only published at AAAI 2022 [66].

1.1. Our contribution

In this paper, we demonstrate the possible advantages of crossover for EMO by presenting examples of n-bit pseudo-Boolean functions on which the use of crossover has a drastic effect on performance. We prove using rigorous runtime analysis that well-known EMO algorithms GSEMO and NSGA-II using crossover can find the whole Pareto front in polynomial expected time, while these and large classes of black-box optimisation algorithms without crossover require exponential time with overwhelming probability and in expectation. These functions therefore can be regarded as a "royal road" for the success of crossover in multi-objective optimisation, similar to previous results for single-objective optimisation [37]. To our knowledge, this is the first proof of an exponential performance gap for the use of crossover for NSGA-II. In parallel independent work, Doerr and Qu [21] showed a polynomial gap for the use of crossover in NSGA-II.

More specifically, we propose a test function $\mathrm{RR}_{\mathrm{MO}}$ as a "royal road" for one-point crossover. Our function is inspired by Jansen and Wegener's royal road functions for single-objective optimisation [37]. The function contains a fitness valley of exponential size and very poor fitness. In order to locate the Pareto front, black-box algorithms typically have to cross this large fitness valley. This is hard for all unbiased mutation operators (operators treating bit values and all bit positions symmetrically) as a linear number of bits have to be flipped and the probability of choosing the right bits to flip in order to hit the Pareto front is exponentially small. However, the function is designed such that one-point crossover can combine the prefix with the suffix of two non-dominated solutions that represent different trade-offs stored in the population of GSEMO and NSGA-II, respectively. We prove that these algorithms can find the whole Pareto front of $\mathrm{RR}_{\mathrm{MO}}$ in expected time $\mathcal{O}(n^4)$ (see Table 1).

Since one-point crossover lacks the symmetry with respect to bit values (neighbouring bits in the bit string have a higher chance to come from the same parent), one may ask whether this positional bias is the reason for the success of algorithms with one-point crossover. To investigate this, we also consider the somatic contiguous hypermutation operator (or hypermutation for short) from artificial immune systems that has an inherent positional bias and is able to mutate a contiguous interval of bits in the bit string. We prove that also this operator is unable to find the whole Pareto front when used as the only variation operator. The reason is that the parameter r that governs the probability of flipping bits in the contiguous interval of bits must be large to discover the Pareto front and it must be small to cover the whole Pareto front afterwards. We prove that the algorithm requires exponential time for all values of r. However, when using a hybrid algorithm that may employ both hypermutations (with $r = 1$) and standard bit mutations, this combination can simulate a one-point crossover and optimise the function in expected polynomial time.

We also design a function $\mathrm{uRR}^*_{\mathrm{MO}}$ as a "royal road" for uniform crossover. As first observed by Jansen and Wegener [37] for single-objective royal road functions, designing a royal road function for uniform crossover is generally harder than for one-point crossover as uniform crossover can create an exponential number of offspring in the Hamming distance of the two parents. We extend the construction from Jansen and Wegener [37] towards multi-objective optimisation in such a way that the function makes GSEMO and NSGA-II find the Pareto front in expected time $\mathcal{O}(n^3)$, while large classes of mutation-only algorithms require exponential time. This class includes all unbiased mutation operators as well as the hypermutation operator. The function is constructed such that a large fitness valley has to be crossed. GSEMO and NSGA-II with crossover can do so by crossing two non-dominated solutions as parents whose bit values agree in the left half of the string and whose bit values are complementary in the right half of the string. The uniform crossover is likely to create an even balance of ones and zeros in the right half, while keeping all bits in the left half, on which both parents agree, intact. It thus has a good probability of finding a target set of exponential size from which it is easy to cover the Pareto front. On this function unbiased mutation fails to find the target as it must treat all bit positions symmetrically and so is not able to keep half the bits unchanged while also making large changes to the other half of bits. The same holds for the hypermutation operator if the order of bits in the function definition is permuted in such a way that the operator has no way of choosing precisely the bits that should be mutated. Since unbiased mutations and uniform crossover operator are independent from the order of bit positions, all other results hold for arbitrary permutations of bits. These are captured by the function class $\mathrm{uRR}_{\mathrm{MO}}$. All our results are summarised in Table 1.

As another technical contribution, we refine and generalise previous arguments from the analysis of NSGA-II (Lemma 4) about the survival of useful search points from function-specific arguments to general classes of fitness functions. Our work may serve as a stepping stone towards analyses of the benefits of crossover on wider problem classes, in the same way that this was achieved for single-objective optimisation.

Table 1
Summary of the results for the $\mathrm{RR}_{\mathrm{MO}}$ function and the $\mathrm{uRR}_{\mathrm{MO}}$ function class. We assume standard bit mutation with mutation rate $1/n$, hypermutation with any parameter $r \in (0,1]$ and the crossovers (if used) with probability $p_c \in (0,1)$ to be applied. The bounds on $\mathrm{E}[T]$ are in terms of number of fitness evaluations. The i-th bounds (from top to bottom) for $i \in \{1,2,4,5\}$ first appeared in [15]. The bound for hybrid mutation assumes that each operator is chosen with equal probability and that the parameter r of hypermutation is set to 1.

problem	algorithm(s)	crossover	mutation	bounds on $\mathrm{E}[T]$
$\mathrm{RR}_{\mathrm{MO}}$	GSEMO and NSGA-II	none	standard bit mutation	$n^{\Omega(n)}$ (Theorem 8)
	all elitist $(\mu+\lambda)$ black-box algorithms	none	any unbiased mutation operator	$2^{\Omega(n)}$ (Theorem 9)
	GSEMO	none	hypermutation	$2^{\Omega(n)}$ (Theorem 12)
	GSEMO	one-point	standard bit mutation	$\mathcal{O}\left(\frac{n^4}{1-p_c}+\frac{n}{p_c}\right)$ (Theorem 10)
	NSGA-II	one-point	standard bit mutation	$\mathcal{O}\left(\frac{\mu n^3}{1-p_c}+\frac{\mu^2}{np_c}\right)$ (Theorem 11)
	GSEMO	none	hybrid of hypermutation and standard bit mutation	$\mathcal{O}(n^4)$ (Theorem 14)
$\mathrm{uRR}_{\mathrm{MO}}$	GSEMO and NSGA-II	none	standard bit mutation	$n^{\Omega(n)}$ (Theorem 20)
	GSEMO and NSGA-II	none	any unbiased mutation operator	$2^{\Omega(n)}$ (Theorem 21)
	GSEMO and NSGA-II	none	hypermutation	$2^{\Omega(n)}$ (Theorem 21)
	all elitist $(\mu+\lambda)$ black-box algorithms	none	any unbiased mutation operator	$2^{\Omega(n)}$ (Theorem 21)
	all elitist $(\mu+\lambda)$ black-box algorithms	none	hypermutation	$2^{\Omega(n)}$ (Theorem 21)
	GSEMO	uniform	standard bit mutation	$\mathcal{O}\left(\frac{n^3}{p_c(1-p_c)}\right)$ (Theorem 24)
	NSGA-II	uniform	standard bit mutation	$\mathcal{O}\left(\frac{\mu n^2}{1-p_c}+\frac{\mu^2 n}{p_c}\right)$ (Theorem 25)

A preliminary version of this work appeared at AAAI 2023 (see Dang et al. [15]) where only $\mathrm{RR}_{\mathrm{MO}}$ was defined and compared against unbiased mutations. In this extended and improved manuscript we added results on hypermutations and the function $\mathrm{uRR}^*_{\mathrm{MO}}$ that requires a completely different construction from $\mathrm{RR}_{\mathrm{MO}}$, along with corresponding runtime analyses for GSEMO, NSGA-II and classes of algorithms without crossover that require exponential time.

1.2. Related work

In single-objective optimisation, the first proof that using crossover can speed up EAs was provided by Jansen and Wegener [36] for the function class $\textsc{Jump}_k$, where a fitness valley of size k has to be crossed. For $k = \log n$, the performance gap was between polynomial and superpolynomial times. These results were refined in [41,14].

Most relevant to our work is the seminal paper by Jansen and Wegener [37] that showed the first exponential performance gaps for the use of crossover. The functions were called "real royal road" functions as previous attempts at defining "royal road" functions were unsuccessful [50,31]. Jansen and Wegener [37] defined a royal road function for one-point crossover called RealRoyalRoad, which encourages EAs to evolve strings with all 1-bits gathered in a single block, and then one-point crossover can easily assemble the optimal string when choosing parents whose blocks of 1-bits are located at opposite ends of the bit string. A $(\mu+1)$ Genetic Algorithm with one-point crossover optimises RealRoyalRoad in expected time $\mathcal{O}(n^4)$, while all mutation-only EAs need exponential time with overwhelming probability. Jansen and Wegener [37] also defined a class of royal road functions for uniform crossover that can be solved by a $(\mu+1)$ Genetic Algorithm using uniform crossover in expected time $\mathcal{O}(n^3)$ whereas evolutionary algorithms without crossover need exponential time with overwhelming probability. Solving these royal road functions required crossover and a population of at least linear size. In follow-on work, Storch and Wegener [59] presented different royal road functions designed such that similar results could be shown for genetic algorithms using crossover and a small population size of only 2.

Advantages through crossover were also proven for combinatorial problems: shortest paths [23], graph colouring problems [30, 60] and the closest string problem [63]. Crossover speeds up hill climbing on $\textsc{OneMax}(x)$ that simply counts the number of ones in x by a constant factor [61,7,8,54]. A cleverly designed EA called $(1+(\lambda,\lambda))$ GA outperforms the best mutation-only EAs on OneMax by a factor of $\mathcal{O}(\log n)$ [24,19]. Finally, crossover increases robustness on difficult monotone pseudo-Boolean functions [46].

Artificial Immune System (AIS) is a design framework for algorithms inspired from by the biological immune system, and has been applied to optimisation [16]. Its mutation operators, the so-called hypermutations [40], were studied in [38] to show some advantageous speed-up for a specific problem. Jansen and Zarges [39], with references therein, gave an overview of the theoretical studies of AIS algorithms and argued that they are efficient alternatives to crossover-based EAs. Corus et al. [9] proved that the easiest function for hypermutations, called MinBlocks, is one of the hardest for standard bit mutations, they further showed that hybridising the two operators allows the optimal asymptotic performance on both OneMax and MinBlocks. A similar superiority of hypermutation was demonstrated for the NP-hard partitioning problem in [10].

Finding an efficient way to hybridise or to mix operators in an evolutionary framework is referred in the literature as the study of hyper-heuristics [4], for which runtime analysis has proven to be a valuable tool to build theoretical foundations, i. e. see Lehre and Özcan [43], Lissovoi et al. [47,48,49]. However, there also exist single-objective optimisation problems with local optima where mixing operators with a fixed distribution is not sufficient, hence more advanced approaches like self-adaptation [13,5] or memetic computing [51] are required.

Early rigorous analysis of EMO focused on simple algorithms, like SEMO (flipping a single bit for mutation) and its variant GSEMO (using standard bit mutations as a global search operator), without crossover. Laumanns et al. [42] introduced two biobjective benchmark functions LeadingOnesTrailingZeroes (LOTZ) and CountingOnesCountingZeroes (COCZ) to prove linear and sub-linear speed-ups in the expected optimisation time of two variants of SEMO over a single-individual algorithm called (1+1) EMO.

Giel and Lehre [32] gave an example of a biobjective function on which an exponential performance gap between SEMO and (1+1) EMO can be proven. Covantes Osuna et al. [12] proposed the use of diversity measures such as crowding distance in the parent selection for SEMO. They proved that the use of a power-law ranking selection to select parents ranked by the crowding distances yields a linear speed-up in the expected optimisation time for LOTZ, and for also the OneMinMax (OMM) function. Doerr and Zheng [22] introduced the OneJumpZeroJump function, which generalises Jump_k to the multi-objective setting, to show that GSEMO, in contrast to SEMO, can fully cover the Pareto set, and to further prove that the performance of GSEMO can be improved with the use of heavy-tailed mutation.

NSGA-II [18] is a practical and hugely popular reference algorithm for EMO, however its theoretical analysis only succeeded recently. Zheng, Liu, and Doerr [66] conducted the first runtime analysis of NSGA-II without crossover and proved expected time bounds $\mathcal{O}(\mu n^2)$ and $\mathcal{O}(\mu n \log n)$ to find the whole Pareto set of LOTZ and OMM, resp., if the population size μ is at least four times the size of the Pareto set. The drawback of using a too small population size was further studied in Zheng and Doerr [65]. The authors proposed an incremental procedure to compute the crowding distances as an improvement to the standard algorithm. Doerr and Qu [20] showed that heavy-tailed mutations presented for the single objective settings [25] are also highly beneficial for EMO. Cerf et al. [6] showed that NSGA-II efficiently finds good solutions for the NP-hard bi-objective minimum spanning tree problem. The upper bound $\mathcal{O}(m^2 n w_{\max} \log(n w_{\max}))$ on the expected number of iterations depends on the number of edges, m, the number of vertices, n, and the largest edge weight, $w_{\max}$.

Only a few works rigorously prove the advantages of crossover in EMO, even though experimentally they are noticeable (e. g. Doerr and Qu [20]). Qian et al. [56] proposed the REMO algorithm that initialises the population with local optima and uses crossover to quickly fill the Pareto set of example functions LOTZ and COCZ. This constitutes a speedup of order n compared to the SEMO algorithm. They later extended this work to more general function classes and multi-objective minimum spanning trees [57]. Qian et al. [58] compared two variants of GSEMO with and without crossover, called POSS and PORSS respectively, for the sub-set selection problem and showed that the recombination-based GSEMO is almost always superior. In particular, they provide an exponential performance gap for constructed instances of sub-set selection. [34,35] compared the effectiveness of immune-inspired hypermutation operators against classical mutation operators in runtime analysis of a simple multi-objective evolutionary algorithm. The results on four bi-objective optimization problems imply that the hypermutation operators can always achieve the Pareto fronts in polynomial expected runtime and also have advantage in maintaining balance between exploration and exploitation compared to the classical mutation operators. Doerr et al. [28] introduced the $(1+(\lambda,\lambda))$ GSEMO algorithm and proved that it optimizes OMM in expected time $\mathcal{O}(n^2)$, a speedup of order $\log n$ compared to GSEMO.

Bian and Qian [2] introduced a new parent selection strategy named stochastic tournament selection using crowding distance to favour diverse parents as in Covantes Osuna et al. [12] to improve the expected running time upper bounds of LOTZ, OMM and COCZ to $\mathcal{O}(n^2)$. Their analysis relies on crossover to fill the Pareto set quickly. However, the work by Covantes Osuna et al. [12] already showed that for SEMO on LOTZ the same performance guarantee can be obtained, with a similar parent selection mechanism called power-law ranking, and that crossover is not required for an $\mathcal{O}(n^2)$ bound.

Finally, Doerr and Qu [21], which appeared at the same time as our preliminary work [15], gives the joint first proof of speedups through crossover in NSGA-II. In contrast to this work, they consider a benchmark function OneJumpZeroJump based on the well-known Jump function and show a polynomial speedup through crossover. While their function is conceptually simpler than ours, the resulting speedup is polynomial, whereas ours is exponential. Their work uses uniform crossover and we provide results for both one-point crossover and uniform crossover.

2. Preliminaries

Consider maximising a function $f(x) = (f_1(x), \dots, f_d(x))$ where $f_i : \{0,1\}^n \to \mathbb{N}_0$ for $1 \le i \le d$. Given two search points $x, y \in \{0,1\}^n$, x *weakly dominates* y, denoted by $x \succeq y$, if $f_i(x) \ge f_i(y)$ for all $1 \le i \le d$; and x *dominates* y, denoted $x \succ y$, if one inequality is strict. Each solution that is not dominated by any other in $\{0,1\}^n$ is called *Pareto-optimal*. A set of these solutions that cover all possible non-dominated fitness values of f is called a *Pareto-optimal set* (or *Pareto set* for short) of f. Let $\mathcal{A}$ be an algorithm that seeks to optimise f through querying the values of $f(x)$. Such queries are called *fitness evaluations*. We assume $\mathcal{A}$ operates in *iterations* (also referred to as *generations*) and that it maintains a multi-set P_t of solutions called *the population*. By $\mu(t)$ we denote the number of fitness evaluations made in iteration t. The running time of $\mathcal{A}$ on f in terms of generations is defined as

$$T^{\text{gen}}_{\mathcal{A}(f)} := \min\{t \mid F \subseteq P_t \wedge (F \text{ is a Pareto-optimal set})\},$$

i. e. the first generation t where a Pareto-optimal set F is found in P_t, and the running time in terms of fitness evaluations is $T_{\mathcal{A}(f)} := \sum_{i=0}^{T^{\text{gen}}_{\mathcal{A}(f)}} \mu(i)$. For algorithms that make randomised decisions like EAs, we are often interested in the expected running time $\mathrm{E}\left[T_{\mathcal{A}(f)}\right]$. For a class $\mathbb{F}$ of multi-objective functions, the expected running time on $\mathbb{F}$ is considered as in the worst case performance:

$$\mathrm{E}\left[T_{\mathcal{A}(\mathbb{F})}\right] = \max_{f \in \mathbb{F}} \mathrm{E}\left[T_{\mathcal{A}(f)}\right].$$

For a search point $x \in \{0,1\}^n$, x_i denotes the i-th bit of x for $i \in \{1, \dots, n\}$. Occasionally, we also use the bit indexing in a circular sense, e. g. x_k can be also the $(k \bmod n)$-th bit of x and we identify x_0 with x_n. Let $|x|_1$ be the number of ones in x, similarly $|x|_0$ is the number of zeroes. Furthermore, we use $\mathrm{LO}(x)$, $\mathrm{TO}(x)$, $\mathrm{LZ}(x)$ and $\mathrm{TZ}(x)$ to denote, respectively, the numbers of leading ones, of trailing ones, of leading zeroes and of trailing zeros of x. For example, $x = 111001011011$ has $x_4 = 0$, $|x|_1 = 8$, $|x|_0 = 4$,

Algorithm 1 GSEMO Algorithm.

```
Initialize P_0 := {s} where s ~ Unif({0,1}^n);
for t := 0 to ∞ do
    Sample p_1 ~ Unif(P_t);
    Sample u ~ Unif([0,1]);
    if (u < p_c) then
        Sample p_2 ~ Unif(P_t);
        Create s by crossover between p_1 and p_2;
    else
        Create s as a copy of p_1;
    Create s' by mutation on s;
    if (s' is not dominated by any individual in P_t) then
        Create the next population P_{t+1} := P_t ∪ {s'};
        Remove all x ∈ P_{t+1} weakly dominated by s';
```

$\mathrm{LO}(x) = 3$, $\mathrm{TO}(x) = 2$, $\mathrm{LZ}(x) = 0$ and $\mathrm{TZ}(x) = 0$. For two strings x, y of the same length, the Hamming distance between them is denoted $\mathrm{H}(x, y)$. For strings (or bit values) $x, y, z, \dots$, their concatenation into a single string is denoted $(x, y, z, \dots)$. Let $\sigma : [n] \to [n]$ be any permutation of $[n]$ (i. e. σ is a bijection), then for any string $x \in \{0,1\}^n$ the output of permutating the bits of x according to σ is denoted $\sigma(x) := (x_{\sigma(1)}, \dots, x_{\sigma(n)})$. Let $\vec{1}$ be the vector of all ones, i. e. $\vec{1} := (1, 1, \dots)$. For any natural number n, $[n]$ denotes the set of natural numbers $\{1, \dots, n\}$. The natural logarithm is denoted $\ln n$ and that of base 2 is denoted $\log n$. The symbol $\oplus$ denotes the XOR (exclusive OR) operator.

2.1. Analytical tools

The well-known method of typical runs (see [64] Section 11) will be used for our analysis of the algorithms, and we will detail this method when it comes to the proofs. We also use drift analysis [33], which has become a standard tool in runtime analysis [45].

Theorem 1 *(Additive drift theorem [33]). Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables with a finite state space $S \subseteq \mathbb{R}^+$ such that $0 \in S$. Define $T := \inf\{t \geq 0 \mid X_t = 0\}$.*

(i) If $\exists \delta > 0$ such that $\forall s \in S \setminus \{0\}, \forall t \geq 0 :\ \mathrm{E}\left[X_t - X_{t+1} \mid X_t = s\right] \geq \delta$ then $\mathrm{E}\left[T \mid X_0\right] \leq \frac{X_0}{\delta}$.
(ii) If $\exists \delta > 0$ such that $\forall s \in S \setminus \{0\}, \forall t \geq 0 :\ \mathrm{E}\left[X_t - X_{t+1} \mid X_t = s\right] \leq \delta$ then $\mathrm{E}\left[T \mid X_0\right] \geq \frac{X_0}{\delta}$.

Note that, by the tower rule, the above statements also imply $\mathrm{E}\left[\mathrm{E}\left[T \mid X_0\right]\right] \leq \mathrm{E}\left[X_0/\delta\right] = \mathrm{E}\left[X_0\right]/\delta$ for statement (i) and likewise for (ii) with a reversed inequality.

Theorem 2 *(Negative drift theorem, e. g. see [52,53,45]). For all $a, b, \delta, \eta, r > 0$ with $a < b$, there exist $c > 0$, $n \in \mathbb{N}$ such that the following holds for all $n \geq n_0$. Suppose $(X_t)_{t\geq 0}$ is a sequence of random variables with a finite state space $S \in \mathbb{R}^+$ and adapted to filtration $\mathcal{F}_t$. Assume that $X_0 \geq bn$, and let $T_a := \min\{t \geq 0 \mid X_t \leq an\}$ be the hitting time of $S \cap [0, an]$. Assume further that for all $s \in S$ with $s > an$, for all $j \in \mathbb{N}$ and for all $t \geq 0$ the following conditions hold:*

(a) $\mathrm{E}\left[X_t - X_{t+1} \mid \mathcal{F}_t, X_t = s\right] \leq -\delta$
(b) $\Pr\left(|X_t - X_{t+1}| \geq j \mid \mathcal{F}_t, X_t = s\right) \leq \frac{r}{(1+\eta)^j}$

Then $\Pr\left(T_a \leq e^{cn}\right) \leq e^{-cn}$.

The following bounds on the amplified success rate in λ independent trials are useful.

Lemma 3 *(Lemma 10 in [1]). For every $p \in [0,1]$ and every $\lambda \in \mathbb{N}$,*

$$1 - (1-p)^\lambda \in \left[\frac{p\lambda}{1+p\lambda}, \frac{2p\lambda}{1+p\lambda}\right].$$

2.2. Algorithms

The GSEMO algorithm is shown in Algorithm 1. Starting from a randomly generated solution, in each generation a new search point s' is created as follows. With probability $p_c \in [0,1]$, a crossover is performed on parents selected uniformly at random, and then mutation is applied to the outcome. Otherwise, only mutation is applied to a parent selected uniformly at random. If the new offspring s' is not dominated by any solutions of the current population P_t then it is inserted, and all search points weakly dominated by s' are removed from the population. Thus the population size $|P_t|$ may vary over time.

The NSGA-II [18,17] is shown in Algorithm 2. In each generation, a population Q_t of μ offspring is created by applying binary tournament selection for parent selection and then applying crossover and mutation or just mutation, according to a crossover probability $p_c \in [0,1]$. Note that the crossover operator produces two offspring, and that the binary tournaments (with replacement) in line 6 use the same criteria as the sorting in line 17. The merged population R_t of both parents and offspring is then partitioned into layers $F^1_{t+1}, F^2_{t+1}, \dots$ of non-dominated solutions such that F^1_{t+1} contains all non-dominated solutions in R_t and for $i \geq 2$, F^i_{t+1} contains all non-dominated solutions in $R_t \setminus \{F^1_{t+1} \cup \dots \cup F^{i-1}_{t+1}\}$. Then crowding distances are computed within each layer. Search points of R_t are then sorted with respect to the indices of the layer that they belong to as the primary criterion, and then with the computed crowding distances as the secondary criterion. Only the μ best solutions of R_t are kept in the next generation.

Algorithm 2 NSGA-II Algorithm [18].

1: Initialize $P_0 \sim \text{Unif}((\{0,1\}^n)^\mu)$;
2: Partition P_0 into layers $F^1_0, F^2_0, \dots$ of non-dominated fitnesses, then for each layer F^i_0 compute the crowding distance $\text{CDIST}(x, F^i_0)$ for each $x \in F^i_0$;
3: **for** $t := 0$ **to** ∞ **do**
4: Initialize $Q_t := \emptyset$;
5: **for** $i := 1$ **to** $\mu/2$ **do**
6: Sample p_1 and p_2, each by a binary tournament;
7: Sample $u \sim \text{Unif}([0,1])$;
8: **if** $(u < p_c)$ **then**
9: Create s_1, s_2 by crossover on p_1, p_2;
10: **else**
11: Create s_1, s_2 as exact copies of p_1, p_2;
12: Create s'_1 by mutation on s_1 with rate $1/n$;
13: Create s'_2 by mutation on s_2 with rate $1/n$;
14: Update $Q_t := Q_t \cup \{s'_1, s'_2\}$;
15: Set $R_t := P_t \cup Q_t$;
16: Partition R_t into layers $F^1_{t+1}, F^2_{t+1}, \dots$ of non-dominated fitnesses, then for each layer F^i_{t+1} compute $\text{CDIST}(x, F^i_{t+1})$ for each $x \in F^i_{t+1}$;
17: Sort R_t lexicographically in descending order of $(1/i, \text{CDIST}(x, F^i_{t+1}))$;
18: Create the next population $P_{t+1} := (R[1], \dots, R[\mu])$;

For a set $S = (x_1, x_2, \dots, x_{|S|})$ of search points the crowding distances are calculated by computing a distance function for each objective and then summing up these values for all objectives. Thus S is sorted separately for each objective f_k $(k \leq d)$ and the first and last ranked individuals are assigned an infinite crowding distance. The remaining individuals are then assigned the differences between the values of f_k of those ranked immediate above and below the search point and normalized by the difference in f_k-values of the first and last ranked search points. Let $S_k = (x_{k_1}, x_{k_2}, \dots, x_{k_{|S|}})$ denote the elements of S sorted in descending order w. r. t. f_k, then $\text{CDIST}(x_i, S) := \sum_{k=1}^{d} \text{CDIST}_k(x_i, S)$ where

$$\text{CDIST}_k(x_{k_i}, S) := \begin{cases} \infty & \text{if } i \in \{1, |S|\}, \\ \frac{f_k(x_{k_{i-1}}) - f_k(x_{k_{i+1}})}{f_k(x_{k_1}) - f_k(x_{k_{|S|}})} & \text{otherwise.} \end{cases}$$

The following lemma shows that with a sufficiently large population size, search points of the first-ranked layer are protected between generations. No assumption about the search space is required, thus the result also holds for search spaces other than bit strings. The proof generalises the arguments from Zheng et al. [66] which correspond to the specific application of the lemma with $m = n + 1$.

Lemma 4. *Let t be one generation of the NSGA-II optimising a biobjective function $g(x) := (g_1(x), g_2(x))$. Suppose that F^1_t and F^1_{t+1} cover at most m distinct fitness vectors. Then the following holds.*

(i) At most $4m$ individuals in F^1_t have positive crowding distance.
(ii) If $\mu \geq 4m$ then for every $x \in F^1_{t+1}$ there exists a $y \in P_{t+1}$ with $g(y) = g(x)$.

Proof. (i) Note that $|M| \leq m$ for $M := \{(g_1(x), g_2(x)) \mid x \in F^1_t\}$. To obtain the result it suffices to show that for each $(a,b) \in M$ at most four individuals in F^1_t with fitness vector (a,b) have a positive crowding distance. Let $K := |F^1_t|$ and assume that $x_1, \dots, x_K$ are the individuals in F^1_t and let $S_1 = (x_{1_1}, \dots, x_{1_K})$ and $S_2 = (x_{2_1}, \dots, x_{2_K})$ be the sorting of F^1_t with respect to f_1 and f_2 respectively. Suppose there are $L \geq 1$ individuals in F^1_t with fitness (a,b). By the definition of the sorting, then there must exist $r, s \in \{1, \dots, K - L + 1\}$ such that the subsequences $(x_{1_{r+i}})_{0 \leq i \leq L-1}$ of S_1, and $(x_{2_{s+j}})_{0 \leq j \leq L-1}$ of S_2 have that $f(x_{1_{r+i}}) = f(x_{2_{s+j}}) = (a,b)$, in other words, they are the L individuals of F^t_1 with fitness (a,b). Furthermore, for each individual x from these that is not in $\{x_{1_r}, x_{1_{r+L-1}}, x_{2_s}, x_{2_{s+L-1}}\}$ there exist $2 \leq i \leq L-2$ and $2 \leq j \leq L-2$ such that $x = x_i$ and $x = x_j$, and $\text{CDIST}(x, S) = \frac{f_1(x_{1_{i-1}}) - f_1(x_{1_{i+1}})}{f_k(x_{1_1}) - f_k(x_{1_K})} + \frac{f_2(x_{2_{j-1}}) - f_2(x_{2_{j+1}})}{f_2(x_{2_1}) - f_2(x_{2_K})} = 0.$
This means only points in $\{x_{1_r}, x_{1_{r+L-1}}, x_{2_s}, x_{2_{s+L-1}}\}$ can have positive crowding distances, and the claim follows by noting that the cardinality of this set is at most 4.
(ii) Since F^1_{t+1} covers at most m distinct fitness vectors, it follows from (i) and $\mu \geq 4m$ that P_{t+1} contains every search point from F^1_{t+1}

with positive crowding distance. The statement follows since for every $x \in F^1_{t+1}$ there is $y \in F^1_{t+1}$ with positive crowding distance and $g(x) = g(y)$. □

The statement of Lemma 4 holds for any implementation of the calculation of crowding distances. However, the factor 4 in front of m there can be reduced to 2 if a stable sort algorithm is used to sort each objective, i. e. sort objective j then feed the result to the sorting of objective $j+1$. This reduction is also valid for the calculation of crowding distance in [18] that is tailored to two objectives.

2.3. The operators

We consider two well-known crossover operators in our study. In one-point crossover, which is described in Algorithm 3, a cutting point c is chosen uniformly at random. Then the offspring inherits the first c bit values from one parent and the last $n-c$ bit values from the other parent. In the uniform crossover operator each bit value is chosen independently from one of the parents chosen uniformly at random, i. e. Algorithm 4. The algorithms, which are illustrated by the pseudocode, are the two-offspring versions of these operators, i. e. two offspring solutions z and $\bar{z}$ are produced, and are used by NSGA-II. In the one-offspring versions, which are used by GSEMO, one output offspring is chosen uniformly at random from $\{z, \bar{z}\}$. These operators are known as *binary variation operators* as two inputs are required to produce the offspring solution(s).

Algorithm 3 One point crossover between strings $x, y \in \{0,1\}^n$.

1: Choose a cutting point $c \sim \text{Unif}(\{0, \dots, n\})$;
2: Create $z = (z_1, \dots, z_n)$ and $\bar{z} = (\bar{z}_1, \dots, \bar{z}_n)$
where $z_i = x_i$ for $i \le c$ and otherwise $z_i = y_i$,
and $\bar{z}_i = y_i$ for $i \le c$ and otherwise $\bar{z}_i = x_i$;
3: **return** Strings z and $\bar{z}$;

Algorithm 4 Uniform crossover between strings $x, y \in \{0,1\}^n$.

1: Sample $z = (z_1, \dots, z_n)$ and $\bar{z} = (\bar{z}_1, \dots, \bar{z}_n)$
where independently $z_i \sim \text{Unif}(\{x_i, y_i\})$,
then $\bar{z}_i = x_i$ if $z_i = y_i$ or otherwise $\bar{z}_i = y_i$, for each $i \in [n]$;
2: **return** Strings z and $\bar{z}$;

As for the mutation operators, we first consider standard bit mutation which is shown in Algorithm 5. In this operator each bit is independently copied from the parent to the offspring but has a small probability $1/n$ of being flipped (inverted). This operator is a *global operator*, in the sense that any search point can be reached, i. e. has a positive probability of being the output, from a given parent.

When analysing evolutionary algorithms on the proposed royal road functions, we will encounter scenarios in which there is no fitness gradient on a part of the bit string. This implies that a sequence of standard bit mutations will tend to create a nearly even balance between zeros and ones in the considered part, regardless of the initial search point. Since we will use this observation in several proofs, we give a lemma bounding the expected time for reaching a state where zeros and ones are nearly balanced.

Lemma 5. *Consider a sequence of search points $x(0), x(1), x(2), \dots$ from $\{0,1\}^n$ such that $x(t+1)$ results from applying a standard bit mutation (Algorithm 5) to $x(t)$. For every $x(0)$ and every constants $\varepsilon, c \in (0,1]$ the following holds. Let $L \subseteq [n]$ be a set of fixed locations in $x(t)$ with $|L| = cn =: m$ and let X_t be the number of ones in $x(t)$ from these locations, define $Y_t := \max\{X_t, m - X_t\}$, i. e. the maximum between the numbers of ones and of zeroes in $x(t)$ at L. If $T_1 := \min\{t \mid X_t \le (1+\varepsilon)m/2\}$ and $T_2 := \min\{t \mid Y_t \le (1+\varepsilon)m/2\}$, then $\mathrm{E}[T_1] \le \mathrm{E}[T_2] = \mathcal{O}(n)$, and this also holds if we swap the roles of 1 with 0 in the definition of X_t.*

Proof. The case $\varepsilon = 1$ is trivial since $T_1 = T_2 = 0$, thus in the following we assume $\varepsilon < 1$. The first inequality follows by noting that $Y_t \le (1+\varepsilon)m$ implies $X_t \le (1+\varepsilon)m$, therefore $T_1 \le T_2$ and so $\mathrm{E}[T_1] \le \mathrm{E}[T_2]$.

To prove the bound for $\mathrm{E}[T_2]$, we compare the expected change (called *drift*) of the two processes $(X_t)_{t\ge 0}$ with support $[0,m]$ and $(Y_t)_{t\ge 0}$ with support $[m/2, m]$: $\delta_t(s) := \mathrm{E}[X_t - X_{t+1} \mid X_t = s]$ with $\sigma_t(s) := \mathrm{E}[Y_t - Y_{t+1} \mid Y_t = s]$. Note that by changing the indices, i. e. by setting $k := s - i$, we can write $\delta_t(s)$ and $\sigma_t(s)$ as:

$$\delta_t(s) = \sum_{i=-(m-s)}^{s} i \Pr\left(X_t - X_{t+1} = i \mid X_t = s\right) = \sum_{k=0}^{m} (s-k) \Pr\left(X_{t+1} = k \mid X_t = s\right),$$

$$\sigma_t(s) = \sum_{i=-(m-s)}^{s-m/2} i \Pr\left(Y_t - Y_{t+1} = i \mid Y_t = s\right) = \sum_{k=m/2}^{m} (s-k) \Pr\left(Y_{t+1} = k \mid Y_t = s\right).$$

We first claim that for all $s > (1+\varepsilon)m/2$ and all $k \in [m/2, m]$, it holds that $\Pr\left(X_{t+1} = k \mid X_t = s\right) \le \Pr\left(Y_{t+1} = k \mid Y_t = s\right)$. Assume $k > m/2$, then if $X_t \ge m/2$ we get $Y_t = X_t$ and

$$\begin{aligned}\Pr\left(Y_{t+1}=k \mid Y_t=s\right) &= \Pr\left(Y_{t+1}=k \mid X_t=s\right)\\ &= \Pr\left(X_{t+1}=k \mid X_t=s\right)+\Pr\left(X_{t+1}=m-k \mid X_t=s\right)\\ &\geq \Pr\left(X_{t+1}=k \mid X_t=s\right).\end{aligned}$$

Otherwise if $X_t < m/2$ we have $Y_t = m - X_t$ and

$$\begin{aligned}\Pr\left(Y_{t+1}=k \mid Y_t=s\right) &= \Pr\left(Y_{t+1}=k \mid X_t=m-s\right)\\ &= \Pr\left(X_{t+1}=k \mid X_t=m-s\right)+\Pr\left(X_{t+1}=m-k \mid X_t=m-s\right)\\ &\geq \Pr\left(X_{t+1}=m-k \mid X_t=m-s\right)=\Pr\left(X_{t+1}=k \mid X_t=s\right),\end{aligned}$$

where the last equality follows from the symmetry of zeros and ones. When $k = m/2$ we also have that $\Pr\left(Y_{t+1}=k \mid Y_t=s\right) = \Pr\left(X_{t+1}=k \mid X_t=s\right)$. These prove the claim.

Hence for any $s > (1+\varepsilon)m/2$, we get:

$$\begin{aligned}\delta_t(s) &= \sum_{k=0}^{m}(s-k)\Pr\left(X_{t+1}=k \mid X_t=s\right)\\ &= \sum_{k=0}^{m/2-1}(s-k)\Pr\left(X_{t+1}=k \mid X_t=s\right)+\sum_{k=m/2}^{m}(s-k)\Pr\left(X_{t+1}=k \mid X_t=s\right)\\ &\leq \sum_{k=0}^{m/2-1} m\Pr\left(X_{t+1}=k \mid X_t=s\right)+\sum_{k=m/2}^{m}(s-k)\Pr\left(Y_{t+1}=k \mid Y_t=s\right)\\ &\leq (m/2)m\binom{s}{\varepsilon m/2}(1/n)^{\varepsilon m/2}+\sigma_t(s),\end{aligned}$$

where the last inequality follows since at least $\varepsilon m/2$ ones (among s) at locations L need to be flipped to have $X_{t+1} < m/2$ given that $X_t = s > (1+\varepsilon)m/2$. Using $\binom{s}{\varepsilon m/2} \leq 2^s \leq 2^m = 2^{cn}$ we have $(m/2)m\binom{s}{\varepsilon m/2}(1/n)^{\varepsilon m/2} \leq \mathcal{O}(n^2)2^{cn}n^{-\Omega(n)} = o(1)$. To compute $\delta_t(s)$, note that the number of ones and zeroes being flipped at L follows binomial laws $\mathrm{Bin}(X_t, 1/n)$ and $\mathrm{Bin}(m - X_t, 1/n)$, respectively. Therefore, for all states $s > (1+\varepsilon)m/2$, we have

$$\delta_t(s) = (1/n)(s-(m-s)) = 2s/n - c > (1+\varepsilon)c - c = c\varepsilon.$$

Thus overall, we get $\sigma_t(s) \geq \delta_t(s) - o(1) \geq c\varepsilon - o(1)$. Now consider the process Z_t which has $Z_t = Y_t - m/2$ if $Y_t > (1+\varepsilon)m/2$ and $Z_t = 0$ otherwise (i. e. $Y_t \in \{0\} \cup [m/2, (1+\varepsilon)m/2]$), then $\mathrm{E}\left[Z_t - Z_{t+1} \mid Z_t = s\right] \geq \sigma_t(s) \geq c\varepsilon - o(1)$. By Theorem 1 we have that $\mathrm{E}\left[T_2 \mid x(0)\right] \leq (m/2)/(c\varepsilon - o(1)) = \mathcal{O}(n)$ for any $x(0)$.

The final remark follows from the symmetry of zeros and ones. □

Standard bit mutation is a so-called *unbiased unary variation operator* [44]. The term *unary* means that only one input search point is used to produce the offspring, and this holds true for any mutation operator. The term *unbiased* means that the operator treats all bit positions $\{1, \ldots, n\}$ equally, as well as the bit values $\{0, 1\}$. The formal definition of unbiasedness can be found in [44], but algorithms that only make use of unbiased variation operators are resistant to the following modifications of any function that can shuffle the search space. Let $f : \{0,1\}^n \to \mathbb{R}$ be any pseudo-Boolean function on bit strings of length n, $\sigma : [n] \to [n]$ be any permutation of $[n]$ and a fixed $z \in \{0,1\}^n$, then an algorithm that only makes use of unbiased variation operators behaves the same on $f(x)$ as on $g(x) := f(\sigma(x) \oplus z)$ [44]. Furthermore, it is easy to see that this statement extends to the multi-objective setting, i. e. for any $f : \{0,1\}^n \to \mathbb{R}^d$. Owing to Lemma 1 in Doerr et al. [27], every unary unbiased variation operator on $\{0,1\}^n$ can be modelled as a two-step process: first choose a Hamming radius r from some distribution, then return a search point on the Hamming sphere $S_r(x) := \{y \in \{0,1\}^n \mid H(x,y) = r\}$ chosen uniformly at random. This is summarised in Algorithm 6.

Algorithm 5 Standard bit mutation of a string $x \in \{0,1\}^n$.

1: Sample $z = (z_1, \ldots, z_n)$ where independently for each $i \in [n]$:

$$z_i = \begin{cases} x_i & \text{with prob. } 1 - 1/n,\\ 1 - x_i & \text{with prob. } 1/n\end{cases}$$

2: **return** String z;

Algorithm 6 Unary unbiased variation operator applied on string $x \in \{0,1\}^n$.

1: Sample r from some distribution on $\{0, \ldots, n\}$;
2: Sample string $z \sim \mathrm{Unif}(S_r(x))$ where $S_r(x) := \{y \in \{0,1\}^n \mid H(x,y) = r\}$;
3: **return** String z;

Algorithm 7 Somatic contiguous hypermutation with parameter $r \in (0,1]$ of string $x \in \{0,1\}^n$.

Choose a location $c \sim \text{Unif}(\{1,\dots,n\})$;
Choose a length $\ell \sim \text{Unif}(\{0,1,\dots,n\})$;
if $\ell = 0$ **then**
    Create z as an exact copy of x;
else
    Sample $z = (z_1,\dots,z_n)$ where $z_i = x_i$ for $i \notin \{c,\dots,c+\ell-1 \bmod n\}$ and otherwise

$$z_i = \begin{cases} x_i & \text{with prob. } 1-r, \\ 1-x_i & \text{with prob. } r \end{cases}$$

return String z;

The somatic contiguous hypermutation operator has been introduced by Kelsey and Timmis [40] and is described in Jansen and Zarges [39, slide 33] as a possible alternative to a crossover operator. We consider the refined operator presented as CHM_3 in Jansen and Zarges [38] and include this operator (referred simply as *hypermutation*) in our study, and allow the algorithms to use it in place of standard bit mutation. The hypermutation operator is described in Algorithm 7. It is parametrised by a parameter $r \in (0,1]$ and the input string is copied to the output, except for a contiguous region of the string (in a circular sense), starting from a random location p and with a random length ℓ. In this region bit values are flipped independently with probability r. In the special case of two complementary parents and $r = 1$, a hypermutation produces the same distribution of offspring as a two-point crossover (where two cutting points are chosen and substrings are chosen from alternating parents). If, additionally, one end of the interval coincides with one end of the bit string, this is equivalent to a one-point crossover.

Even though every bit has the same probability $r/2$ of being flipped by this operator (see Lemma 4 in [38]), hypermutation is not an unbiased variation operator. This is due to the sense of consecutiveness induced by the operator: conditioned on flipping a bit position i, there is a higher chance for the bits next to i both to the left and to the right to be flipped, compared to the other positions.

2.4. The general framework of elitist black-box algorithms

In the following, we aim to show that crossover incorporated in GSEMO and NSGA-II leads to efficient runtimes on our proposed royal road functions, while algorithms without crossover fail badly. To this end, and to derive negative results that are as general as possible, we use a general model for elitist black-box algorithms, similar to the one introduced by Doerr and Lengler [29].

Algorithm 8 Elitist $(\mu+\lambda)$ black-box algorithm.

Initialize $P_0 \sim \text{Unif}((\{0,1\}^n)^\mu)$;
Query the ranking $\rho(P_0, f)$ induced by f;
for $t := 0$ **to** ∞ **do**
    Choose a probability distribution $D_t(P_t, \rho(P_t, f))$ on $\{\{0,1\}^n\}^\lambda$ which only depends on its two arguments;
    Sample Q_t from D_t, and set $R_t := P_t \cup Q_t$;
    Query the ranking $\rho(R_t, f)$ induced by f;
    Sort R_t according to $\rho(R_t, f)$;
    Set $P_{t+1} := (R[1],\dots,R[\mu])$;

In this model, shown in Algorithm 8, adapted from Doerr and Lengler [29], the algorithm keeps μ *best* solutions, according to the ranking function $\rho(P, f)$, it has seen so-far and can only sample new offspring solutions based on these elitist solutions. The ranking function is deterministic and provides a ranking of all search points seen so far.

Note that NSGA-II can be seen as a $(\mu+\lambda)$ GA black-box algorithm with $\lambda = \mu$. A ranking function is obtained by imposing a total order on the objective space, i. e. by sorting the population into layers of non-dominated fitnesses and further ordering search points within a layer using the crowding distance measure, as explained before. Hence, NSGA-II fits in the elitist black-box model of Algorithm 8.

3. A multi-objective royal road function for one-point crossover

Now we introduce a multi-objective royal road function designed to show the benefits of one-point crossover. The design of this function is deliberately simple as we believe that this best illustrates the necessity of the one-point crossover operator. Our construction is inspired by the "real roal road" function for one-point crossover in single-objective optimisation by Jansen and Wegener [37]. However, while their function contains a single global optimum at 1^n, our function features a set of Pareto-optimal search points at a linear Hamming distance from 1^n.

The idea behind the design of our royal road function is to encourage EMO algorithms to evolve a specific number of ones in a search point x, denoted as $|x|_1$, and then to evolve a prefix and a suffix of zeros. This is achieved by introducing two conflicting objectives that involve maximising the number of leading zeros and trailing zeros, respectively, in particular for all search points with $3n/5$ ones. There is an additional set of high-fitness search points with $4n/5$ ones that can be created easily using one-point crossover whereas common mutation operators require exponential time for the same task.

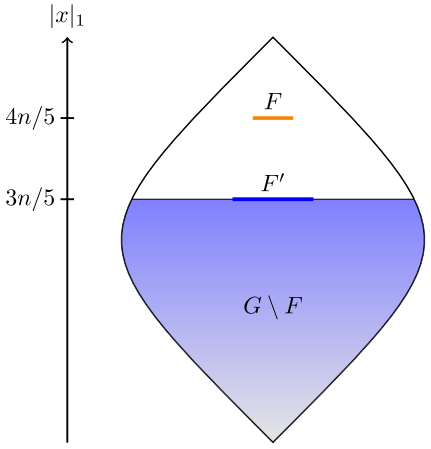


Fig. 1. A sketch of the function $\mathrm{RR}_{\mathrm{MO}}$. Here the Boolean hypercube $\{0,1\}^n$ is illustrated where the y axis shows search points with the same number of ones. The sketch illustrates the Pareto front F, the set F' and $G \setminus F$.

Definition 6. For all $n \in \mathbb{N}$ divisible by 5 the bi-objective function $\mathrm{RR}_{\mathrm{MO}} : \{0,1\}^n \to \mathbb{N}_0^2$ is defined as follows. Let $F := \{x \mid |x|_1 = 4n/5 \wedge \mathrm{LZ}(x) + \mathrm{TZ}(x) = n/5\}$ and $G := \{x \mid |x|_1 \leq 3n/5\} \cup F$, then

$$\mathrm{RR}_{\mathrm{MO}}(x) := \begin{cases} (n|x|_1 + \mathrm{TZ}(x),\ n|x|_1 + \mathrm{LZ}(x)) & \text{if } x \in G \\ (0,0) & \text{if } x \notin G. \end{cases}$$

Note that all $x \in G$ strictly dominate all $y \notin G$ as $f(0^n) = (n,n)$ and for $x \in G \setminus \{0^n\}$ both objective values are at least $n|x|_1 \geq n$. Algorithms initialising their population uniformly at random will typically start with search points having at most $3n/5$ ones, that is, only search points in G that fall into the first case of Definition 6. Then the function gives a strong fitness signal to increase the number of ones. In fact, every search point $x \in G$ dominates all search points y with $|y|_1 < |x|_1$. Every search point $x \in F$ dominates all search points $y \notin F$. Comparing two solutions $x, y \in G$ with $|x|_1 = |y|_1$, x weakly dominates y if $\mathrm{TZ}(x) \geq \mathrm{TZ}(y)$ and $\mathrm{LZ}(x) \geq \mathrm{LZ}(y)$; it strongly dominates y if one of these inequalities is strict. Thus, the set

$$F := \{0^i 1^{4n/5} 0^{n/5-i} \mid 0 \leq i \leq n/5\}$$

where all zeros contribute to either $\mathrm{TZ}(x)$ or $\mathrm{LZ}(x)$ is the Pareto-optimal set for $\mathrm{RR}_{\mathrm{MO}}$ and all search points in

$$F' := \{0^i 1^{3n/5} 0^{2n/5-i} \mid 0 \leq i \leq 2n/5\}$$

dominate all search points $y \notin F' \cup F$. Fig. 1 illustrates the structure of the search space that we have described.

By the design of the function, a solution in G with a larger number of ones always dominates those in G with fewer ones. Therefore, a set S of non-dominated solutions in G only contains solutions with the same number of ones. The following lemma bounds the number of non-dominated solutions in S more precisely.

Lemma 7. *If S is a set of non-dominated solutions in G of $\mathrm{RR}_{\mathrm{MO}}$ (i. e. for $x, y \in S$ with $x \neq y$ we neither have $x \succeq y$ nor $y \succeq x$ with respect to $\mathrm{RR}_{\mathrm{MO}}$) with k ones then $|S| \leq n - k + 1$.*

Proof. For $k = 0$ there is only one search point, thus we assume $k \geq 1$. Consider $x \in S$ with $f(x) = (kn + i, kn + j)$. If there is a search point $y \in S$, $y \neq x$, with $f(y) = (kn + i, kn + j')$ then y weakly dominates x if $j' \geq j$ and otherwise x weakly dominates y. Thus, for every value i there can only be one search point in S with an f_1-value of $kn + i$. Since the range of TZ is $0, \ldots, n - k$, the claim follows. □

3.1. Hardness of $\mathrm{RR}_{\mathrm{MO}}$ for EMOs without crossover

We first show that disabling crossover by setting $p_c = 0$ makes GSEMO and NSGA-II highly inefficient on $\mathrm{RR}_{\mathrm{MO}}$. Without crossover, both algorithms require exponential time even to discover a first Pareto-optimal search point.

Theorem 8. *The following algorithms require at least $n^{\Omega(n)}$ evaluations with probability $1 - 2^{-\Omega(n)}$ and in expectation to find any Pareto-optimal search point for $\mathrm{RR}_{\mathrm{MO}}$:*

- *GSEMO (Algorithm 1) with $p_c = 0$ and standard bit mutation,*
- *NSGA-II (Algorithm 2) with $p_c = 0$, $\mu \in \mathrm{poly}(n)$ and standard bit mutation.*

Proof. We first show the result for GSEMO. By classical Chernoff bounds the probability of initialising the algorithm with a search point of at most $3n/5$ ones, i. e. a search point in $G \setminus F$, is $1 - 2^{-\Omega(n)}$. We assume in the following that this has happened and note that then the algorithm will never accept a search point s' with fitness $(0,0)$, i. e. $s' \notin G$. Furthermore, because $p_c = 0$ the algorithm can only rely on the standard bit mutation operator to generate a search point on F. Fix a search point $y \in F$, then for each search point $x \in G \setminus F$ the Hamming distance to y is at least $H(x,y) \geq |y|_1 - |x|_1 \geq n/5$. Therefore, flipping $n/5$ specific bits is required to create y as a mutant of x, and this occurs with probability $n^{-n/5}$. Taking a union bound over all $y \in F$, the probability of creating any search point in F is at most $|F| \cdot n^{-n/5} = \mathcal{O}(n^{-n/5+1})$. By a further union bound over $n^{n/10}$ generations, the probability of creating any search point in F in the first $n^{n/10}$ generations is at most $n^{n/10} \cdot n^{-n/5+1} = n^{-\Omega(n)}$. Along with a union bound over all failure probabilities, we have shown that with probability at least $1 - 2^{-\Omega(n)}$ the goal is not reached after $n^{n/10} = n^{\Omega(n)}$ evaluations. The expected number of evaluations is at least $(1 - 2^{-\Omega(n)}) \cdot n^{-\Omega(n)} \geq n^{-\Omega(n)}$.

The proof for NSGA-II follows closely the above arguments, except for the initialisation. The probability of initialising the whole population of NSGA-II with μ search points of at most $3n/5$ ones is at least $1 - \mu \cdot 2^{-\Omega(n)} = 1 - 2^{-\Omega(n)+\log(\mu)} = 1 - 2^{-\Omega(n)}$ by a Chernoff and a union bound, using $\mu \in \text{poly}(n)$. If this occurs, then afterwards the algorithm will never accept a search point with fitness $(0,0)$ during the survival selection. Therefore, flipping $n/5$ 0s to 1s by mutation is still required to create the first Pareto-optimal solution, and we obtain the claim as above for GSEMO. □

Furthermore, we prove that *every* algorithm from the general framework of Algorithm 8 also requires exponential optimisation time in expectation to create a first Pareto-optimal point of RR_{MO} if only unary unbiased variation operators are used. This larger generality comes at the expense of a weaker lower bound where the base of the exponential function is 2 instead of n.

Theorem 9. *Every black-box algorithm that fits the model of Algorithm 8 with $\mu = \text{poly}(n)$ and only uses unary unbiased variation operators (Algorithm 6) for choosing the distribution D_t requires at least $2^{\Omega(n)}/\lambda$ generations, or $2^{\Omega(n)}$ fitness evaluations, with probability at least $1 - 2^{-\Omega(n)}$ and in expectation to find any Pareto-optimal point of RR_{MO}.*

Proof. Let $G' := \{x \mid 2n/5 \leq |x|_1 \leq 3n/5\}$ be a subset of G. Using Chernoff bounds and a union bound over $\mu = \text{poly}(n)$ initial search points, the probability of initialising the first μ search points in G' is $1 - \mu \cdot 2^{-\Omega(n)} = 1 - 2^{-\Omega(n)}$. Then, owing to elitism the algorithm will only accept points in $G' \cup F$.

For all search points $x \in G' \setminus F$ and all $y \in F$ we have $H(x,y) \geq |y|_1 - |x|_1 \geq n/5$. Moreover, since $|x|_1 \geq 2n/5$ and $|y|_1 = 4n/5$ there are at least $n/5$ bit positions i in which $x_i = y_i = 1$. Together, $n/5 \leq H(x,y) \leq 4n/5$. Let op be a unary unbiased variation operator according to Algorithm 6, then even when the radius r is chosen as $r := H(x,y)$, the probability that $\text{op}(x)$ creates y is $1/\binom{n}{H(x,y)} \leq 1/\binom{n}{n/5} \leq \frac{(n/5)^{n/5}}{n^{n/5}} = 5^{-n/5}$. Taking a union bound over all search points $y \in F$, the probability of creating any Pareto-optimal search point is at most $(n/5+1) \cdot 5^{-n/5} := p$. The probability of this happening in the first $5^{n/10}$ evaluations is at most $5^{n/10} \cdot (n/5+1) \cdot 5^{-n/5} = 2^{-\Omega(n)}$. This implies the claim on the number of evaluations. The claim on the number of generations follows since every generation makes λ evaluations. □

3.2. Use of one-point crossover implies expected polynomial optimization time on RR_{MO}

While RR_{MO} is hard for many EMO algorithms without crossover, now we show for GSEMO and for NSGA-II that they both succeed in finding the whole Pareto set of RR_{MO} in expected polynomial time.

3.2.1. Analysis of GSEMO

We start with GSEMO as the algorithm is conceptually simpler.

Theorem 10. *GSEMO (Algorithm 1) with one-point crossover, standard bit mutation and $p_c \in (0,1)$ requires at most $\mathcal{O}\left(\frac{n^4}{1-p_c} + \frac{n}{p_c}\right)$ fitness evaluations in expectation to find the whole Pareto-optimal set of RR_{MO}.*

Proof. We use the well-known method of typical runs [64, Section 11] and divide a run into several phases that reflect "typical" search dynamics. Each phase has a defined goal and we provide upper bounds on the expected time to achieve these goals. When a phase ends, the next phase starts; however, phases may be skipped if the goal of a later phase is achieved before the phase starts.

Phase 1: Create a search point in G.

By a Chernoff bound the probability that the initial search point is not in G is at most $2^{-\Omega(n)}$ as it is necessary to create a search point with more than $3n/5$ ones. Since all search points not in G have the same fitness vector $(0,0)$, while no search point in G is found, the population always consists of the latest search point. As the population then only contains a single search point, crossover, if executed, has no effect as it will recombine two identical genotypes and thus produce a clone. The process is then a repeated application of standard bit mutation, and by Lemma 5 with $c = 1$, $\varepsilon = (3/5 - 1/2)(2/c) = 1/5$, the expected time to reach a search point with at most $3/5n$ ones is $\mathcal{O}(n)$. Consequently, the expected number of generations for finding a search point in G is at most $1 + 2^{-\Omega(n)} \cdot \mathcal{O}(n) = 1 + o(1)$.

Phase 2: Create a search point with $3n/5$ ones.

Once an individual in G is found, every individual in P_t always has the same number i of ones because otherwise those with the highest number of ones will dominate and remove the others. We now compute the expected time for P_t to contain individuals with exactly $3n/5$ ones using a fitness-level argument. Note that creating a search point with a higher number of ones always removes the previous population and advances the process. Suppose that P_t contains individuals with $i \in \{0, \dots, 3n/5 - 1\}$ ones, then the number of ones can be increased by selecting an arbitrary individual as a parent, choosing not to apply crossover, and during the mutation flipping exactly one zero bit while keeping the other bits unchanged. The probability of this event is at least $(1 - p_c) \cdot \frac{n-i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{(1-p_c)(n-i)}{en}$. Thus, summing up expected waiting times of all levels i gives a bound of

$$\frac{en}{1-p_c} \sum_{i=0}^{3n/5-1} \frac{1}{n-i} = \mathcal{O}\left(\frac{n}{1-p_c}\right).$$

Phase 3: Create the first search point in F'.

To make progress towards F', it suffices to first select an individual x with a maximum value of $\mathrm{LZ}(x)+\mathrm{TZ}(x)$, denoted by $2n/5-i$, and to increase this sum while maintaining $3n/5$ ones. By Lemma 7, the probability of selecting x as parent is at least $1/|P_t| \geq 1/n$. If the algorithm then omits crossover, either flips the first 1-bit or the last 1-bit and flips one of the i 0-bits that do not contribute to $\mathrm{LZ}(x) + \mathrm{TZ}(x)$, the fitness is increased. The probability for this event is at least $\frac{1}{n} \cdot (1 - p_c) \cdot \frac{1}{n} \cdot \frac{i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{i(1-p_c)}{en^3}$, and the expected number of generations to complete this phase, by summing up the expected waiting times over all i, is at most

$$\sum_{i=1}^{2n/5} \frac{en^3}{i(1-p_c)} = \frac{en^3}{1-p_c} \sum_{i=1}^{2n/5} \frac{1}{i} = \mathcal{O}\left(\frac{n^3 \log n}{1-p_c}\right).$$

Phase 4: Cover F' entirely.

Suppose F' is not completely covered, then there must exist a missing individual z on $F' \setminus P_t$ next to a $y \in P_t \cap F'$, i. e. $|\mathrm{TZ}(z) - \mathrm{TZ}(y)| = 1$ and $\mathrm{TZ}(z) - \mathrm{TZ}(y) = \mathrm{LZ}(y) - \mathrm{LZ}(z)$. Individual z can be generated from y by omitting crossover and flipping a one at one extreme of the consecutive block of ones to a zero and a zero at the other extreme to a one while keeping the other bits unchanged. Since the parent is chosen uniformly at random, the probability of that event is $\frac{1-p_c}{n^3} \cdot \left(1 - \frac{1}{n}\right)^{n-2} \geq \frac{1-p_c}{en^3}$. As $2n/5$ such steps suffice to cover F', the expected number of generations is at most

$$\frac{en^3}{1-p_c} \cdot \frac{2n}{5} = \mathcal{O}\left(\frac{n^4}{1-p_c}\right).$$

Phase 5: Create the first search point in F.

Starting from a population $P_t = F'$, thus $|P_t| = 2n/5 + 1$, the first search point on F can be created by crossover as follows. If the algorithm picks parents $p_1 = 0^i 1^{3n/5} 0^{2n/5-i}$ with $1 \leq i \leq n/5$ and $p_2 = 0^{i+n/5} 1^{3n/5} 0^{n/5-i}$ and any cutting point $\ell \in [i + n/5, i + 3n/5]$, the result of the one-point crossover contains a single block of $4n/5$ ones and thus belongs to F. The same applies to the final offspring if the mutation following crossover does not flip any bit. The probability for selecting p_1 and p_2 is $\frac{n/5}{(|P_t|)^2} = \frac{n/5}{(2n/5+1)^2} = \Omega(1/n)$. So the probability of creating an offspring in F is at least $p_c \cdot \frac{2n/5}{n+1} \cdot \Omega(1/n) \cdot (1 - 1/n)^n = \Omega(p_c/n)$ and the expected number of generations for this to happen is $\mathcal{O}\left(\frac{n}{p_c}\right)$.

Phase 6: Cover F entirely.

The creation of the first search point on F removes all the individuals on F'. We rely on 2-bit-flip mutation steps to cover F similarly to the arguments in Phase 4. A minor difference is that the number of missing points like z is now $n/5$ since $|F| = n/5 + 1$. Nevertheless, the asymptotic number of generations to fully cover F is still $\mathcal{O}\left(\frac{n^4}{1-p_c}\right)$.

Summing up the time bounds of all phases gives a bound of $\mathcal{O}\left(\frac{n^4}{1-p_c} + \frac{n}{p_c}\right)$ on the expected number of generations (or evaluations) for GSEMO to find the Pareto set. □

3.2.2. Analysis of NSGA-II

We now turn to the analysis of NSGA-II. The search dynamics of NSGA-II are more complex than those of GSEMO due to the non-dominated sorting and the use of crowding distance. Furthermore, the uniform parent selection of GSEMO is replaced with a binary tournament selection, complicating the analysis. Unlike for GSEMO, it is not always guaranteed that all non-dominated solutions survive to the next generation, especially if the population size is chosen too small.

Theorem 11. *NSGA-II (Algorithm 2) with 1-point crossover, standard bit mutation, $p_c \in (0, 1)$ and $\mu \geq 2n + 5$ finds the whole Pareto set of $\mathrm{RR}_{\mathrm{MO}}$ in expected $\mathcal{O}\left(\frac{\mu}{np_c} + \frac{n^3}{1-p_c}\right)$ generations and $\mathcal{O}\left(\frac{\mu^2}{np_c} + \frac{\mu n^3}{1-p_c}\right)$ fitness evaluations.*

Proof. Consider the following phases of a run.

Phase 1: Create a search point in G.

By a Chernoff bound the probability that every initial individual has more than $3n/5$ ones is at most $2^{-\mu\Omega(n)}$. If this happens, the probability of creating a specific individual in G by mutation is at least n^{-n}, regardless of the input solution and the preceding

operations. By the law of total probability, the expected number of evaluations to obtain a search point in G is at most $\mu + 2^{-\mu\Omega(n)}n^n = \mu + 2^{-\Omega(n^2)} \cdot n^n = \mu + o(1)$, and these are $1 + o(1)$ generations.

Phase 2: Create a search point with $3n/5$ ones.

Suppose that the maximal number of ones in P_t is $i \in \{0, \dots, 3n/5 - 1\}$. Let x' be an individual with that number of ones, then x' is picked as a competitor in the two binary tournaments to choose $\{p_1, p_2\}$ with probability $1 - (1 - 1/\mu)^4 \geq 4/(\mu + 4)$ by Lemma 3 and this guarantees that at least one of the parents has i ones. Therefore, with probability at least $\frac{4}{\mu+4} \cdot (1 - p_c) \cdot \frac{n-i}{en} =: s_i$, one of the offspring $\{s'_1, s'_2\}$ has more than i ones, as it suffices to skip the crossover step, then flip a zero to a one while keeping the remaining bits unchanged in the mutation step. This reproduction process is repeated $\mu/2$ times, so the chance of at least one success is at least $1 - (1 - s_i)^{\mu/2} \geq \frac{s_i\mu/2}{s_i\mu/2+1}$ by the same lemma. The expected waiting time by summing up all possible values of i is no more than $\sum_{i=0}^{3n/5-1} \left(1 + \frac{2}{\mu s_i}\right)$ which is

$$\mathcal{O}(n) + \frac{2}{\mu} \sum_{i=0}^{3n/5-1} \frac{(\mu + 4)en}{4(1 - p_c)(n - i)} = \mathcal{O}\left(\frac{n}{1 - p_c}\right).$$

Phase 3: Create the first search point in F'.

After Phase 2 has completed, as long as no search point in F has been found, all non-dominated search points in P_t, that is, all search points in F_t^1, will contain $3n/5$ ones. Note that, unlike for GSEMO, the multiset F_t^1 may contain duplicates of the same search point. Let $\widehat{F_t^1}$ denote F_t^1 after removing duplicates, then $\widehat{F_t^1}$ is a set of non-dominated solutions and applying Lemma 7 to $\widehat{F_t^1}$ yields $|\widehat{F_t^1}| \leq n - 3n/5 + 1 = 2n/5 + 1$. Thus, the multiset F_t^1 contains at most $2n/5 + 1$ different fitness vectors. Applying Lemma 4 with $m := 2n/5 + 1$, statement (i) yields that at most $4 \cdot (2n/5 + 1) = 8n/5 + 4$ individuals in F_t^1 have a positive crowding distance. Let us denote the set of all individuals in F_t^1 with positive crowding distance as S_t^*.

Let $x'' \in S_t^*$ be a search point a maximum value of $\mathrm{LZ}(x) + \mathrm{TZ}(x) =: 2n/5 - i$ for $i \in \{1, \dots, 2n/5\}$. Note that x'' wins a binary tournament against all individuals in $P_t \setminus S_t^*$ by definition of the non-dominated sorting. If x'' appears as the first competitor in a binary tournament (which happens with probability $1/\mu$), and the second competitor is in $P_t \setminus S_t^*$ then x'' wins the tournament. This happens with probability at least $(\mu - |S_t^*|)/\mu \geq (\mu - 8n/5 - 4)/\mu \geq (2n/5 + 1)/(2n + 5) = 1/5$. The same holds for the disjoint event where the roles of the first and second competitor are swapped. Thus, the probability of x'' winning the tournament is at least $\frac{2}{5\mu}$.

Furthermore there are two tournaments in generating a pair of offspring. Consequently, the probability of x'' being the outcome of at least one of them is at least $1 - (1 - \frac{2}{5\mu})^2 \geq \frac{4/(5\mu)}{4/(5\mu)+1} = \frac{4}{4+5\mu}$ by Lemma 3. So, similarly to the proof of Theorem 10, the probability of increasing the maximum value of LZ + TZ in the population beyond $2n/5 - i$ is at least $\frac{4}{4+5\mu} \cdot \frac{i(1-p_c)}{en^2} =: s'_i$ during the creation of the pair. The success probability for $\mu/2$ pairs is then at least $1 - (1 - s'_i)^{\mu/2} \geq \frac{s'_i\mu/2}{s'_i\mu/2+1}$, and the expected waiting time to complete this phase is no more than $\sum_{i=1}^{2n/5} \left(1 + \frac{2}{\mu s'_i}\right)$ which is

$$\mathcal{O}(n) + \frac{2}{\mu} \sum_{i=1}^{2n/5} \frac{(4 + 5\mu)en^2}{4(1 - p_c)i} = \mathcal{O}\left(\frac{n^2 \log n}{1 - p_c}\right).$$

Phase 4: Cover F' entirely.

Let y and z be as defined *in the fourth phase* in the proof of Theorem 10 and additionally assume that y has a positive crowding distance in F_t^1. Similarly to the proof of Theorem 10 and also to the argument of the previous phase, the probability of selecting y and then winning in at least one of two tournaments, and the subsequent mutation step creating z is at least $\frac{4}{4+5\mu} \cdot \frac{1-p_c}{en^2} =: s''$, and the probability of at least one success in $\mu/2$ trials is at least $\frac{s''\mu/2}{s''\mu/2+1}$ by Lemma 3. Furthermore, applying Lemma 4 (ii) while noticing that $\mu \geq 2n + 5 > 4(2n/5 + 1)$ and F_t^1 covers at most $2n/5 + 1$ distinct fitness vectors implies that a copy of z always survives in future generations. So $2n/5$ such steps suffice to cover F' and thus the expected time to finish the phase is no more than

$$\mathcal{O}(n) + \frac{2}{\mu} \cdot \frac{(4 + 5\mu)en^2}{4(1 - p_c)} \cdot \frac{2n}{5} = \mathcal{O}\left(\frac{n^3}{1 - p_c}\right).$$

Phase 5: Create the first search point in F.

Now P_t contains all search points of F' and they are in F_t^1. There are at least $n/5$ solutions of the form $0^i 1^{3n/5} 0^{2n/5-i}$ in P_t with $i \leq n/5$ and positive crowding distance, thus they win the tournament for selecting p_1 with probability at least $2 \cdot \frac{1}{5} \cdot \frac{n/5}{\mu}$. Then it suffices to select a specific solution in $P_t \cap F'$ with positive crowding distance as p_2, i. e. with probability at least $2 \cdot \frac{1}{5} \cdot \frac{1}{\mu}$, to form compatible parents so that we have a probability of at least $p_c \cdot \frac{2n/5}{n+1} \cdot (1 - 1/n)^n$ to create one offspring in F. So in each creation of a pair, a point in F is created with probability at least $p_c \cdot \frac{4}{25} \cdot \frac{2n/5}{n+1} \cdot \frac{n/5}{\mu} \cdot \frac{1}{\mu} \cdot (1 - 1/n)^n = \Omega(\frac{np_c}{\mu^2}) =: s$, and among $\mu/2$ pairs produced at least one success occurs with probability at least $1 - (1 - s)^{\mu/2} \geq \frac{s\mu/2}{s\mu/2+1}$ (Lemma 3) and at most $1 + \frac{2}{\mu s} = \mathcal{O}\left(\frac{\mu}{np_c}\right)$ generations are required in expectation for this phase.

Phase 6: Cover F entirely.

Once a search point in F is created, the process of covering F is similar to that of covering F' with only minor differences (e. g. applying Lemma 7 with $k = 4n/5$ and then Lemma 4 with $m = n/5 + 1$). The expected number of generations in Phase 6 is $\mathcal{O}(\frac{n^3}{1-p_c})$.

Summing up expected times of all the phases gives an upper bound $\mathcal{O}\left(\frac{\mu}{np_c} + \frac{n^3}{1-p_c}\right)$ on the expected number of generations to optimise $\mathrm{RR}_{\mathrm{MO}}$. Multiplying this bound with μ gives the result in terms of fitness evaluations. □

3.3. Can hypermutation help?

So far we have discussed crossover-based algorithms, in which the 1-point crossover is able to flip a contiguous interval of bits in a search point x if the other parent has complementary bit values to those in x on that interval. Hypermutations can do the same, hence they can simulate the effect of 1-point crossover under appropriate circumstances.

Recall that hypermutation is a unary, but not unbiased operator, and thus evolutionary algorithms using hypermutation are not covered by the general lower bound from Theorem 9. Furthermore, Corus et al. [9,10] have shown the superiority of hypermutations over standard bit mutations in single-objective optimisation. Thus it is an interesting question whether hypermutation can optimise $\mathrm{RR}_{\mathrm{MO}}$ efficiently or not, especially with and without combining with the other operator. We first show that, when disabling crossover and replacing standard bit mutations with hypermutations in GSEMO, the answer is negative. The intuition for this failure is that the parameter r of the operator has to be set substantially differently between the two tasks: finding the first Pareto-optimal solution, and covering the whole Pareto front afterwards.

Theorem 12. *GSEMO (Algorithm 1) with $p_c = 0$, and using hypermutation with any parameter $r > 0$ as the only mutation operator requires at least $2^{\Omega(n)}$ evaluations with probability at least $1 - 2^{-\Omega(n)}$ and in expectation to optimise* $\mathrm{RR}_{\mathrm{MO}}$.

In order to prove Theorem 12, we first show the following lemma on the parameter ℓ of hypermutation when converting between Pareto-optimal solutions of $\mathrm{RR}_{\mathrm{MO}}$.

Lemma 13. *For any two distinct Pareto-optimal solutions x and y of* $\mathrm{RR}_{\mathrm{MO}}$*, in order to create y from x by hypermutation (Algorithm 7) it is necessary that the parameter ℓ is at least $n/5 + \mathrm{H}(x, y)/2$.*

Proof. Because x and y are Pareto optimal and distinct, we can write them as $x = 0^a 1^{4n/5} 0^{n/5-a}$ and $y = 0^b 1^{4n/5} 0^{n/5-b}$ for some integers $a, b \in [0, n/5]$, $a \neq b$. We first consider the case $a < b$, which is illustrated in Fig. 2.

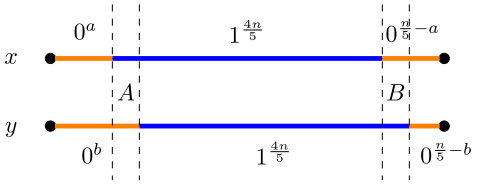


Fig. 2. Creating another Pareto-optimal solution y from x. Blocks of 1s are depicted in blue (center) while those of 0s are shown in orange (both ends of the strings).

Recall the location c and the length ℓ from the definition of hypermutations (Algorithm 7). Let A and B be the sets of positions where x and y differ on the left and on the right, respectively, as shown in the figure. It is necessary that these A, B positions are covered entirely between c and $c + \ell - 1 \bmod n$ in order to convert x to y by the operator. Note that $|A| = |B| = b - a = \mathrm{H}(x, y)/2 \leq n/5$. Consider the following choices for the parameter c of the algorithm. If c is between the end of A and the beginning of B, i. e. $b < c \leq 4n/5 + a$, then a wrap-around is required to fully cover A and particularly the end of A has to be reached, therefore $\ell \geq b + (n - c) \geq b + (n - (4n/5 + a)) = n/5 + (b - a)$. For the other choices of c, i. e. $c \leq b$ (c is before the end of A) or $c > 4n/5 + a$ (c is after the beginning of B), note that at least the whole block $1^{4n/5}$ of y has to be covered by the ℓ positions, and so $\ell \geq 4n/5 > n/5 + n/5 \geq n/5 + (b - a)$.

The result for $a > b$ follows by noting that creating x from y or creating y from x requires exactly the same set of bits (defined by $\mathrm{H}(x, y)$) to be flipped, so the necessary condition on ℓ is the same. □

With Lemma 13 in place, we now prove Theorem 12.

Proof of Theorem 12. By classical Chernoff bounds, the probability of initialising the algorithm with a search point belonging to $G \setminus F$, is $1 - 2^{-\Omega(n)}$. Assuming this occurs, then we consider two following cases.

Case of $r \leq 1/2$: In this case, we only need to focus on the expected time to create the first individual in F from those in $G \setminus F$. Fix a search point $y \in F$ and assume hypermutation is applied to a search point $x \in G \setminus F$. As $H(x, y) \geq n/5$, there is a set of $n/5$ bits that must all be flipped in order to create y. Even if all these bits are part of the interval chosen by the hypermutation, the probability

of flipping all these bits is at most $r^{n/5}$. Thus, by a union bound on the $n/5+1$ solutions of F, the probability of creating any search point in F is at most $(n/5+1)r^{n/5} \leq (n/5+1)2^{-n/5}$ and the probability of this happening in the first $2^{n/10}$ generations is $2^{-\Omega(n)}$.

Case of $r > 1/2$: Under this setting, it suffices to focus on the expected time to find a Pareto-optimal set starting from the first Pareto-optimal solution. We consider the solution $x = 0^{n/10}1^{4n/5}0^{n/10} \in F$ and distinguish two subcases:

If x is the first Pareto-optimal solution found by the algorithm, the population only contains x and we consider the expected time to create a second Pareto-optimal solution $y \neq x$. It follows from Lemma 13 that at least $n/5 + \mathrm{H}(x,y)/2$ bits are considered in x for modification to create y, however creating y only requires modifying $\mathrm{H}(x,y)$ bits. Note that $\mathrm{H}(x,y) \leq n/5$ as in y the block of ones is shifted by at most $n/10$ compared to x, and every shift by one position increases the Hamming distance by 2. Therefore there are at least $(n/5 + \mathrm{H}(x,y)/2) - \mathrm{H}(x,y) = n/5 - \mathrm{H}(x,y)/2 \geq n/5 - (n/5)/2 = n/10$ bits that are considered for modification but should not be changed to create y. By a union bound over the $n/5$ solutions in $F \setminus \{x\}$ that can be created, the probability of creating a second Pareto-optimal solution is at most $(n/5)(1-r)^{n/10} < (n/5)2^{-n/10}$.

Otherwise, x is not the first Pareto-optimal solution found by the algorithm. In this subcase, x still needs to be created from the other Pareto-optimal points in order to complete a Pareto-optimal set. By exactly the same argument as in the previous subcase, we note that in order to create x from any Pareto-optimal solution $y \neq x$, there are at least $n/10$ bits that are considered for modification but should not be flipped. The probability for hypermutation to achieve this is at most $(1-r)^{n/10} \leq 2^{-n/10}$.

Thus, in both subcases the probability of finding a Pareto-optimal set in $2^{n/20}$ generations is at most $2^{n/20} \cdot (n/5)2^{-n/10} = 2^{-\Omega(n)}$. □

The second case in the proof uses one specific Pareto-optimal solution in which the block of $4n/5$ ones is exactly centered on the bit string and argues that generating that solution is difficult. Nevertheless, the argument also holds for any Pareto-optimal solution with the block of ones sufficiently centered, i. e. at any sublinear distance, and this means that any sublinear number of points on the Pareto set can be missed and not only one. The result makes explicit use of the fact that GSEMO can only produce one offspring in every generation and all dominated solutions are removed. These properties do not hold for NSGA-II hence it remains an open question whether a similar result can be shown for NSGA-II. Finally, we also note that mixing the hypermutation with standard bit mutation, e. g. see [9], can make GSEMO efficient again, as shown in the following result.

Theorem 14. *GSEMO (Algorithm 1) with $p_c = 0$ and using a hybrid mutation operator: applying either a hypermutation with $r = 1$ or a standard bit mutation, chosen with equal probability, in line 10 of the algorithm, finds the Pareto-optimal set of $\mathrm{RR}_{\mathrm{MO}}$ in expected $\mathcal{O}(n^4)$ fitness evaluations.*

Proof. We follow the same steps and phases as the proof of Theorem 10 with the method of typical runs. Thus we mainly need to highlight the differences due to the hybrid mutation and the removal of crossover.

In Phase 1, the goal is to create the first search point in G. Again with probability $2^{-\Omega(n)}$ the algorithm can start from a search point with more than $3n/5$ ones and not in G. If this occurs, instead of using Lemma 5 like in the previous proof we argue as follows. We ignore the steps where the standard bit mutation operator is applied and pessimistically assume that these steps only create search points not in G. However, each time hypermutation is selected, i. e. with probability $1/2$, regardless of the parent, there is always a chance that the parameter ℓ of hypermutation is set to n, i. e. with probability $1/(n+1) = \Omega(1/n)$, then all bits are flipped. This turns a point with more than $3n/5$ ones (and not in G) into a point with at most $3n/5$ ones, hence in G. The expected waiting time to create a point in G by such event is then $1/((1/2)\Omega(1/n)) = \mathcal{O}(n)$, and the length of Phase 1 is again $1 + \mathcal{O}(n) \cdot 2^{-\Omega(n)} = 1 + o(1)$.

In Phase 5, we start with a population fully covering F' and the phase is completed when the first solution in F is created. Any selected parent string therefore has the form $0^a1^{3n/5}0^{2n/5-a}$ for some integer a in $[0, 2n/5]$. For $a < n/5$, if the hypermutation operator is selected, i. e. with probability $1/2$, and if its parameters are chosen as $c = a + 3n/5 + 1$ and $\ell = n/5$, i. e. with probability $(1/n)(1/(n+1)) = \Omega(1/n^2)$, then the offspring created is $0^a1^{4n/5}0^{n/5-a} \in F$. The probability of these events is at least $\Omega(1/n^2)$. Similarly for $a \geq n/5$, the same operator can also be selected and applied to create the offspring $0^{a-n/5}1^{4n/5}0^{2n/5-a} \in F$ with probability $\Omega(1/n^2)$, i. e. by choosing $c = a - n/5 + 1$ and $\ell = n/5$. So, regardless of the parent selection there is always a probability $\Omega(1/n^2)$ to create a point in F, thus the expected length of the phase is $\mathcal{O}(n^2)$.

For the remaining phases, we use the same calculations in the previous proof because the arguments in those phases only rely on the progress made by steps of only applying the standard bit mutation. Since such a step now occurs with a probability of $1/2$, these calculations give $\mathcal{O}(n), \mathcal{O}(n^3 \log n)$, $\mathcal{O}(n^4)$ and $\mathcal{O}(n^4)$ respectively for the expected lengths of Phases 2, 3, 4, and 6. The proof is completed by summing up the bounds on expected lengths of all phases. □

This result can be easily generalised for NSGA-II, i. e. without crossover but hybridising hypermutation with the standard bit mutation. However, the necessity of such a hybridisation remains an open question.

4. Multi-objective royal road functions for uniform crossover

Now we turn to designing a royal road function for uniform crossover. As already observed by Jansen and Wegener [37], royal road functions for uniform crossover are harder to design than those for one-point crossover. The reason is that one-point crossover

on two search points x and y can create $H(x,y)+1$ distinct offspring,[1] where $H(x,y)$ is the Hamming distance of x and y. In contrast, uniform crossover creates one out of $2^{H(x,y)}$ possible distinct offspring since there are two possible choices for each bit on which x and y differ. If $H(x,y)$ is small then mutation is effective at creating any such offspring and there is no large benefit from using uniform crossover. To enforce a large benefit from crossover over mutation, the Hamming distance between relevant search points to be crossed must be large. But if we want uniform crossover to create a target point (e. g. a Pareto-optimal search point), this target set must be exponentially large in the Hamming distance.

Jansen and Wegener [37] already designed a "real royal road" function for uniform crossover with the described property, albeit for single-objective optimisation. Their idea was to split the bit string into two halves and to design a fitness gradient that guides evolutionary algorithms to evolve a specific bit pattern in the left half that must be kept intact to guarantee good fitness. The right half is divided into three equal-sized parts. In the right half the algorithm populates a plateau of search points with equal fitness in such a way that the population is likely to contain many pairs of search points that agree in their left half and are complementary in their second half. A uniform crossover applied to such a pair then maintains the good bit pattern in the left half while creating a uniform random pattern in the right half. This uniform pattern has a good probability to create an equal number of zeros and ones in all sub-parts. Such search points make up a target region of exponential size that is assigned a globally optimal fitness. In contrast, mutation operators are unable to jump into the target area in the same way as they would need to keep the bit pattern in the left half intact while flipping many bits in the right half (this argument exploits the subdivison of the right half in three sub-parts). For common mutation operators that treat all bits symmetrically, this has an exponentially small probability.

Our approach for the design of our multi-objective uRR_{MO} function class extends the construction in Jansen and Wegener [37] in several ways. Firstly, we lift the construction from single-objective to multi-objective optimisation and ensure that there is a Pareto front of polynomial size. Second, we aim to ensure that the function cannot be optimised by any unary unbiased mutation operator. Third, we also aim to ensure that hypermutation cannot optimise the function effectively. To meet these aims, our design extends the construction in Jansen and Wegener [37] by sub-dividing both the left part and the right part into four parts as explained in the following. In addition, we will consider a specific permutation of bits as this prevents hypermutation from reaching the target set efficiently while uniform crossover is oblivious to the order of bits and hence performs the same on every permutation of bits.

We first specify one function, denoted uRR^*_{MO}, of the class uRR_{MO} where the bit positions of these parts are consecutive in the bit string. Later on in the section, we will generalise this function to the class, for which the consecutiveness of the bits in each part does not necessarily hold but still the results for uniform crossover hold.

Recall the notation for $[n]$, thus $[4]=\{1,2,3,4\}$.

Definition 15. Fix a natural number n which is divisible by 16. For any search point $x\in\{0,1\}^n$, we partition x into substrings x_ℓ, x_r such that $x=(x_\ell,x_r)$ where each $x_\ell:=(x^1_\ell,x^2_\ell,x^3_\ell,x^4_\ell)$ and $x_r:=(x^1_r,x^2_r,x^3_r,x^4_r)$ has length $n/2$ while x^i_ℓ, x^j_r have lengths $n/8$ for $i,j\in[4]$. On the space $\{0,1\}^{n/2}$ of these substrings, we define the following subsets:

$$\begin{aligned}
\text{For } x_\ell:\ U &:= \left\{x_\ell\in\{0,1\}^{n/2} \mid \forall i\in[4]:\ |x^i_\ell|_1\in[n/24,n/12]\right\},\\
P &:= \left\{x_\ell\in\{0,1\}^{n/2} \mid \text{LO}(x_\ell)+\text{TZ}(x_\ell)=n/2\right\},\\
\text{For } x_r:\ C &:= \left\{x_r\in\{0,1\}^{n/2} \mid \text{LO}(x_r)+\text{TZ}(x_r)=n/2\vee \text{LZ}(x_r)+\text{TO}(x_r)=n/2\right\},\\
T &:= \left\{x_r\in\{0,1\}^{n/2} \mid \forall j\in[4]:\ |x^j_r|_1=|x^j_r|_0=(n/8)/2=n/16\right\}.
\end{aligned}$$

The set P forms a *path* of search points $1^i0^{n/2-i}$ of cardinality $n/2+1$. It is easy to see that if $x_\ell\in P$ then there exists $y_\ell\in P:\ \text{H}(x_\ell,y_\ell)=1$. The set C was called a *circle* in Jansen and Wegener [37] and it is a closed Hamming path of length n. If $x_r\in C$ then there exists $y_r\in C:\ \text{H}(x_r,y_r)=1$. The set T is a *target* that contains all search points for which all sub-parts of x_r have an equal number of ones and zeros. The cardinality of T is $|T|=\binom{n/8}{n/16}^4=2^{n/2}\cdot(\Theta(1/(\sqrt{n/8})))^4=2^{n/2}\cdot\Theta(1/n^2)$ as the size of the largest binomial coefficient is $\binom{n/8}{n/16}=2^{n/8}\cdot\Theta(1/(\sqrt{n/8}))$. Finally, the set U contains an exponential number of search points for which the ratio of zeros and ones is fairly *uniform* in all sub-parts of x_ℓ. A bit string chosen uniformly at random from $\{0,1\}^{n/2}$ is in U with overwhelming probability, by a straightforward application of Chernoff bounds and a union bound over all four sub-parts.

We will use P and T to define the Pareto-optimal solutions of uRR^*_{MO}. These sets typically have to be reached from search points x with $x_\ell\in U$ and $x_r\in C$. The following lemma shows that U and P have a large Hamming distance from one another, and the same applies to C and T. Hence, large changes are required to create a Pareto-optimal solution from x.

Lemma 16. *The following properties hold for the subsets of Definition 15:*

(i) $\forall x_\ell\in U,\forall y_\ell\in P:\ \text{H}(x_\ell,y_\ell)\in[n/8,3n/8]$.
(ii) $\forall x_r\in C,\forall y_r\in T:\ \text{H}(x_r,y_r)\in[3n/16,5n/16]$.

[1] For instance, a uniform crossover of 1^n and 0^n can create $n+1=H(1^n,0^n)+1$ offspring 1^i0^{n-i} for $i\in\{0,\dots,n\}$. Since bits on which both parents agree are always copied to the offspring, these bits can be ignored in this consideration and the above argument applies to arbitrary parents x,y and the subset of $H(x,y)$ bits that differ in x and y.

Proof. For (i), the definition of U implies $|x_\ell^j|_1 \in [n/24, n/12]$ and thus also $|x_\ell^j|_0 \in [n/8 - 2n/24, n/8 - n/24] = [n/24, n/12]$ for every $j \in [4]$. By the definition of the path P, $y_\ell = 1^a 0^{n/2-a}$ for some integer $a \in [0, n/2]$ and thus we have that $y_\ell^j \in \{0^{n/8}, 1^{n/8}\}$ for at least three distinct indices $j \in [4]$. Hence, for such a j we obtain $n/12 \geq \mathrm{H}(x_\ell^j, y_\ell^j) \geq n/24$ while for the remaining block i (that differs from these three j) we get $n/8 \geq \mathrm{H}(x_\ell^i, y_\ell^i) \geq 0$. Adding the distances of each block gives $\mathrm{H}(x_\ell, y_\ell) \in [3(n/24), 3(n/12) + n/8] = [n/8, 3n/8]$.

Similarly for (ii), by the definition of the circle C either $x_r = 1^a 0^{n/2-a}$ or $x_r = 0^a 1^{n/2-a}$ for some integer $a \in [0, n/2]$, we have that there are at least three indices $j \in [4]$ such that $x_r^j \in \{1^{n/8}, 0^{n/8}\}$, while by the definition of the target T we have $|y_r^j|_1 = |y_r^j|_0 = n/16$. Therefore, for these three j indices we have $\mathrm{H}(x_r^j, y_r^j) = n/16$. Let i be the remaining index, then $n/8 \geq \mathrm{H}(x_r^m, y_r^m) \geq 0$, and adding the distances implies $\mathrm{H}(x_r, y_r) \in [3(n/16), 3(n/16) + n/8] = [3n/16, 5n/16]$. □

Definition 17. Based on the partition and the subsets from Definition 15, we define a function

$$f(x) = (f_1(x), f_2(x)) := \begin{cases} (\mathrm{LO}(x_r), \frac{n}{2} + \mathrm{TZ}(x_r)) & \text{if } \mathrm{LO}(x_r) \neq 0, \\ (\frac{n}{2} + \mathrm{LZ}(x_r), \mathrm{TO}(x_r)) & \text{otherwise.} \end{cases}$$

Then our $\mathrm{uRR}^*_{\mathrm{MO}}$ function is:

$$\mathrm{uRR}^*_{\mathrm{MO}}(x) := \begin{cases} f(x) & \text{if } x_\ell \in U \wedge x_r \notin C, \\ f(x) + (2n - |x_\ell|_1) \cdot \vec{1} & \text{if } x_r \in C, \\ (\mathrm{LO}(x_\ell), \mathrm{TZ}(x_\ell)) + 3n \cdot \vec{1} & \text{if } x_\ell \in P \wedge x_r \in T, \\ (0,0) & \text{otherwise.} \end{cases}$$

Algorithms that initialise their population uniformly at random will typically start with search points x with $x_\ell \in U$ and $x_r \notin C$, i. e. with fitness $f(x)$. Then the fitness gives signals to gradually generate a search point x with $x_r \in C$. After this the number of ones in x_ℓ will be minimized since the fitness increases in both components with decreasing $|x_\ell|_1$. This gives a search point x with $x_\ell = 0^{n/2}$ and $x_r \in C$. The fitness of x is then at least $2n$ in every objective and thus x strictly dominates every search point y with fitness $f(y)$ or smaller. Moreover, two search points x, y with $x \neq y$, $x_r, y_r \in C$ and $x_\ell, y_\ell = 0^{n/2}$ are incomparable with respect to $\mathrm{uRR}^*_{\mathrm{MO}}$. All search points z with $z_\ell \in P$ and $z_r \in T$ have fitness at least $3n$ in each objective and thus dominates every other search point of the search space, hence they are Pareto-optimal. Particularly, an optimal search point z^* with $z^*_\ell = 0^{n/2}$ and $z^*_r \in T$ can be created by uniform crossover between the two search points x, y previously mentioned if their x_r and y_r parts are complement strings.

Since two distinct search points with fitness at least $3n$ in each objective are incomparable, the set of all possible non-dominated fitness vectors of $\mathrm{uRR}^*_{\mathrm{MO}}$ is

$$\{(3n + k, 3n + n/2 - k) \mid k \in \{0, \dots, n/2\}\},$$

and the set of all Pareto-optimal solutions for $\mathrm{uRR}^*_{\mathrm{MO}}$ is $W := \{x \in \{0,1\}^n \mid x_\ell \in P \wedge x_r \in T\}$. Thus we can bound the number of non-dominated solutions of $\mathrm{uRR}^*_{\mathrm{MO}}$ contained in any population as follows.

Lemma 18. *If S is a set of non-dominated solutions of $\mathrm{uRR}^*_{\mathrm{MO}}$ (i. e. for $x, y \in S$ with $x \neq y$ we neither have $x \succeq y$ nor $y \succeq x$ with respect to $\mathrm{uRR}^*_{\mathrm{MO}}$) with positive fitness then $|S| \leq n$.*

Proof. Let $h := \mathrm{uRR}^*_{\mathrm{MO}}$. As in the proof of Lemma 7 we see that two search points in S have different h_1-values. We may assume that $h(x) \neq (0,0)$ for every x since otherwise S consists of only one point. Note that for each $x \in S$ one of the following statements holds: (i) $h(x) = f(x)$, or (ii) $h(x) = f(x) + (2n - |x_\ell|_1) \cdot \vec{1}$, or (iii) $h(x) = (\mathrm{LO}(x_\ell), \mathrm{TZ}(x_\ell) + 3n \cdot \vec{1}$. If (i) or (ii) holds, then $|S| \leq n$ since the range of f_1 is $1, \dots, n$. If (iii) holds we also obtain $|S| \leq n$ since there are at most $n/2 + 1$ distinct values for $\mathrm{LO}(x_\ell)$, i. e. for the first component of h. □

As explained at the beginning of this section, we now generalise $\mathrm{uRR}^*_{\mathrm{MO}}$ to the class $\mathrm{uRR}_{\mathrm{MO}}$ of functions.

Definition 19. Let Π_n be the set of all permutations of $[n]$, the class $\mathrm{uRR}_{\mathrm{MO}}$ of functions is defined as:

$$\mathrm{uRR}_{\mathrm{MO}} := \{\mathrm{uRR}^{\sigma,z}_{\mathrm{MO}}\}_{\sigma \in \Pi_n, z \in \{0,1\}^n}.$$

where each $\mathrm{uRR}^{\sigma,z}_{\mathrm{MO}}(x) := \mathrm{uRR}^*_{\mathrm{MO}}(\sigma(x) \oplus z)$.

Note that if e is the identity permutation, i. e. $\mathrm{e}(i) = i$ for all $i \in [n]$, then $\mathrm{uRR}^{\mathrm{e},0^n}_{\mathrm{MO}} \equiv \mathrm{uRR}^*_{\mathrm{MO}}$. As discussed in preliminaries, the behaviour of an algorithm that only employs unbiased variation operators, like standard bit mutation and uniform crossover, remains the same across every function of $\mathrm{uRR}_{\mathrm{MO}}$. Therefore in the proofs of the results for such an algorithm, the choice of which function does not matter to the expected running time and we will automatically assume the specific function $\mathrm{uRR}^*_{\mathrm{MO}}$.

4.1. Hardness of uRR_{MO} for EMOs without uniform crossover

We first show that without uniform crossover, both GSEMO and NSGA-II using standard bit mutation are inefficient in optimising uRR_{MO} functions. For this purpose, we define the following subset of the search space $\{0,1\}^n$ where the algorithms will typically start in and then likely remain there for long time if crossover is disabled:

$$K := \{x \in \{0,1\}^n \mid x_\ell \in U \vee x_r \in C\}.$$

Note that any point $x \in K$ is not Pareto-optimal, i. e. $K \cap W = \emptyset$, because $U \cap P = \emptyset$ and $C \cap T = \emptyset$, but any $y \in W$ requires that $y_\ell \in P$ and $y_r \in T$. Since K includes all search points falling into the first two cases in the definition of uRR^*_{MO}, the fitness of all search points that are neither Pareto optimal nor in K is $(0,0)$. Hence, every search point in K dominates every search point outside of K, except for Pareto optima.

Theorem 20. *The following algorithms requires $n^{\Omega(n)}$ fitness evaluations with probability $1 - 2^{-\Omega(n)}$ and in expectation to find any Pareto-optimal search point of any function of the class uRR_{MO},*

- *GSEMO (Algorithm 1) with $p_c = 0$ and standard bit mutation,*
- *NSGA-II (Algorithm 2) with $\mu \in \text{poly}(n)$, $p_c = 0$ and standard bit mutation.*

Proof. We first show the result for GSEMO. By a Chernoff bound and a union bound, the probability that the initial search point $x(0)$ has that $x(0)^j_\ell \notin [n/24, n/12]$ for one $j \in [4]$ is at most $2^{-\Omega(n)}$. Thus with probability $1 - 2^{-\Omega(n)}$, we have that $x(0) \in K$. Assume that this holds, then, as discussed above, the algorithm will only accept points from K or from W. This means that any point $y \in W$ (i. e. with $y_\ell \in P \wedge y_r \in T$), can only be created from a solution $x \in K$ by standard bit mutation. We consider the following cases. If $x_\ell \in U$, then by Lemma 16 we have $\text{H}(x_\ell, y_\ell) \geq n/8$, thus it is necessary that at least $n/8$ bits in x_ℓ have to be flipped to create any $y_\ell \in P$. This happens with probability at most $|P|(1/n)^{n/8} = (n/2+1)(1/n)^{n/8} = n^{-\Omega(n)}$ by a union bound. Otherwise if $x_r \in C$, then again by Lemma 16 we get $\text{H}(x_r, y_r) \geq 3n/16$. Thus in this case it is necessary to flip at least $3n/16$ bits and by a union bound the probability of creating $y_r \in T$ from x_r, is at most

$$|T|(1/n)^{3n/16} \leq 2^{n/2} n^{-3n/16}$$

since $T \subset \{0,1\}^{n/2}$ and $|\{0,1\}^{n/2}| = 2^{n/2}$. Thus, the probability of this happening in the first $n^{2n/16}$ generations is at most $n^{2n/16} \cdot 2^{n/2} n^{-3n/16} = 2^{n/2} n^{-n/16} = n^{-\Omega(n)}$. Taking a union bound over all failure probabilities implies the claim.

For NSGA-II, the probability of having any solution x of the initial population P_0 with $|P_0| = \mu$ and $x_\ell \notin U$ is at most $\mu \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$ by a Chernoff bound and a union bound, using $\mu \in \text{poly}(n)$. Assume that this does not happen, then in every generation of NSGA-II no solution created with fitness $(0,0)$ can survive to the next generation. Therefore, the algorithm will only accept search points from K or from W, and this means that the first $y \in W$ can only be created from a parent $x \in K$ by standard bit mutation. As in the case for GSEMO, the probability of such an event is at most $n^{-\Omega(n)}$. Continuing as above completes the proof. □

4.2. Both unary unbiased variation and hypermutation fail on uRR_{MO}

We now generalise the results of the previous section to unary unbiased variation and hypermutation operators, and also to elitist black-box algorithms. The following theorem shows that without uniform crossover, these algorithms and operators require an exponential expected number of fitness evaluations to optimise the function class uRR_{MO}, particularly in the worst case even to find any Pareto-optimal solution. The result on elitist black-box algorithms even holds for algorithms that are allowed to choose an arbitrary unbiased mutation operator or the hypermutation operator, and where this decision can be made anew in every step of the algorithm. In particular, this includes GSEMO with the hybrid mutation operator from Theorem 14.

Theorem 21. *The following algorithms require $2^{\Omega(n)}$ fitness evaluations with probability $1 - 2^{-\Omega(n)}$ and in expectation to optimise the function class uRR_{MO}:*

- *GSEMO (Algorithm 1) with $p_c = 0$,*
- *NSGA-II (Algorithm 2) with $\mu \in \text{poly}(n)$ and $p_c = 0$,*
- *Any $(\mu+\lambda)$ elitist black-box algorithm (Algorithm 8),*

that are allowed, in each step, to choose from any of the following mutation operators:

- *any unary unbiased variation operator (Algorithm 6),*
- *hypermutation (Algorithm 7) using any parameter $r \in (0,1]$.*

To prove the results for unary unbiased variation operators, it suffices to look at the basic function uRR^*_{MO} of the class, and we will use the same set K as defined in the previous section for this. The drawback of these operators is depicted in the following lemma

which shows that the probability of creating any Pareto-optimal search point from any $x \in K$ by such an operator is exponentially small.

Lemma 22. *Let* op *be any unary variation operator according to Algorithm 6, then for every $x \in K$ it holds that* $\Pr(\mathrm{op}(x) \in W) = 2^{-\Omega(n)}$.

Proof. For any substring x_{sub} of x, conditional on op flipping r bits in x and, among these, s bits in x_{sub}, the positions of the s bits to be flipped in x_{sub} are uniformly distributed in x_{sub}. Let $y := \mathrm{op}(x)$, we consider two cases:

Case $x_\ell \in U$: By Lemma 16 we get that $\mathrm{H}(x_\ell, y_\ell) \in [n/8, 3n/8]$, therefore even if we optimistically assume that the number of bits to be flipped in x_ℓ is $s = \mathrm{H}(x_\ell, y_\ell)$ (otherwise the sought probability is zero), still such a number of specific bits need to be flipped on x_ℓ to create $y_\ell \in P$. By a union bound and the well-known inequality $\binom{n}{k} \geq \left(\frac{n}{k}\right)^k$, the probability of creating any $y_\ell \in P$ by $\mathrm{op}(x)$ (a necessary condition for $y \in W$) in one offspring production is at most:

$$|P| \left(1 / \binom{n/2}{\mathrm{H}(x_\ell, y_\ell)}\right) \leq |P| \left(1 / \binom{n/2}{n/8}\right) \leq (n/2+1) \left(\frac{n/2}{n/8}\right)^{-n/8} = 4^{-\Omega(n)} \leq 2^{-\Omega(n)}.$$

Case $x_r \in C$: Let B be the event that $y \in W$ is created from $x \in K$, and $A_{s,q}$ be the event that s and q bits are flipped in x_ℓ and x_r, respectively, by $\mathrm{op}(x)$ for any fixed $s, q \in \{0, \dots, n/2\}$. By the law of total probability, it holds that

$$\Pr(B) = \sum_{s=0}^{n/2} \sum_{q=0}^{n/2} \Pr(B \mid A_{s,q}) \cdot \Pr(A_{s,q}). \tag{1}$$

We have $\Pr(A_{s,q}) \leq \binom{n/2}{s}\binom{n/2}{q} / \binom{n}{s+q}$ for all $0 \leq s+q \leq n$ because if for the chosen Hamming radius r we have $r \neq s+q$ then $\Pr(A_{s,q}) = 0$ and otherwise we have an equality due to the unbiasedness. Also due to this unbiasedness for any $s \in \{0, \dots, n/2\}$ the probability of generating one individual y with a specific y_ℓ-part if s bits on the left are flipped is $1/\binom{n/2}{s}$. Then by a union bound over the points of P, we obtain for any fixed $q, s \in \{0, \dots, n/2\}$ that $\Pr(B \mid A_{s,q}) \leq |P| / \binom{n/2}{s} = (n/2+1)/\binom{n/2}{s}$. Furthermore, by Lemma 16 we have that $\Pr(B \mid A_{s,q}) = 0$ for $q \notin [3n/16, 5n/16]$. Thus the inner sum of (1) for any s is at most

$$\sum_{q=0}^{n/2} \Pr(B \mid A_{s,q}) \cdot \Pr(A_{s,q}) \leq \sum_{q=3n/16}^{5n/16} \frac{\binom{n/2}{s} \cdot \binom{n/2}{q}}{\binom{n}{s+q}} \cdot \frac{n/2+1}{\binom{n/2}{s}} = (n/2+1) \sum_{q=3n/16}^{5n/16} \frac{\binom{n/2}{q}}{\binom{n}{s+q}}.$$

We define $M(s,q) := \binom{n/2}{q} / \binom{n}{s+q}$ and claim that $M(s,q) = 2^{-\Omega(n)}$ for all integers $s \in [0, n/2]$, $q \in [3n/16, 5n/16]$. This implies the statement because then $\Pr(B) \leq (n/2+1) \sum_{q=3n/16}^{5n/16} M(s,q) \leq n^2 \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$. We consider two subcases.

If $0 \leq s \leq 3n/8$: We have $q \leq s+q \leq 3n/8 + 5n/16 = 11n/16$ and also $n - q \geq n - 5n/16 = 11n/16$, thus $q \leq s+q \leq n-q$. This means that $\binom{n}{s+q}$ is not farther away from the central binomial coefficient $\binom{n}{n/2}$ than $\binom{n}{q} = \binom{n}{n-q}$, thus the former coefficient is no less than the latter, i. e. $\binom{n}{s+q} \geq \binom{n}{q}$. Hence, we get

$$\begin{aligned} M(s,q) &\leq \binom{n/2}{q} / \binom{n}{q} = \frac{(n/2)!}{n!} \cdot \frac{(n-q)!}{(n/2-q)!} = \prod_{j=1}^{n/2} \frac{n/2-q+j}{n/2+j} \overset{q \geq 3n/16}{\leq} \prod_{j=1}^{n/2} \frac{n/2-3n/16+j}{n/2+j} \\ &= \prod_{j=1}^{n/2} \frac{5n/16+j}{n/2+j} = \prod_{j=1}^{n/2} \left(1 - \frac{3n/16}{n/2+j}\right) \leq \prod_{j=1}^{n/2} \left(1 - \frac{3n/16}{n/2+n/2}\right) = \left(1 - \frac{3}{16}\right)^{n/2} = 2^{-\Omega(n)}. \end{aligned} \tag{2}$$

If $3n/8 + 1 \leq s \leq n/2$: This implies $s + q > 3n/8 + 3n/16 = 9n/16 > n/2$ and because $\binom{n}{k}$ is a decreasing function of k when $k \geq n/2$ and given $s \leq n/2$, it holds that $\binom{n}{s+q} \geq \binom{n}{n/2+q}$. Hence, we have

$$M(s,q) \leq \frac{\binom{n/2}{q}}{\binom{n}{n/2+q}} = \frac{(n/2)!}{n!} \cdot \frac{(n/2+q)!}{q!} = \prod_{j=1}^{n/2} \frac{q+j}{n/2+j} \overset{q \leq 5n/16}{\leq} \prod_{j=1}^{n/2} \frac{5n/16+j}{n/2+j} \overset{(2)}{=} 2^{-\Omega(n)}. \quad \square$$

Regarding hypermutations, we remark their following ability to imitate uniform crossover on an ideal function (e. g. $\mathrm{uRR}^*_{\mathrm{MO}}$), but also their drawbacks on other functions of the class. Uniform crossover keeps the bit values of bit positions where both parents agree. On all disagreeing bits, uniform crossover generates uniform random bit values. Hypermutation can simulate the latter effect of uniform crossover within the substring given by parameters c and $c + \ell - 1$ if $r = 0.5$ is chosen. Then all bits in this substring are set to uniform random values. Jansen and Zarges [38] showed that this can speed up the optimisation process. However, hypermutation subjects *all* bits in the substring to mutation, whereas uniform crossover only "mutates" bits in which both parents disagree. In particular, uniform crossover treats bits where both parents agree differently from those where both parents disagree. Our royal road function requires bits in x_ℓ to be kept and only bits in x_r to be mutated with a high mutation rate. Here the ability to treat these subsets of bits differently is crucial. Hypermutation does not have this ability, and we show that there is a function in the class $\mathrm{uRR}_{\mathrm{MO}}$ for which hypermutation fails badly.

We consider the function $\text{uRR}^{\sigma,0^n}_{\text{MO}} = \text{uRR}^{*}_{\text{MO}}(\sigma(x))$ with the permutation σ satisfying:

$$\forall x \in \{0,1\}^n : \sigma(x) = \sigma(\underbrace{(x^1_\ell, x^2_\ell, x^3_\ell, x^4_\ell}_{x_\ell}, \underbrace{x^1_r, x^2_r, x^3_r, x^4_r}_{x_r})) = (\underbrace{x^1_\ell, x^3_\ell, x^1_r, x^3_r}_{\sigma(x)_\ell}, \underbrace{x^2_\ell, x^4_\ell, x^2_r, x^4_r}_{\sigma(x)_r}). \tag{3}$$

Hereinafter, we use the notation in Definition 15 to write the parts of a bit string x and also apply this notation to the permuted string $\sigma(x)$ of x, e. g. $\sigma(x)_\ell = (x^1_\ell, x^3_\ell, x^1_r, x^3_r)$. For the sake of clarity, we also introduce $U_\sigma, P_\sigma, C_\sigma, T_\sigma$ as the analogous sets to those defined in Definition 15, however these sets are defined on the permuted string $\sigma(x)$ instead of on x, e. g. $U_\sigma := \left\{ \sigma(x)_\ell \in \{0,1\}^{n/2} \mid \forall i \in [4] : |\sigma(x)^i_\ell|_1 \in [n/24, n/12] \right\}$. The set of non Pareto-optimal solutions with non-null fitness vectors, and that of Pareto-optimal solutions for $\text{uRR}^{\sigma,0^n}_{\text{MO}}$ are then:

$$K_\sigma := \{x \in \{0,1\}^n \mid \sigma(x)_r \in U_\sigma \vee \sigma(x)_r \in C_\sigma\},$$
$$W_\sigma := \{x \in \{0,1\}^n \mid \sigma(x)_\ell \in P_\sigma \wedge \sigma(x)_r \in T_\sigma\}.$$

Note that the function to optimise is $\text{uRR}^{*}_{\text{MO}}(\sigma(x))$, however hypermutations can only manipulate the original string x. Given the intertwined locations of the blocks as shown in (3), the probability of creating any Pareto-optimal solution for this function from any $x \in K_\sigma$ is then exponentially small.

Lemma 23. *Let* op *be any hypermutation operator according to Algorithm 7 with any parameter* $r \in (0,1]$*, then for any* $x \in K_\sigma$ *it holds that* $\Pr\left(\text{op}(x) \in W_\sigma\right) = 2^{-\Omega(n)}$.

Proof. We distinguish two cases for x.

If $\sigma(x)_\ell \in U_\sigma$ then we argue as in the proof of Lemma 16(i): creating a mutant $y \in W_\sigma$ requires $\sigma(y)_\ell \in P_\sigma$ which means $\sigma(y)_\ell = 1^a 0^{n/2-a}$ for some non-negative integer a, and this implies $\sigma(y)^j_\ell \in \{0^{n/8}, 1^{n/8}\}$ for at least three distinct $j \in [4]$. We call these *bad blocks*. By definition of U_σ, $|\sigma(x)^j_\ell|_1 \in [n/24, n/12]$ for all blocks j, thus all bad blocks must be altered by the hypermutation as a necessary condition for creating y. Recall that hypermutation operates on the original string x. Therefore, the values of c and ℓ in Algorithm 7 have to be chosen so that the positions between c and $c + \ell - 1 \bmod n$ contain bits from all bad blocks. Since σ permutes blocks as a whole, every interval containing bits from three bad blocks must cover at least one bad block entirely. This block has to be mutated into $0^{n/8}$ or $1^{n/8}$ in one application of the hypermutation. By a union bound this happens with probability at most $2(r \cdot (1-r))^{n/24} \leq 2 \cdot (1/4)^{n/24} = 2^{-\Omega(n)}$ since we have to flip at least $n/24$ bits while keeping at least $n/24$ bits unchanged.

If $\sigma(x)_r \in C_\sigma$ then $\sigma(x)_r \in \{1^a 0^{n/2-a}, 0^a 1^{n/2-a} \mid 0 \leq a \leq n/2\}$, thus $\sigma(x)^j_r \in \{0^{n/8}, 1^{n/8}\}$ for at least three[2] distinct $j \in [4]$, and we again refer to these bits as *bad blocks*. So, in order to have $\sigma(y)_r \in T_\sigma$, we have to flip half of the bits in each of these three blocks and this implies that the interval of bits in x subjected to hypermutation has to be chosen such that all bad blocks in $\sigma(x)_r = (x^2_\ell, x^4_\ell, x^2_r, x^4_r)$ are touched. Since the interval is chosen in x and $x = (x^1_\ell, x^2_\ell, x^3_\ell, x^4_\ell, x^1_r, x^2_r, x^3_r, x^4_r)$, every interval containing bits from three bad blocks also contains a whole block from $\sigma(x)_\ell = (x^1_\ell, x^3_\ell, x^1_r, x^3_r)$. This can be easily verified by considering cases where blocks x^2_ℓ and x^2_r are bad and otherwise considering bad blocks x^4_ℓ and x^4_r. Thus, every interval contains a whole bad block in $\sigma(x)_r$ and also one whole block of $\sigma(x)_\ell = (x^1_\ell, x^3_\ell, x^1_r, x^3_r)$. We bound the probability of creating optimal configurations in these two blocks. Flipping half of the bits in one block of $\sigma(x)_r$ happens with probability $\binom{n/8}{n/16} \cdot r^{n/16} \cdot (1-r)^{n/16}$. For the block in $\sigma(y)_\ell$ we note that there are at most $(n/8)+1$ optimal configurations $1^b 0^{n/8-b}$ for $b \in \{0, \dots, n/8\}$. We fix one optimal configuration and argue that all bits in $\sigma(y)_\ell$ differing from in must be flipped, and all other bits in that block must not be flipped. If m bits differ, the probability for this event is $r^m \cdot (1-r)^{n/8-m} \leq \max(r, 1-r)^m \cdot \max(r, 1-r)^{n/8-m} = \max(r, 1-r)^{n/8}$ where the last expression no longer depends on m. By a union bound over all $(n/8)+1$ optimal configurations, the probability of creating any $y \in W_\sigma$ from $x \in K_\sigma$ is at most, using $\binom{n/8}{n/16} \leq 4^{n/16}$,

$$\binom{n/8}{n/16} (r(1-r))^{\frac{n}{16}} \cdot \left(\frac{n}{8}+1\right) \max(r, 1-r)^{\frac{n}{8}} \leq \left(\frac{n}{8}+1\right) \left(4 \max(r^2(1-r), r(1-r)^2)\right)^{\frac{n}{16}} =: p.$$

Note that $4\max(r^2(1-r), r(1-r)^2)$ on $r \in (0,1]$ reaches the maximum value $16/27$ at either $r = 1/3$ or $r = 2/3$, therefore $p \leq (n/8+1)(16/27)^{n/16} = 2^{-\Omega(n)}$. □

We can now prove Theorem 21.

Proof of Theorem 21. Due to the unbiasedness of the variation operators, the result of Lemma 22 not only holds for $\text{uRR}^{*}_{\text{MO}}$ but also holds for $\text{uRR}^{\sigma,0^n}_{\text{MO}}$ with σ defined as in Equation (3) (but with the sets K_σ, W_σ in the statement of the lemma) and in fact for

[2] Note that this can be exactly three blocks if for one $i \in [4]$ we have $\sigma(x)^i_r \in \{1^{n/16}0^{n/16}, 0^{n/16}1^{n/16}\}$, because then $|\sigma(x)^i_r|_1 = |\sigma(x)^i_r|_0 = n/16$ and nothing needs to be changed for this block i.

any function of the class uRR^*_{MO} (with its appropriate sets K and W). Thus, for simplification we will use the same function $\text{uRR}^{\sigma,0^n}_{\text{MO}}$ throughout the proof.

Regarding the algorithms, note that the result of NSGA-II follows immediately from that of the elitist black-box algorithm because NSGA-II with $p_c = 0$ is a special case of an elitist algorithm with $\lambda = \mu$. Hence, it remains to show the results for GSEMO and the elitist black-box algorithm.

Starting with GSEMO, the probability that the initial search point belongs to K_σ is at least $1 - 2^{-\Omega(n)}$ by a Chernoff bound. Assume this does happen, then the first $y \in W_\sigma$ can be only created from a point $x \in K_\sigma$. By Lemmas 22 and 23 this happens with probability is both at most $2^{-\Omega(n)}$ by a unary variation operator or a hypermutation. Thus the expected number of fitness evaluations is at least $(1 - 2^{-\Omega(n)})2^{\Omega(n)} = 2^{\Omega(n)}$.

Now consider an elitist black-box algorithm. Initialising every search point in K_σ happens with probability $1 - \mu \cdot 2^{-\Omega(n)} = 1 - 2^{-\Omega(n)}$ by a Chernoff and a union bound, using $\mu \in \text{poly}(n)$. If this happens then due to the elitism the algorithm accepts only search points from K_σ or W_σ. Again by Lemmas 22 and 23 the probability for creating a point in W_σ from a point in K_σ with an unbiased variation operator or hypermutation is $2^{-\Omega(n)}$. The probability of this happening in 2^{cn} evaluations, $c > 0$ a small enough constant, is $2^{-\Omega(n)}$. □

Unlike the previous result for GSEMO on RR_{MO} (Theorem 12) where only one hypermutation operator with a fixed parameter r is allowed, such a restriction is not required here for the class uRR_{MO}. In particular, Theorem 12 includes algorithms that may choose from all unary unbiased variation operators and hypermutation in every generation, and the hypermutation parameter r can be chosen anew in every application of the operator.

4.3. Use of uniform crossover implies expected polynomial optimisation time on uRR^*_{MO}

We show for GSEMO and for NSGA-II that they can find the whole Pareto front for uRR^*_{MO} in polynomial expected time when using both standard bit mutation and uniform crossover.

4.3.1. Analysis of GSEMO

Theorem 24. *GSEMO (Algorithm 1) with $p_c \in (0,1)$, uniform crossover and standard bit mutation finds a Pareto-optimal set of any function in the class uRR_{MO} in $\mathcal{O}\left(\frac{n^3}{p_c(1-p_c)}\right)$ fitness evaluations in expectation.*

Proof. As in the proof of Theorem 10 for RR_{MO} we use the method of typical runs and divide the optimisation procedure into several phases.

Phase 1: Create a search point x with $\text{uRR}^*_{\text{MO}}(x) \neq (0,0)$.

By a Chernoff bound and a union bound, the probability of initializing an individual x with fitness $(0,0)$ is at most $2^{-\Omega(n)}$ as it is necessary to create a search point x with $|x^i_\ell|_1 \notin [n/24, n/12]$ for some $i \in [4]$ and the expected number of ones in x^1_ℓ or x^2_ℓ is $n/16$ after initialization. If this occurs then as long as every individual created has fitness $(0,0)$, the population always consists of the latest search point and crossover, if executed, has no effect. To bound the expected number of generations for all $|x^i_\ell|_1$ to simultaneously reach the interval $[n/24, n/12]$, we divide the run of the phase into subphases. Each subphase lasts for $\tau := \max_{i\in[4]}\{\tau_i\}$ generations where τ_i for each $i \in [4]$ is the first point during the subphase that $|x^i_\ell|_1$ reaches interval $[3n/56, n/14]$, which is a subinterval of our target $[n/24, n/12]$. A subphase is called *successful* if during its length, none of the $|x^i_\ell|_1$ which have entered the subinterval leave our target. If a subphase is unsuccessful, a new one is assumed to start and we repeat our analysis, thus the expected number of generations of the phase is bounded from above by the expected waiting time for a successful subphase.

Recall that each block x^i_ℓ has length $n/8$, thus it follows from Lemma 5 with $c = 1/8$ and $\varepsilon = (1/14 - 1/16)(2/c) = 1/7$ that $\text{E}[\tau_i] = \mathcal{O}(n)$ for all $i \in [4]$. Because $\tau \leq \sum_{i=1}^4 \tau_i$, then by linearity of expectation we get the expected length of a subphase is $\text{E}[\tau] \leq \text{E}\left[\sum_{i=1}^4 \tau_i\right] = \sum_{i=1}^4 \text{E}[\tau_i] = \mathcal{O}(n)$. Furthermore, for any $i \in [4]$ if we define $Y_t := \max\{|x^i_\ell|_1, |x^i_\ell|_0\}$, then as shown in the proof Lemma 5 it holds that $\text{E}[Y_{t+1} - Y_t \mid Y_t = s] \leq -c\varepsilon + o(1) = -1/56 + o(1)$ for all $s \in [n/14, n/12]$. Additionally, in order to have $|Y_t - Y_{t+1}| \geq m$ at least m bits in x^i_ℓ have to be flipped in order to have either $Y_t - Y_{t+1} \geq m$ or $Y_{t+1} - Y_t \geq m$, hence by a union bound

$$\Pr\left(|Y_t - Y_{t+1}| \geq m\right) \leq 2\binom{n/8}{m} n^{-m} = 2 \cdot \frac{(n/8)!}{(n/8-m)!m!} \cdot n^{-m} \leq 2(n/8)^m n^{-m} \leq \frac{1}{4^m}.$$

Theorem 2 applied to the process $Y'_t = n/8 - Y_t$ with $a = (n/8 - n/12)/n = 1/24$, $b = (n/8 - n/14)/n = 3/56$, $r = 1$ and $\eta = 3$ implies that even within an exponential number of generations, the probability that Y_t goes above $n/12$ again, or in other words $|x^i_\ell|_1$ leaves the target interval $[n/24, n/12]$, is at most $2^{-\Omega(n)}$. Since this holds for any block x^i_ℓ, the probability of an unsuccessful subphase is still at most $4! \cdot 3 \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$ even by taking into account any possible order for the blocks to reach the subinterval and a union bound on the failure probabilities of the three first blocks. By the method of typical runs, the expected length of the first phase is then $1 + 2^{-\Omega(n)}\left(\text{E}[\tau]/(1 - 2^{-\Omega(n)})\right) = 1 + 2^{-\Omega(n)}\left(\mathcal{O}(n)/(1 - 2^{-\Omega(n)})\right) = 1 + o(1)$.

Phase 2: Create a search point x with $x_r \in C$.

Suppose that there is no such search point in P_t. Let

$$k_1 := \max\{\mathrm{LO}(x_r) + \mathrm{TZ}(x_r) \mid x \in P_t \text{ with } \mathrm{LO}(x_r) \neq 0\}$$

and

$$k_2 := \max\{\mathrm{LZ}(x_r) + \mathrm{TO}(x_r) \mid x \in P_t \text{ with } \mathrm{LO}(x_r) = 0\}.$$

Let $m := \max\{k_1, k_2\} \in \{1, \dots, n/2 - 1\}$. Since a search point y with $\mathrm{LO}(y_r) + \mathrm{TZ}(y_r) = k_1$ (or $\mathrm{LZ}(y_r) + \mathrm{TO}(y_r) = k_2$) can be only dominated by a search point z with $\mathrm{LO}(z_r) + \mathrm{TZ}(z_r) \geq k_1$ (or $\mathrm{LZ}(z_r) + \mathrm{TO}(z_r) \geq k_2$), we see that m cannot decrease. To increase the value m it suffices to choose an individual y with $y_\ell \neq 0^{n/2}$, $\mathrm{LO}(y_r) + \mathrm{TZ}(y_r) = k_1$ if $m = k_1$ or $\mathrm{LZ}(y_r) + \mathrm{TO}(y_r) = k_2$ if $m = k_2$, omit crossover and flip one specific bit of y in the mutation step. Since the parent is chosen uniformly at random, i. e. with probability at least $1/n$ by Lemma 18, the probability for this event is at least $(1 - p_c) \cdot 1/n^2 \cdot (1 - 1/n)^{n-1} \geq \frac{1-p_c}{en^2}$. Since there are at most $n/2$ different values for m we need at most $n/2 - 1$ such improving steps. So the expected number of fitness evaluations to complete this phase is at most $\frac{1}{1-p_c}(n/2 - 1)en^2 = \mathcal{O}\left(\frac{n^3}{1-p_c}\right)$.

Phase 3: Create a search point x with $x_r \in C$ and $x_\ell = 0^{n/2}$.

We pessimistically assume that there is no such search point in the current population. To make progress towards the goal to this phase, it suffices to first select an individual x which has a minimum value i of ones in the first half of the string amongst those individuals with $x_r \in C$, then to omit crossover and to flip one of the i 1-bits in the first half during mutation. The probability of this event is at least $\frac{1}{n} \cdot (1 - p_c) \cdot \binom{i}{1} \cdot \frac{1}{n} \cdot (1 - 1/n)^{n-1} = \frac{1}{n} \cdot (1 - p_c) \cdot \frac{i}{n} \cdot (1 - 1/n)^{n-1} \geq \frac{i(1-p_c)}{en^2}$. Since $n/2$ such steps are enough we have that the goal of this phase is reached after at most

$$\sum_{i=1}^{n/2} \frac{en^2}{i(1 - p_c)} = \mathcal{O}\left(\frac{n^2 \log(n)}{1 - p_c}\right)$$

fitness evaluations in expectation.

Phase 4: The population contains every search point x with $x_r \in C$ and $x_\ell = 0^{n/2}$.

Suppose that this condition is not fulfilled. Then there must exist a search point $z \notin P_t$ with $z_\ell = 0^{n/2}$ and $z_r \in C$ as well as a search point $y \in P_t$ with $y_\ell = 0^{n/2}$, $y_r \in C$ and $H(y, z) = 1$. Then the individual z can be generated by choosing y as a parent, omitting crossover and flipping a specific bit while keeping the other bits unchanged. The probability of this event is at least $(1 - p_c) \cdot 1/n^2 \cdot (1 - 1/n)^{n-1} \geq (1 - p_c) \cdot 1/n^2 \cdot 1/e$. Note that $n - 1$ such steps suffice to finish this phase since there at most n different search points x with $x_\ell = 0^{n/2}$ and $x_r \in C$. Thus the expected number of fitness evaluations is at most

$$\frac{en^2(n - 1)}{1 - p_c} = \mathcal{O}\left(\frac{n^3}{1 - p_c}\right).$$

Phase 5: Create one point in W.

Suppose that there is no such point in the current population. Note that the population is the set of all search points x with $x_\ell = 0^{n/2}$ and $x_r \in C$. Since C is a circle of length n, the population size is n during this phase. Then $x \in P_t$ if and only if $x = (x_\ell, x_r)$ with $x_\ell = 0^{n/2}$ and $x_r \in C$. To generate a point in W one can choose two individuals x, y with maximal Hamming distance (which happens with probability $1/n$), apply uniform crossover to generate an individual $z = (z_\ell, z_r)$ with $z_\ell = 0^{n/2}$ and $z_r \in T$ and not flip any bit during the mutation step. For the outcome $z_r = (z_r^1, z_r^2, z_r^3, z_r^4)$ of uniform crossover it is sufficient that z_r^i contains $n/16$ zeros and $n/16$ ones. The probability of this event is $\binom{n/8}{n/16}(\frac{1}{2})^{n/8} = \Theta(n^{-1/2})$ by Stirling's formula for a given $i \in [4]$. Since these events are independent for different i, the probability of generating a suitable z is at least $\Theta(n^{-2})$. Thus, the probability of generating a search point in W is at least $1/n \cdot p_c \cdot \Theta(n^{-2}) \cdot (1 - 1/n)^n = \Theta(p_c n^{-3})$ (since $(1 - 1/n)^n \geq 1/4$). So the expected number of fitness evaluations to complete this phase is at most $\Theta\left(\frac{1}{p_c} \cdot n^3\right)$.

Phase 6: Find a Pareto-optimal set.

Once a search point x with $\mathrm{LO}(x_\ell) + \mathrm{TZ}(x_\ell) = n/2$ and $x_r \in T$ is created, the process of covering the front is similar to that of creating every search point y with $y_r \in C$ and $y_\ell = 0^{n/2}$ in Phase 4, with only minor differences (e. g. we have to consider only $n/2$ optimization steps instead of $n - 1$ many where we flip a specific bit in the first half of the string). The expected number of fitness evaluations to finish this phase is at most $\mathcal{O}\left(\frac{n^3}{1-p_c}\right)$.

Summing up the expected times of all the phases gives an upper bound $\mathcal{O}\left(\frac{1}{p_c} \cdot n^3 + \frac{1}{1-p_c} \cdot n^3\right)$ on the number of generations to optimize $\mathrm{uRR}^*_{\mathrm{MO}}$. Noting $\frac{1}{p_c} + \frac{1}{1-p_c} = \frac{1-p_c}{p_c(1-p_c)} + \frac{p_c}{p_c(1-p_c)} = \frac{1}{p_c(1-p_c)}$ completes the proof. □

4.3.2. Analysis of NSGA-II

We now turn to the analysis of NSGA-II. Here we deal with a population size of $\mu = 5n$ since we have at most n non-dominated solutions.

Theorem 25. *NSGA-II (Algorithm 2) with $p_c \in (0, 1)$ and $\mu \geq 5n$ finds a Pareto-optimal set of any function of the class $\mathrm{uRR}_{\mathrm{MO}}$ in $\mathcal{O}\left(\frac{\mu n}{p_c} + \frac{n^2}{1-p_c}\right)$ generations, or $\mathcal{O}\left(\frac{\mu^2 n}{p_c} + \frac{\mu n^2}{1-p_c}\right)$ fitness evaluations, in expectation.*

Proof. As in the proof of Theorem 11 we use the method of typical runs and divide the optimisation procedure into several phases. Since by Lemma 18 there are at most n non-dominated search points with respect to uRR^*_{MO} we see with Lemma 4 that there can be at most $4n \leq (4/5)\mu$ individuals in F_t^1 with positive crowding distance. Therefore at least $\mu/5$ individuals are either in F_t^1 and have zero crowding distance or are in a layer below F_t^1.

Phase 1: Create a search point x with $\text{uRR}^*_{\text{MO}}(x) \neq 0$.

By a Chernoff bound the probability that every initial individual x fulfills $|x_\ell^i|_1 \notin [n/12, n/24]$ for $i \in [4]$ is at most $2^{-\mu\Omega(n)}$. If this happens, the probability of creating a specific individual by mutation is at least n^{-n}, regardless of the input solution and the preceding crossover. By the law of total probability, the expected number of evaluations to obtain a search point with fitness $f(x)$ is at most $\mu + 2^{-\mu\Omega(n)} n^n = \mu + 2^{-\Omega(n^2)} \cdot n^n = \mu + o(1)$, and these are $1 + o(1)$ generations.

Phase 2: Create a search point x with $x_r \in C$.

Suppose that there is no such search point x in the population. Let k_1, k_2 and m be as in Phase 3 in the proof for GSEMO. Note that m cannot decrease, since there are at most n non-dominated solutions and a search point with $\text{LO}(y_r) + \text{TZ}(y_r) = k_1$ (or $\text{LZ}(y_r) + \text{TO}(y_r) \geq k_2$) can only be dominated by a search point z with $\text{LO}(z_r) + \text{TZ}(z_r) \geq k_1$ (or $\text{LZ}(z_r) + \text{TO}(z_r) \geq k_2$). Suppose that $m = k_1$. As in the proof for GSEMO, the case "$m = k_2$" is handled similarly. Let y be an individual with positive crowding distance, $\text{LO}(y_r) + \text{TZ}(y_r) = k_1$ if $m = k_1$ or $\text{LZ}(y_r) + \text{TO}(y_r) = k_2$ if $m = k_2$. If y appears as the first competitor in a binary tournament (which happens with probability $1/\mu$), and the second competitor is from F_t^1 and has zero crowding distance or is a layer below F_t^1 (which happens with probability at least $1 - \frac{4n}{5n} = 1/5$ as remarked at the beginning of the proof of this theorem), then y wins the tournament. The same holds for swapping the roles of the first and second competitor. Therefore we have with probability at least $\frac{2}{5\mu}$ that y is the outcome of one tournament (since the multiset of individuals with positive crowding distance is disjoint to the multiset of individuals which are either below F_t^1 or from F_t^1 with zero crowding distance). Since there are two tournaments in generating a pair of offspring we obtain that the probability for generating y as an outcome of one of the two tournaments is at least $1 - (1 - \frac{2}{5\mu})^2 \geq \frac{4/(5\mu)}{4/(5\mu)+1} = \frac{4}{4+5\mu}$ by Lemma 3. Thus to increase k_1 it suffices to generate y as such an outcome, omit crossover and flip a specific bit to one while keeping the other bits unchanged. Therefore, with probability at least $\frac{4}{4+5\mu} \cdot (1 - p_c) \cdot \frac{1}{n} \cdot (1 - 1/n)^{n-1} := b$, one of the offspring s from $\{s'_1, s'_2\}$ has a larger $(\text{LO}(s_r) + \text{TZ}(s_r))$-value than m if $m = k_1$ (larger $(\text{LZ}(s_r) + \text{TO}(s_r))$-value than m if $m = k_2$). This reproduction process is repeated $\mu/2$ times, so the chance of increasing m in one generation is at least $1 - (1 - b)^{\mu/2} \geq \frac{b\mu/2}{b\mu/2+1}$ (also by Lemma 3). Since we have to increase m at most $n/2 - 1$ times, the expected number of generations is no more than

$$\left(\frac{n}{2} - 1\right)\left(1 + \frac{1}{b\mu/2}\right) = \frac{n}{2} - 1 + \frac{2n(4+5\mu)}{4\mu \cdot (1-p_c) \cdot (1-1/n)^{n-1}} \cdot \left(\frac{n}{2} - 1\right) = \mathcal{O}\left(\frac{n^2}{1-p_c}\right).$$

Phase 3: Create a search point x with $x_r \in C$ and $x_\ell = 0^{n/2}$.

We pessimistically assume that there is no individual with these properties in the current population. We say that individual $x \in P_t$ has property $\mathcal{Q}_i$ for $i \in \{0, \ldots, m\}$ if $x_r \in C$ and $|x_\ell|_1 = i$. Let $i > 0$ be minimal such that there is an individual $x \in P_t$ with property $\mathcal{Q}_i$. Then $x \in F_t^1$ and thus at least one of the outcomes of the two binary tournaments has property $\mathcal{Q}_i$ if x is picked as a competitor and the second competitor is in a layer below F_t^1 or has zero crowding distance. Similar as in Phase 2, this happens with probability at least $\frac{4}{4+5\mu}$. To generate an individual with property $\mathcal{Q}_j$ for $j < i$ from such an outcome it suffices to omit crossover and flip a 1-bit in the first half of the string to zero while keeping the remaining bits unchanged. Therefore the probability is at least $r_i := \frac{4}{4+5\mu} \cdot (1 - p_c) \cdot \frac{i}{n} \cdot (1 - 1/n)^{n-1}$ that one of the offspring $\{s'_1, s'_2\}$ has property $\mathcal{Q}_j$ for $j < i$. Since the reproduction process is repeated $\mu/2$ times we get no more than

$$\sum_{i=1}^{n/2-1} \left(1 + \frac{1}{r_i \mu/2}\right) = \sum_{i=1}^{n/2-1} \left(1 + \frac{2n(4+5\mu)}{4 \cdot i \cdot \mu \cdot (1-p_c) \cdot (1-1/n)^{n-1}}\right) = \mathcal{O}\left(\frac{n \log n}{1-p_c}\right)$$

for the expected number of generations.

Phase 4: Create every search point x with $x_r \in C$ and $x_\ell = 0^{n/2}$.

Suppose that C is not completely covered by search points x with $x_r \in C$ and $x_\ell = 0^{n/2}$. Let y and z be defined as in Phase 4 in the proof of Theorem 1. Additionally assume that y has a positive crowding distance in F_t^1. As in the previous phases the probability that y is the outcome of a binary tournament is at least $\frac{4/(5\mu)}{4/(5\mu)+1} = \frac{4}{4+5\mu}$. Thus the probability of generating z is at least $a := \frac{4}{4+5\mu} \cdot (1 - p_c) \cdot 1/n \cdot (1 - 1/n)^{n-1} \geq \frac{4}{4+5\mu} \cdot (1 - p_c) \cdot 1/n \cdot 1/e$ during the creation of the pair of offspring. The success probability for $\mu/2$ pairs is then at least $1 - (1 - a)^{\mu/2} \geq \frac{a\mu/2}{a\mu/2+1}$ by Lemma 3. Therefore the expected number of generations to complete this phase is no more than $(n - 1) \cdot (1 + \frac{1}{a\mu/2})$ which is

$$n - 1 + \frac{2en(n-1)(4+5\mu)}{4\mu(1-p_c)} = \mathcal{O}\left(\frac{n^2}{1-p_c}\right).$$

Phase 5: Create a search point $x \in W$.

Note that P_t contains all search points x with $x_\ell = 0^{n/2}$ and $x_r \in C$ and they are in F_t^1. Note that F_t^1 consists only of search points of this form. There are at least n solutions x with $x_\ell = 0^{n/2}$, $x_r \in C$ and positive crowding distance in P_t, thus one of them wins the

tournament for selecting p_1 with probability at least $\frac{2n}{5\mu}$. Then it suffices to select a specific solution in P_t with positive crowding distance as p_2, which happens with probability at least $\frac{2}{5\mu}$. (This could be a solution y with $y_\ell = 0^{n/2}$, $y_r \in C$, maximal Hamming distance to p_1 and positive crowding distance). Then apply uniform crossover on p_1 and p_2 to generate an individual $z = (z_\ell, z_r)$ with $z_\ell = 0^{n/2}$ (i. e. $\mathrm{LO}(z_\ell) + \mathrm{TZ}(z_\ell) = n/2$) and $z_r \in T$ (which happens with probability at least $p_c \cdot \left(\binom{n/8}{n/16}(1/2)^{n/8}\right)^4$ as in the proof in Phase 5 for GSEMO) and do not flip any bit during the mutation step. Therefore we have a probability of at least

$$w := p_c \cdot \frac{4}{25} \cdot \frac{n}{\mu^2} \cdot \left(\binom{n/8}{n/16}(1/2)^{n/8}\right)^4 \cdot (1-1/n)^n = \Omega\left(\frac{p_c}{\mu^2 n}\right)$$

that an offspring x with $x_r \in T$ and $\mathrm{LO}(x_\ell) + \mathrm{TZ}(x_\ell) = n/2$ (i. e. $x \in W$) is generated during the creation of a pair of offspring. By Lemma 3 at least one success occurs in a generation with probability at least $1-(1-w)^{\mu/2} \geq \frac{w\mu/2}{w\mu/2+1}$. Thus $1 + \frac{2}{\mu w} = \mathcal{O}\left(\frac{\mu n}{p_c}\right)$ generations are sufficient in expectation to finish this phase.

Phase 6: Find a Pareto-optimal set.

Once a search point x with $\mathrm{LO}(x_\ell) + \mathrm{TZ}(x_\ell) = n/2$ and $x_r \in T$ is created, the process of covering the front is similar to that of creating every search point y with $y_r \in C$ and $y_\ell = 0^{n/2}$ in Phase 4 with only minor differences (e. g. we have to consider only $n/2$ optimization steps instead of $n-1$ many where we flip a specific bit in the first half of the string while keeping the rest of the bits unchanged). The expected number of generations to finish this phase is at most $\mathcal{O}\left(\frac{n^2}{1-p_c}\right)$.

We see that the expected number of generations for finding a global optimum in total is $\mathcal{O}(\frac{\mu n}{p_c} + \frac{n^2}{1-p_c})$ by adding the run times of the single phases. We obtain the expected number of fitness evaluations by multiplying the expected number of generations with 2μ. □

5. Discussion and conclusions

We have identified the function classes $\mathrm{RR_{MO}}$ and $\mathrm{uRR_{MO}}$ as examples on which EMO algorithms GSEMO and NSGA-II using crossover and standard bit mutation can find a whole Pareto set in expected polynomial time, i. e. $\mathcal{O}(n^4)$ for $\mathrm{RR_{MO}}$ and $\mathcal{O}(n^3)$ for $\mathrm{uRR_{MO}}$, with any constant crossover probability $p_c \in (0,1)$. For NSGA-II, these results hold for a linear population size, specifically we require $\mu \geq 2n+5$ for $\mathrm{RR_{MO}}$ with one-point crossover and $\mu \geq 5n$ for $\mathrm{uRR_{MO}}$ with uniform crossover. More generally, the function classes can be optimised in expected polynomial time if $1/p_c$, $1/(1-p_c)$ and μ are polynomials in n. Note that the requirement on $1/(1-p_c)$ is only an artefact of our analysis as the term reflects the expected waiting time for a step executing only a standard bit mutation. In fact, one can easily show that our upper bounds which are polynomial for $p_c \in (1-\Omega(1)) \cap \Omega(1)$ remain polynomial for $p_c = 1$ (or $p_c = 1 - o(1)$) because when selecting two identical parents for crossover, a standard bit mutation is performed. For any arbitrary but fixed population member x, the expected waiting time for witnessing a standard bit mutation of x is $\mathcal{O}(n^2)$ for GSEMO as both parents are chosen as x with probability at least $1/\mu^2 = \Omega(1/n^2)$. For NSGA-II the expected waiting time is $\mathcal{O}(\mu^3)$ since four search points are selected across two binary tournaments, and a copy of x is created if all four search points are identical to x. The probability of creating a clone of x after crossover in $\mu/2$ binary tournaments is at least $1-(1-1/\mu^4)^{\mu/2} = \Omega(1/\mu^3)$. Obviously, $\mathcal{O}(\mu^3)$ is polynomial if μ is polynomial. We have shown that crossover is a vital operator on these functions, as simply finding any Pareto-optimal point requires exponential time for GSEMO and NSGA-II when disabling crossover ($p_c = 0$), with overwhelming probability and in expectation. This generalises to all elitist black-box algorithms, even when allowing arbitrary unbiased mutation operators.

While previous work has mostly used crossover to speed up filling the Pareto set [56,57,2], our work shows that crossover can also be essential for discovering the Pareto set in the first place. Another novel aspect is the consideration of hypermutation (which was only considered for GSEMO in [35]) and we showed that for $\mathrm{uRR_{MO}}$ allowing the use of hypermutation does not avoid exponential optimisation times. The same holds when using GSEMO with hypermutation as the only mutation operator on $\mathrm{RR_{MO}}$, but it remains an open problem whether this also holds for NSGA-II.

We are hopeful that our results and the proofs may serve as stepping stones towards a better understanding of the role of crossover in EMO, and that this paves the way for identifying more natural examples where crossover is beneficial, in the same way that this was achieved for single-objective optimisation.

Our results have highlighted that common variation operators have weaknesses that can be mitigated by using other operators that have complementary strengths. Standard bit mutations are unable to make large jumps, where many bits have to be flipped in one mutation. Hypermutation is also unable to do this if the parameter r is small. But if the parameter r is large, hypermutation may struggle to keep bits fixed that should not be changed. Crossover operators are able to make larger changes, but they need mutation in order to create novelty. In our positive results, the ability to apply both crossover and mutation (i. e. being able to make large and small changes) was crucial for efficient optimisation. Finally, our royal road functions are tailored towards one specific crossover operator, hence choosing the right type of crossover operator is important as well. A lesson learnt from these analyses is that it can be beneficial to combine several operators, as previously shown for only single-objective optimisation (see, e. g. [9,51,3] and references therein) and, in particular, to try different crossover and mutation operators in case the problem in hand is not well understood.

CRediT authorship contribution statement

Duc-Cuong Dang: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing. **Andre Opris:** Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing. **Dirk Sudholt:** Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors thank Bahare Salehi for her contributions to the preliminary version of this article [15]. This work benefited from discussions at Dagstuhl seminar 22081.

References

[1] G. Badkobeh, P.K. Lehre, D. Sudholt, Black-box complexity of parallel search with distributed populations, in: Proceedings of the Foundations of Genetic Algorithms (FOGA '15), ACM Press, 2015, pp. 3–15.
[2] C. Bian, C. Qian, Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection, in: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN '22), in: Lecture Notes in Computer Science, vol. 13399, Springer, 2022, pp. 428–441.
[3] L. Branson, A.M. Sutton, Focused jump-and-repair constraint handling for fixed-parameter tractable graph problems closed under induced subgraphs, Theor. Comput. Sci. (ISSN 0304-3975) 951 (2023) 113719.
[4] E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg Hyper-heuristics, An emerging direction in modern search technology, in: Handbook of Metaheuristics, in: International Series in Operations Research & Management Science, vol. 57, Kluwer / Springer, 2003, pp. 457–474.
[5] B. Case, P.K. Lehre, Self-adaptation in nonelitist evolutionary algorithms on discrete problems with unknown structure, IEEE Trans. Evol. Comput. 24 (4) (2020) 650–663.
[6] S. Cerf, B. Doerr, B. Hebras, Y. Kahane, S. Wietheger, The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-23, International Joint Conferences on Artificial Intelligence Organization, 8 2023, pp. 5522–5530.
[7] D. Corus, P.S. Oliveto, Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms, IEEE Trans. Evol. Comput. 22 (5) (2018) 720–732.
[8] D. Corus, P.S. Oliveto, On the benefits of populations for the exploitation speed of standard steady-state genetic algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2019), ACM, New York, NY, USA, 2019, pp. 1452–1460.
[9] D. Corus, J. He, T. Jansen, P.S. Oliveto, D. Sudholt, C. Zarges, On easiest functions for mutation operators in bio-inspired optimisation, Algorithmica 78 (2) (2017) 714–740.
[10] D. Corus, P.S. Oliveto, D. Yazdani, Artificial immune systems can find arbitrarily good approximations for the NP-hard number partitioning problem, Artif. Intell. 274 (2019) 180–196.
[11] D. Corus, A. Lissovoi, P.S. Oliveto, C. Witt, On steady-state evolutionary algorithms and selective pressure: why inverse rank-based allocation of reproductive trials is best, ACM Trans. Evol. Learn. Optim. 1 (1) (2021) 1–38.
[12] E. Covantes Osuna, W. Gao, F. Neumann, D. Sudholt, Design and analysis of diversity-based parent selection schemes for speeding up evolutionary multi-objective optimisation, Theor. Comput. Sci. 832 (2020) 123–142.
[13] D.-C. Dang, P.K. Lehre, Self-adaptation of mutation rates in non-elitist populations, in: Proceeding of the International Conference on Parallel Problem Solving from Nature (PPSN '16), in: Lecture Notes in Computer Science, vol. 9921, Springer, 2016, pp. 803–813.
[14] D.-C. Dang, T. Friedrich, T. Kötzing, M.S. Krejca, P.K. Lehre, P.S. Oliveto, D. Sudholt, A.M. Sutton, Escaping local optima using crossover with emergent diversity, IEEE Trans. Evol. Comput. 22 (2017) 484–497.
[15] D.-C. Dang, A. Opris, B. Salehi, D. Sudholt, A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 12390–12398.
[16] L.N. de Castro, J. Timmis, Artificial Immune Systems: A New Computational Intelligence Paradigm, Springer, ISBN 978-1-85233-594-6, 2002.
[17] K. Deb, NSGA-II source code in C, version 1.1.6, https://www.egr.msu.edu/~kdeb/codes/nsga2/nsga2-gnuplot-v1.1.6.tar.gz, 2011. (Accessed 15 August 2022).
[18] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[19] B. Doerr, C. Doerr, Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm, Algorithmica 80 (5) (2018) 1658–1709.
[20] B. Doerr, Z. Qu, A first runtime analysis of the NSGA-II on a multimodal problem, in: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN '22), in: Lecture Notes in Computer Science, vol. 13399, Springer, 2022, pp. 399–412.
[21] B. Doerr, Z. Qu, Runtime analysis for the NSGA-II: provable speed-ups from crossover, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023, AAAI Press, 2023, pp. 12399–12407.
[22] B. Doerr, W. Zheng, Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2021, AAAI Press, 2021, pp. 12293–12301.
[23] B. Doerr, E. Happ, C. Klein, Crossover can provably be useful in evolutionary computation, Theor. Comput. Sci. 425 (2012) 17–33.
[24] B. Doerr, C. Doerr, F. Ebel, From black-box complexity to designing new genetic algorithms, Theor. Comput. Sci. 567 (2015) 87–104.
[25] B. Doerr, H.P. Le, R. Makhmara, T.D. Nguyen, Fast genetic algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17), ACM Press, 2017, pp. 777–784.
[26] B. Doerr, C. Gießen, C. Witt, J. Yang, The $(1+\lambda)$ Evolutionary Algorithm with self-adjusting mutation rate, Algorithmica 81 (2) (2019) 593–631.

[27] B. Doerr, C. Doerr, J. Yang, Optimal parameter choices via precise black-box analysis, Theor. Comput. Sci. 801 (2020) 1–34.
[28] B. Doerr, O.E. Hadri, A. Pinard, The $(1 + (\lambda, \lambda))$ global SEMO algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22), ACM Press, 2022, pp. 520–528.
[29] C. Doerr, J. Lengler, Introducing elitist black-box models: when does elitist behavior weaken the performance of evolutionary algorithms?, Evol. Comput. 25 (4) (2017) 587–606.
[30] S. Fischer, I. Wegener, The one-dimensional Ising model: mutation versus recombination, Theor. Comput. Sci. 344 (2–3) (2005) 208–225.
[31] S. Forrest, M. Mitchell, Relative building block fitness and the building block hypotheses, in: Proceedings of the Foundations of Genetic Algorithms (FOGA '93), Morgan Kaufmann, 1993, pp. 109–126.
[32] O. Giel, P.K. Lehre, On the effect of populations in evolutionary multi-objective optimisation, Evol. Comput. 18 (3) (2010) 335–356.
[33] J. He, X. Yao, A study of drift analysis for estimating computation time of evolutionary algorithms, Nat. Comput. 3 (2004) 21–35.
[34] Z. Huang, Y. Zhou, Runtime analysis of somatic contiguous hypermutation operators in MOEA/D framework, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 2359–2366.
[35] Z. Huang, Y. Zhou, Runtime analysis of immune-inspired hypermutation operators in evolutionary multi-objective optimization, Swarm Evol. Comput. 65 (2021) 100934.
[36] T. Jansen, I. Wegener, On the analysis of evolutionary algorithms—a proof that crossover really can help, Algorithmica 34 (1) (2002) 47–66.
[37] T. Jansen, I. Wegener, Real royal road functions—where crossover provably is essential, Discrete Appl. Math. 149 (2005) 111–125.
[38] T. Jansen, C. Zarges, Analyzing different variants of immune inspired somatic contiguous hypermutations, Theor. Comput. Sci. 412 (6) (2011) 517–533.
[39] T. Jansen, C. Zarges, Artificial immune systems for optimisation, in: Genetic and Evolutionary Computation Conference, GECCO '14, Companion Material Proceedings, ACM, 2014, pp. 749–764.
[40] J. Kelsey, J. Timmis, Immune inspired somatic contiguous hypermutations for function optimisation, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003), in: LNCS, vol. 2723, Springer, 2003, pp. 207–218.
[41] T. Kötzing, D. Sudholt, M. Theile, How crossover helps in pseudo-Boolean optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '11), ACM Press, 2011, pp. 989–996.
[42] M. Laumanns, L. Thiele, E. Zitzler, Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions, IEEE Trans. Evol. Comput. 8 (2) (2004) 170–182.
[43] P.K. Lehre, E. Özcan, A runtime analysis of simple hyper-heuristics: to mix or not to mix operators, in: Proceedings of the Foundations of Genetic Algorithms (FOGA 13), ACM, 2013, pp. 97–104.
[44] P.K. Lehre, C. Witt, Black-box search by unbiased variation, Algorithmica 64 (4) (2012) 623–642.
[45] J. Lengler, Drift analysis, in: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, in: Natural Computing Series, Springer, 2020, pp. 89–131.
[46] J. Lengler, A general dichotomy of evolutionary algorithms on monotone functions, IEEE Trans. Evol. Comput. 24 (6) (2020) 995–1009.
[47] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2019, AAAI Press, 2019, pp. 2322–2329.
[48] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, How the duration of the learning period affects the performance of random gradient selection hyper-heuristics, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 2376–2383.
[49] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, When move acceptance selection hyper-heuristics outperform metropolis and elitist evolutionary algorithms and when not, Artif. Intell. 314 (2023) 103804.
[50] M. Mitchell, S. Forrest, J.H. Holland, The royal road function for genetic algorithms: fitness landscapes and GA performance, in: Proceedings of the First European Conference on Artificial Life, MIT Press, 1992, pp. 245–254.
[51] P.T.H. Nguyen, D. Sudholt, Memetic algorithms outperform evolutionary algorithms in multimodal optimisation, Artif. Intell. 287 (2020) 103345.
[52] P.S. Oliveto, C. Witt, Simplified drift analysis for proving lower bounds in evolutionary computation, Algorithmica 59 (3) (2011) 369–386.
[53] P.S. Oliveto, C. Witt, Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation. ArXiv e-prints, 2012.
[54] P.S. Oliveto, D. Sudholt, C. Witt, Tight bounds on the expected runtime of a standard steady state genetic algorithm, Algorithmica 84 (6) (2022) 1603–1658.
[55] T. Paixão, G. Badkobeh, N. Barton, D. Corus, D.-C. Dang, T. Friedrich, P.K. Lehre, D. Sudholt, A.M. Sutton, B. Trubenova, Toward a unifying framework for evolutionary processes, J. Theor. Biol. 383 (2015) 28–43.
[56] C. Qian, Y. Yu, Z.-H. Zhou, An analysis on recombination in multi-objective evolutionary optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '11), ACM Press, 2011, pp. 2051–2058.
[57] C. Qian, Y. Yu, Z. Zhou, An analysis on recombination in multi-objective evolutionary optimization, Artif. Intell. 204 (2013) 99–119.
[58] C. Qian, C. Bian, C. Feng, Subset selection by Pareto optimization with recombination, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2020, AAAI Press, 2020, pp. 2408–2415.
[59] T. Storch, I. Wegener, Real royal road functions for constant population size, Theor. Comput. Sci. 320 (2004) 123–134.
[60] D. Sudholt, Crossover is provably essential for the Ising model on trees, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05), ACM Press, 2005, pp. 1161–1167.
[61] D. Sudholt, How crossover speeds up building-block assembly in genetic algorithms, Evol. Comput. 25 (2) (2017) 237–274.
[62] D. Sudholt, The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses, in: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 359–404.
[63] A.M. Sutton, Fixed-parameter tractability of crossover: steady-state GAs on the closest string problem, Algorithmica 83 (4) (2021) 1138–1163.
[64] I. Wegener, Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions, in: Evolutionary Optimization, Kluwer, 2002, pp. 349–369.
[65] W. Zheng, B. Doerr, Better approximation guarantees for the NSGA-II by using the current crowding distance, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22), ACM Press, 2022, pp. 611–619.
[66] W. Zheng, Y. Liu, B. Doerr, A first mathematical runtime analysis of the non-dominated sorting genetic algorithm II (NSGA-II), in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2022, AAAI Press, 2022, pp. 10408–10416.