A Proof That Using Crossover Can Guarantee Exponential Speed-Ups in Evolutionary Multi-Objective Optimisation

Duc-Cuong Dang[1], **Andre Opris**[1], **Bahare Salehi**[1,2], **Dirk Sudholt**[1]

[1] Chair of Algorithms for Intelligent Systems, University of Passau, Passau, Germany
[2] School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran
{duccuong.dang,andre.opris,dirk.sudholt}@uni-passau.de,salehi.bahaar@gmail.com

Abstract

Evolutionary algorithms are popular algorithms for multiobjective optimisation (also called Pareto optimisation) as they use a population to store trade-offs between different objectives. Despite their popularity, the theoretical foundation of multiobjective 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 well-known EMO algorithms GSEMO and NSGA-II to showcase the possible advantages of crossover. We propose a class of problems on which these EMO algorithms using crossover find the Pareto set in expected polynomial time. In sharp contrast, they and many other EMO algorithms without crossover require exponential time to even find a single Pareto-optimal point. This is the first example of an exponential performance gap through the use of crossover for the widely used NSGA-II algorithm.

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 (Deb et al. 2002) are well suited for this task due as they are able to use their population to store multiple trade-offs between objectives. However, the theoretical understanding of evolutionary multiobjective optimisation (EMO) is lagging far behind its success in practice (Zheng, Liu, and Doerr 2022). There is little understanding on how the choice of search operators and parameters affects performance in multiobjective 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 (Doerr et al. 2017), changing the order of crossover and mutation and amplifying the probability of improving mutations (Doerr, Doerr, and Ebel 2015), parent selection preferring worse search points (Corus et al. 2021) or adapting mutation rates during the run (Doerr et al. 2019).

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 (Paixão et al. 2015). In single-objective optimisation there is a body of works on the usefulness of crossover (Sudholt 2020, Section 8.4) on illustrative pseudo-Boolean example problems (Jansen and Wegener 2005; Storch and Wegener 2004; Kötzing, Sudholt, and Theile 2011; Dang et al. 2017; Sudholt 2017; Corus and Oliveto 2018; Doerr, Doerr, and Ebel 2015) and problems from combinatorial optimisation such as shortest paths (Doerr, Happ, and Klein 2012), graph colouring problems (Fischer and Wegener 2005; Sudholt 2005) and the closest string problem (Sutton 2021).

However, in EMO results are scarce. Understanding and 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 (Deb et al. 2002) 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 (Zheng, Liu, and Doerr 2022).

Our contribution: We demonstrate the possible advantages of crossover for EMO by presenting an example of an n-bit pseudo-Boolean function $\mathrm{RR}_{\mathrm{MO}}$ on which the use of crossover has a drastic effect on performance: EMO algorithms like GSEMO and NSGA-II using crossover can find the Pareto set of $\mathrm{RR}_{\mathrm{MO}}$ in expected time $\mathcal{O}(n^4)$, whereas if crossover is disabled, they require expected exponential time to even find a single Pareto-optimal individual. 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 (2023) showed a polynomial gap for the use of crossover in NSGA-II.

Our result showcases the potential benefits of crossover on a function designed to serve as a "royal road" for the success of crossover, that is, an illustrative example of a problem where the use of crossover is essential. The design of

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

RR_{MO} is deliberately simple to support a rigorous theoretical analysis and to be suited for teaching purposes. We study GSEMO as a simple algorithm and present a more involved analysis for NSGA-II as the best known EMO algorithm. Our hardness results apply to a broad class of EMO algorithms to show that all (unbiased) mutation operators are ineffective. Along the way, we refine and generalise previous arguments from the analysis of NSGA-II (Lemma 7) 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.

Related work: In single-objective optimisation, the first proof that using crossover can speed up EAs was provided by Jansen and Wegener (2002) for the function class 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 (Kötzing, Sudholt, and Theile 2011; Dang et al. 2017). The first exponential performance gap was shown by Jansen and Wegener (2005) for a function REALROYALROAD, which encourages EAs to evolve strings with all 1-bits gathered in a single block, and then 1-point crossover can easily assemble the optimal string. A simple EA with 1-point crossover optimises REALROYALROAD in expected time $O(n^4)$, while all mutation-only EAs need exponential time with overwhelming probability. Advantages through crossover were also proven for combinatorial problems: shortest paths (Doerr, Happ, and Klein 2012), graph colouring problems (Fischer and Wegener 2005; Sudholt 2005) and the closest string problem (Sutton 2021). Crossover speeds up hill climbing on $\text{ONEMAX}(x)$ that simply counts the number of ones in x by a constant factor (Sudholt 2017; Corus and Oliveto 2018). A cleverly designed EA called (1+(λ,λ)) GA outperforms the best mutation-only EAs on ONEMAX by a factor of $\mathcal{O}(\log n)$ (Doerr, Doerr, and Ebel 2015; Doerr and Doerr 2018). Finally, crossover increases robustness on difficult monotone pseudo-Boolean functions (Lengler 2020).

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, Thiele, and Zitzler (2004) 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 (2010) 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. (2020) 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 (2021) introduced the ONEJUMPZEROJUMP function, which generalises JUMP_k to the multiobjective 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 (Deb et al. 2002) is a practical and hugely popular reference algorithm for EMO, however its theoretical analysis only succeeded recently. Zheng, Liu, and Doerr (2022) 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 2022), and shown to be due to the consideration of the same fitness multiple times. Thus an incremental procedure to compute the crowding distances was proposed as an improvement to the standard algorithm. Doerr and Qu (2022) showed that heavy-tailed mutations presented for the single objective settings (Doerr et al. 2017) are also highly beneficial for EMO. Our work is similar to these papers in the spirit, however we generalise REALROYALROAD (Jansen and Wegener 2005) to show the advantage of crossover.

Only a few works rigorously prove the advantages of crossover in EMO, despite experimentally they are noticeable (e. g. Doerr and Qu (2022)). Qian, Yu, and Zhou (2011) 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 multiobjective minimum spanning trees (Qian, Yu, and Zhou 2013). Qian, Bian, and Feng (2020) compared two variants of GSEMO with and without crossover, called POSS and PORSS respectively, for the subset 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. Bian and Qian (2022) introduced a new parent selection strategy named stochastic tournament selection using crowding distance to favour diverse parents as in Covantes Osuna et al. (2020) 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. (2020) already showed that for SEMO on LOTZ the same performance guarantee can be obtained through tailoring the parent selection mechanism and that crossover is not required for an $\mathcal{O}(n^2)$ bound. Finally, Doerr, Hadri, and Pinard (2022) introduced the (1+(λ,λ)) GSEMO algorithm and proved that it optimizes OMM in expected time $\mathcal{O}(n^2)$, a speedup of order $\log n$ compared to GSEMO.

Preliminaries

Consider maximising a function $f(x) = (f_1(x), \dots, f_d(x))$ where $f_i \colon \{0,1\}^n \to \mathbb{N}_0$ 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*

Algorithm 1 GSEMO Algorithm

Initialize $P_0 := \{s\}$ where $s \sim \mathrm{Unif}(\{0,1\}^n)$;
for $t := 0$ **to** ∞ **do**
  Sample $p_1 \sim \mathrm{Unif}(P_t)$;
  Sample $u \sim \mathrm{Unif}([0,1])$;
  if $(u < p_c)$ **then**
    Sample $p_2 \sim \mathrm{Unif}(P_t)$;
    Create s by 1-point crossover between p_1 and p_2;
  else
    Create s as a copy of p_1;
  Create s' by bitwise mutation on s with rate $1/n$;
  if (s' is **not** dominated by any individual in P_t) **then**
    Create the next population $P_{t+1} := P_t \cup \{s\}$;
    Remove all $x \in P_{t+1}$ weakly dominated by s';

optimal. A set of these solutions that cover all possible non-dominated fitness values is called a *Pareto/optimal set* of f.

The GSEMO algorithm with 1-point crossover is shown in Algorithm 1. Starting from a randomly generated solution, in each generation a new search point s' is created by crossover, with probability $p_c \in (0,1)$, on parents selected uniformly at random. Mutation then flips each bit independently with probability $1/n$. If s' is not dominated by any solutions of the current population P_t then it is inserted, and those weakly dominated by s' are removed from the population. Thus the population size $|P_t|$ may vary over time.

The NSGA-II (Deb et al. 2002; Deb 2011) with 1-point crossover is shown in Algorithm 2. In each generation, a population Q_t of μ offspring is created by binary tournament, crossover and mutation. Note that the 1-point crossover, if applied with probability $p_c \in (0,1)$, produces two offspring solutions, and that the binary tournaments (with replacement) in line 6 uses the same criteria as the sorting in line 17. The merged population R_t of both parents and offspring are then partitioned into layers $F^1_{t+1}, F^2_{t+1}, \ldots$ of non-dominated solutions and the 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.

For a set $S = (x_1, x_2, \ldots, x_{|S|})$ of search points the crowding distances are computed for each objective and then summed up. Thus S is sorted separately for each objective f_k ($k \le d$) 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 between f_k of the first and last ranked. Let $S_k = (x_{k_1}, x_{k_2}, \ldots, x_{k_{|S|}})$ denote the elements of S sorted in descending order w. r. t. f_k, then $\textsc{cDist}(x_i, S) := \sum_{k=1}^{d} \textsc{cDist}_k(x_i, S)$ where

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

In a nutshell, NSGA-II is a standard (μ+λ) GA with

Algorithm 2 NSGA-II Algorithm (Deb et al. 2002)

1: Initialize $P_0 \sim \mathrm{Unif}((\{0,1\}^n)^\mu)$;
2: Partition P_0 into layers $F^1_0, F^2_0, \ldots$ of non-dominated fitnesses, then for each layer F^i_0 compute the crowding distance $\textsc{cDist}(x, F^i_0)$ 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 $u \sim \mathrm{Unif}([0,1])$;
8: **if** $(u < p_c)$ **then**
9: Create s_1, s_2 by 1-point 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 bitwise mutation on s_1 with rate $1/n$;
13: Create s'_2 by bitwise 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 $F^1_{t+1}, F^2_{t+1}, \ldots$ of non-dominated fitnesses, then for each layer F^i_{t+1} compute $\textsc{cDist}(x, F^i_{t+1})$ for each $x \in F^i_{t+1}$;
17: Sort R_t lexicographically by $(1/i, \textsc{cDist}(x, F^i_{t+1}))$;
18: Create the next population $P_{t+1} := (R[1], \ldots, R[\mu])$;

Algorithm 3 Elitist (μ+λ) black-box algorithm.

Initialize $P_0 \sim \mathrm{Unif}((\{0,1\}^n)^\mu)$;
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 $P_{t+1} := (R[1], \ldots, R[\mu])$;

$\lambda = \mu$ generalized to as multiple-objective setting. This is done 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. The algorithm therefore fits in the elitist black-box model of Algorithm 3.

In this model, adapted from Doerr and Lengler (2017), 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. NSGA-II uses the non-dominated sorting and the crowding distance measure for the ranking function, but other algorithms can have different choices.

The Multi-Objective Royal Road Function

The idea behind the design of our 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. We define the number of leading zeros, $\mathrm{LZ}(x)$ as the length of the longest prefix in x that contains

only zeros. Similarly, the number of trailing zeros, $\mathrm{TZ}(x)$, gives the length of the longest suffix of zeros in x. For example, for $x = 0010110$ we have $\mathrm{LZ}(x) = 2$, $\mathrm{TZ}(x) = 1$.

Our REALROYALROAD function in multi-objective setting is described by a trade-off between leading zeros and leading ones 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 recombination but for which common mutation operators require exponential time.

Definition 1. *For all $n \in \mathbb{N}$ divisible by 5 the bi-objective function $\mathrm{RR}_{\mathrm{MO}} \colon \{0,1\}^n \to \mathbb{N}_0^2$ 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 1. 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 $y \notin F' \cup F$.

The following lemma bounds the number of non-dominated solutions contained in any population.

Lemma 2. *If S is a set of non-dominated solutions of $\mathrm{RR}_{\mathrm{MO}}$ with positive fitness then $|S| \leq n$.*

Proof. By our previous observations, S may only contain search points with the same number of ones, denoted by k. 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 at most n (using $k \geq 1$ and the fact that there are at most $n-1$ zeros), the claim follows. □

Hardness for EAs without Crossover

We show that setting $p_c = 0$ makes GSEMO and NSGA-II highly inefficient, even for discovering the Pareto font.

Theorem 3. *GSEMO (Algorithm 1) with $p_c = 0$ requires at least $n^{\Omega(n)}$ evaluations in expectation to find any Pareto-optimal search point for $\mathrm{RR}_{\mathrm{MO}}$.*

Proof. By classical Chernoff bounds, the probability of initialising the algorithm with a search point of at most $3n/5$ ones, i. e. belonging to $G \setminus F$, 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 bitwise 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 $|F| \cdot n^{-n/5} = O(n^{-n/5+1})$. By the law of total probability, the expected number of evaluations required for this phase is at least $(1 - 2^{-\Omega(n)}) \cdot \Omega(n^{n/5-1}) = n^{\Omega(n)}$. □

Theorem 4. *NSGA-II (Algorithm 2) with $p_c = 0$ and $\mu \in \mathrm{poly}(n)$ requires at least $n^{\Omega(n)}$ generations in expectation to find any Pareto-optimal search point on $\mathrm{RR}_{\mathrm{MO}}$.*

Proof. We follow the same arguments as the ones in the proof of Theorem 3, with only minor differences due to the fixed population size. The initialisation phase can be skipped with probability $\mu 2^{-\Omega(n)}$ by an union bound, but this is still $o(1)$ since $\mu \in \mathrm{poly}(n)$. The second phase then starts with μ search points of strictly positive objective values and 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 complete this phase, and overall the expected number of generations is at least $(1 - o(1))(1 + n^{\Omega(n)}) = n^{\Omega(n)}$. □

Furthermore, we prove that the general framework of Algorithm 3 also requires exponential optimisation time in expectation to create a first Pareto optimal point of $\mathrm{RR}_{\mathrm{MO}}$ if only unary unbiased variation operators are used.

Theorem 5. *Any black-box algorithm that fits the model of Algorithm 3 with $\mu = \mathrm{poly}(n)$ and only uses unary unbiased variation operators for choosing the distribution D_t requires at least $5^{\Omega(n)}/\lambda$ generations, or $5^{\Omega(n)}$ fitness evaluations, in expectation to find any Pareto-optimal point of $\mathrm{RR}_{\mathrm{MO}}$.*

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 $\mu = \mathrm{poly}(n)$ initial search points, the probability of initialising the first μ search points in G' is $1 - \mu \cdot 2^{-\Omega(n)}$. Then, owing to elitism the algorithm will only accept points in $G' \cup F$.

According to Lemma 1 in (Doerr, Doerr, and Yang 2020) every unary unbiased variation operator $\mathrm{op}(x)$ 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. 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, $n/5 \leq H(x,y) \leq 4n/5$. Thus, even when the radius r is chosen as $r := H(x,y)$, the probability that $\mathrm{op}(x)$

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 expected number of evaluations until a Pareto optimal search point is found is thus at least $1/p = 5^{\Omega(n)}$. The expected number of generations follows since every generation makes λ evaluations. □

Use of Crossover Implies Polynomial Time

While $\mathrm{RR}_{\mathrm{MO}}$ 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 $\mathrm{RR}_{\mathrm{MO}}$ in expected polynomial time.

Analysis of GSEMO

Theorem 6. *GSEMO (Algorithm 1) with $p_c \in (0,1)$ 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* $\mathrm{RR}_{\mathrm{MO}}$.

Proof. We use the well-known method of typical runs (Wegener 2002, 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 and crossover, if executed, has no effect. We show that mutation tends to drive the algorithm towards G very quickly. Let x_t denote the single search point at time t and X_t be its number of ones if x_t it is not in G, and zero otherwise. If $X_t > 0$ then $|x_t|_1 > 3n/5$ and in expectation at least $3/5$ ones are flipped to zero and at most $2/5$ zeros are flipped to one by mutation, so

$$\mathrm{E}\left[X_t - X_{t+1} \mid X_t > 0\right] \geq 3/5 - 2/5 = 1/5 =: \delta.$$

Let $T := \inf\{t \geq 0 \mid X_t = 0\} = \inf\{t \geq 0 \mid x_t \in G\}$. Then by the additive drift theorem (see (He and Yao 2004)), it holds that $\mathrm{E}\left[T \mid X_0\right] \leq \frac{X_0}{\delta} \leq 5n$ (since $X_0 \leq n$). Consequently, the expected number of generations for finding a search point in G is $1 + 2^{-\Omega(n)} \cdot 5n = 1 + o(1)$.

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 $i \in \{0, \ldots, 3n/5-1\}$ 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 2, 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 1-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 $\mathcal{O}(\frac{n}{p_c})$.

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 $\mathcal{O}(\frac{n^4}{1-p_c})$.

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. □

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.

The following lemma shows that with a sufficient large population size, search points of the first-ranked layer, with a specific lattice structure in the objective space, are protected between generations. No assumption about the search space is required, thus the result also holds for search spaces other than bitstrings. The proof generalises the arguments from Zheng, Liu, and Doerr (2022) which correspond to the specific application of the lemma with $C = 0, D = n$.

Lemma 7. *Consider two consecutive generations t and $t+1$ of the NSGA-II optimising a biobjective function $f(x) := (f_1(x), f_2(x))$. Suppose that there are natural numbers $C, D \in \mathbb{N}_0$ such that every search point in F_t^1 and F_{t+1}^1 has a fitness vector of $(C+\ell, C+m-\ell)$ for $\ell, m \in \{0, \dots, D\}$ and $\ell \leq m$, then it holds that*

(i) At most $4(D+1)$ individuals in F_t^1 have positive crowding distances, and the same statement holds for F_{t+1}^1.

(ii) If $\mu \geq 4(D+1)$ and there exists a $x \in P_t$ with $f(x) = (C+k, C+D-k)$ for some $k \in \{0, \dots, D\}$ then there exists a $y \in P_{t+1}$ with $f(y) = f(x)$.

Proof. (i) It suffices to show the result for F_t^1, as the same arguments apply for F_{t+1}^1. First we determine the maximal number of distinct fitness values of individuals in F_t^1. Define $M := \{(f_1(x), f_2(x)) \mid x \in F_t^1\}$ and we claim that $|M| \leq D+1$. Let $M_1 := \{f_1(x) \mid x \in F_t^1\}$ then for every $a \in M_1$ there is a unique $b \in \mathbb{N}$ with $(a,b) \in M$. This is because if there are $(a,b_1) \in M$ and $(a,b_2) \in M$ with $b_1 < b_2$ then there are $x, y \in F_t^1$ with $f(x) = (a,b_1)$, $f(y) = (a,b_2)$, thus $x \prec y$ and this contradicts the definition of F_t^1. So, $|M| \leq |M_1| \leq D+1$ and the last inequality is due to $M_1 \subseteq \{C, C+1, \dots, C+D\}$. So there are at most $D+1$ individuals in F_t^1 with distinct fitness values.

To obtain the result, it now suffices to show that for each $(a,b) \in M$ at most four individuals in F_t^1 with this fitness have a positive crowding distance. Let $K := |F_t^1|$ and assume that $x_1, \dots, x_K$ are the individuals in F_t^1, 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_t^1 with respect to f_1 and f_2, respectively. Suppose there are $L \geq 1$ individuals in F_t^1 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_1^t 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 $\textsc{cDist}(x, S) = \frac{f_1(x_{1_{i-1}}) - f_1(x_{1_{i+1}})}{f_k(x_{1_1}) - f_k(x_{1_K})} + \frac{f_2(x_{2_{j-1}}) - f_2(x_{2_{j+1}})}{f_2(x_{2_1}) - f_2(x_{2_K})} = 0$. 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) It follows from (i) and $\mu \geq 4(D+1)$ that P_{t+1} will contain every search point from F_{t+1}^1 with positive crowding distances. Also for each fitness value (a,b) of F_{t+1}^1 there must exist a search point of the layer that has that fitness and a positive crowding distance, i.e. the cardinality of $\{x_{1_r}, x_{1_{r+L-1}}, x_{2_s}, x_{2_{s+L-1}}\}$ is at least 1. Since $x \in F_t^1$ with $f(x) = (C+k, C+D-k)$, then the only reason the fitness vector $f(x)$ would not be contained in the image of F_{t+1}^1 is if it is dominated by some $z \in F_{t+1}^1$. Denoting $f(z) = (C+h, C+m-h)$, we show that no such z exists because the two cases: (a) $C+h \geq C+k \wedge C+m-h > C+D-k$, or equivalently $m-D > h-k \geq 0$, and (b) $C+h > C+k \wedge C+m-h \geq C+D-k$, or equivalently $m-D \geq h-k > 0$, contradict our assumption that $m \leq D$. The non-existence of z implies that there must exist $y \in F_{t+1}^1 \subseteq P_{t+1}$ with $f(y) = f(x)$. □

The statement of Lemma 7 holds for any implementation of the calculation of crowding distances. However, the factor 4 in front of $D+1$ there can be reduced to 2 if a stable sort algorithm is used to sort each objective, or the asymmetric version of the crowding distance (Chu and Yu 2018) is used.

Theorem 8. *NSGA-II (Algorithm 2) with $p_c \in (0,1)$ 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 10 in (Badkobeh, Lehre, and Sudholt 2015), 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'.

Let $x'' \in F_t^1$ be a search point with $3n/5$ ones and maximum value of $\mathrm{LZ}(x)+\mathrm{TZ}(x) =: 2n/5-i$ for $i \in \{1,\dots,2n/5\}$ and positive crowding distance. If x'' appears as the first competitor in a binary tournament (which happens with probability $1/\mu$), and the second competitor has zero crowding distance (which happens with probability at least $1-\frac{8n/5+4}{\mu} \geq 1-\frac{8n/5+4}{2n+5} = 1/5$ as there are at most $8n/4+5$ individuals with positive crowding distances by (i) in Lemma 7), then x'' wins the tournament. The same holds for swapping roles of first and second competitor, and 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 $1-(1-\frac{2}{5\mu})^2 \geq \frac{4/(5\mu)}{4/(5\mu)+1} = \frac{4}{4+5\mu}$. So, similarly to the proof of Theorem 6, the probability of increasing the maximum value of $\mathrm{LZ}+\mathrm{TZ}$ 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 $\sum_{i=1}^{2n/5-1}\left(1+\frac{2}{\mu s_i'}\right)$ which is

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

Phase 4: Cover F' entirely.

Let y and z be as defined in the proof of Theorem 6 and additionally assume that y has a positive crowding distance in F_t^1. Similarly to the proof of Theorem 6 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 $\frac{4}{4+5\mu}\cdot\frac{1-p_c}{en^2} =: s_i''$, and the probability of at least one success in $\mu/2$ trials is at least $\frac{s_i''\mu/2}{s_i''\mu/2+1}$. Furthermore, applying Lemma 7 (ii) with $C=3n^2/5$ and $D=2n/5$ while noticing that $\mu \geq 2n+5 > 4(D+1)$ implies that a copy of z always survives in future generations. 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 $0^i1^{3n/5}0^{2n/5-i}$ 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 $2\cdot\frac{1}{5}\cdot\frac{1}{\mu}$, 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 $p_c\cdot\frac{4}{25}\cdot\frac{2n/5}{n+1}\cdot\frac{n/5}{\mu}\cdot\frac{1}{\mu}\cdot(1-1/n)^n = \Omega(\frac{np_c}{\mu^2}) =: s$, and among $\mu/2$ pairs produced at least one success occurs with probability $1-(1-s)^{\mu/2} \geq \frac{s\mu/2}{s\mu/2+1}$ and only $1+\frac{2}{\mu s} = \mathcal{O}(\frac{\mu}{np_c})$ generations are required in expectation for this phase.

Phase 6: Cover F entirely.

Once a search point on F is created, the process of covering it is similar to that of covering F' with only minor differences (e. g. applying Lemma 7 with $C=4n^2/5, D=n/5$). 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. □

Conclusions

We have identified the function class $\mathrm{RR}_{\mathrm{MO}}$ as examples on which EMO algorithms GSEMO and NSGA-II using crossover can find the whole Pareto set in expected time $\mathcal{O}(n^4)$ with any constant value of $p_c \in (0,1)$ and population size $2n+5 \leq \mu = \mathcal{O}(n)$ for NSGA-II. More generally, the function class can be optimised in expected polynomial time if $1/p_c$ and μ are polynomials in n. Crossover is a vital operator as simply finding any Pareto-optimal point requires exponential expected time for GSEMO and NSGA-II if crossover is omitted. Theorem 5 on a broad class of elitist EMO algorithms showed that this cannot be remedied by using other unbiased mutation operators.

This is the first proof for an exponential performance gap for the use of crossover in NSGA-II. While previous work has mostly used crossover to speed up filling the Pareto set (Qian, Yu, and Zhou 2011, 2013; Bian and Qian 2022), our work shows that it can also be essential for discovering the Pareto set in the first place. 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.

Acknowledgments

This work benefited from discussions at Dagstuhl seminar 22081. The third author was supported by the Erasmus+ Programme of the European Union

References

Badkobeh, G.; Lehre, P. K.; and Sudholt, D. 2015. Black-box Complexity of Parallel Search with Distributed Popula-

tions. In *Proceedings of the Foundations of Genetic Algorithms (FOGA'15)*, 3–15. ACM Press.

Bian, C.; and Qian, C. 2022. 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)*, volume 13399 of *Lecture Notes in Computer Science*, 428–441. Springer.

Chu, X.; and Yu, X. 2018. Improved Crowding Distance for NSGA-II. *CoRR*, abs/1811.12667.

Corus, D.; Lissovoi, A.; Oliveto, P. S.; and Witt, C. 2021. On Steady-State Evolutionary Algorithms and Selective Pressure: Why Inverse Rank-Based Allocation of Reproductive Trials Is Best. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1): 1–38.

Corus, D.; and Oliveto, P. S. 2018. Standard Steady State Genetic Algorithms Can Hillclimb Faster Than Mutation-Only Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 22(5): 720–732.

Covantes Osuna, E.; Gao, W.; Neumann, F.; and Sudholt, D. 2020. Design and Analysis of Diversity-Based Parent Selection Schemes for Speeding Up Evolutionary Multi-objective Optimisation. *Theoretical Computer Science*, 832: 123–142.

Dang, D.-C.; Friedrich, T.; Kötzing, T.; Krejca, M. S.; Lehre, P. K.; Oliveto, P. S.; Sudholt, D.; and Sutton, A. M. 2017. Escaping Local Optima Using Crossover with Emergent Diversity. *IEEE Transactions on Evolutionary Computation*, 22: 484–497.

Deb, K. 2011. 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. Accessed: 2022-08-15.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197.

Doerr, B.; and Doerr, C. 2018. Optimal Static and Self-Adjusting Parameter Choices for the (1+(λ,λ)) Genetic Algorithm. *Algorithmica*, 80(5): 1658–1709.

Doerr, B.; Doerr, C.; and Ebel, F. 2015. From Black-Box Complexity to Designing New Genetic Algorithms. *Theoretical Computer Science*, 567: 87–104.

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

Doerr, B.; Gießen, C.; Witt, C.; and Yang, J. 2019. The (1+λ) Evolutionary Algorithm with Self-Adjusting Mutation Rate. *Algorithmica*, 81(2): 593–631.

Doerr, B.; Hadri, O. E.; and Pinard, A. 2022. The (1 + (λ, λ)) global SEMO algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*, 520–528. ACM Press.

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

Doerr, B.; Le, H. P.; Makhmara, R.; and Nguyen, T. D. 2017. Fast Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*, 777–784. ACM Press.

Doerr, B.; and Qu, Z. 2022. 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)*, volume 13399 of *Lecture Notes in Computer Science*, 399–412. Springer.

Doerr, B.; and Qu, Z. 2023. Runtime Analysis for the NSGA-II: Provable Speed-Ups From Crossover. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2023*, (To appear). AAAI Press.

Doerr, B.; and Zheng, W. 2021. Theoretical Analyses of Multi-Objective Evolutionary Algorithms on Multi-Modal Objectives. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2021*, 12293–12301. AAAI Press.

Doerr, C.; and Lengler, J. 2017. Introducing Elitist Black-Box Models: When Does Elitist Behavior Weaken the Performance of Evolutionary Algorithms? *Evolutionary Computation*, 25(4): 587–606.

Fischer, S.; and Wegener, I. 2005. The One-dimensional Ising Model: Mutation versus Recombination. *Theoretical Computer Science*, 344(2–3): 208–225.

Giel, O.; and Lehre, P. K. 2010. On the Effect of Populations in Evolutionary Multi-Objective Optimisation. *Evolutionary Computation*, 18(3): 335–356.

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

Jansen, T.; and Wegener, I. 2002. On the Analysis of Evolutionary Algorithms—A Proof That Crossover Really Can Help. *Algorithmica*, 34(1): 47–66.

Jansen, T.; and Wegener, I. 2005. Real royal road functions—where crossover provably is essential. *Discrete Applied Mathematics*, 149: 111–125.

Kötzing, T.; Sudholt, D.; and Theile, M. 2011. How Crossover Helps in Pseudo-Boolean Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '11)*, 989–996. ACM Press.

Laumanns, M.; Thiele, L.; and Zitzler, E. 2004. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE Transactions on Evolutionary Computation*, 8(2): 170–182.

Lengler, J. 2020. A General Dichotomy of Evolutionary Algorithms on Monotone Functions. *IEEE Transactions on Evolutionary Computation*, 24(6): 995–1009.

Paixão, T.; Badkobeh, G.; Barton, N.; Corus, D.; Dang, D.-C.; Friedrich, T.; Lehre, P. K.; Sudholt, D.; Sutton, A. M.; and Trubenova, B. 2015. A Unified Model of Evolutionary Processes. *Journal of Theoretical Biology*, 383: 28–43.

Qian, C.; Bian, C.; and Feng, C. 2020. Subset Selection by Pareto Optimization with Recombination. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2020*, 2408–2415. AAAI Press.

Qian, C.; Yu, Y.; and Zhou, Z. 2013. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence*, 204: 99–119.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2011. An Analysis on Recombination in Multi-Objective Evolutionary Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '11)*, 2051–2058. ACM Press.

Storch, T.; and Wegener, I. 2004. Real royal road functions for constant population size. *Theoretical Computer Science*, 320: 123–134.

Sudholt, D. 2005. Crossover is Provably Essential for the Ising Model on Trees. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, 1161–1167. ACM Press.

Sudholt, D. 2017. How Crossover Speeds Up Building-Block Assembly in Genetic Algorithms. *Evolutionary Computation*, 25(2): 237–274.

Sudholt, D. 2020. The Benefits of Population Diversity in Evolutionary Algorithms: A Survey of Rigorous Runtime Analyses. In *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, 359–404. Springer.

Sutton, A. M. 2021. Fixed-Parameter Tractability of Crossover: Steady-State GAs on the Closest String Problem. *Algorithmica*, 83(4): 1138–1163.

Wegener, I. 2002. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In *Evolutionary Optimization*, 349–369. Kluwer.

Zheng, W.; and Doerr, B. 2022. Better Approximation Guarantees for the NSGA-II by Using the Current Crowding Distance. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '22)*, 611–619. ACM Press.

Zheng, W.; Liu, Y.; and Doerr, B. 2022. A First Mathematical Runtime Analysis of the Non-dominated Sorting Genetic Algorithm II (NSGA-II). In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2022*, 10408–10416. AAAI Press.