Runtime Analysis of the $(\mu + 1)$ GA: Provable Speed-Ups from Strong Drift towards Diverse Populations

Benjamin Doerr,[1] **Aymen Echarghaoui,**[2] **Mohammed Jamal,**[2] **Martin S. Krejca**[1]

[1]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris,
[2]École Polytechnique, Institut Polytechnique de Paris,
firstname.lastname@polytechnique.edu

Abstract

Most evolutionary algorithms used in practice heavily employ crossover. In contrast, the rigorous understanding of how crossover is beneficial is largely lagging behind. In this work, we make a considerable step forward by analyzing the population dynamics of the $(\mu + 1)$ genetic algorithm when optimizing the JUMP benchmark. We observe (and prove via mathematical means) that once the population contains two different individuals on the local optimum, the diversity in the population increases in expectation. From this drift towards more diverse states, we show that a diversity suitable for crossover to be effective is reached quickly and, more importantly, then persists for a time that is at least exponential in the population size μ. This drastically improves over the previously best known guarantee, which is only quadratic in μ.

Our new understanding of the population dynamics easily gives stronger performance guarantees. In particular, we derive that population sizes logarithmic in the problem size n already suffice to gain an $\Omega(n)$-factor runtime improvement from crossover (previous works achieved comparable bounds only with $\mu = \Theta(n)$ or via a non-standard mutation rate).

1 Introduction

The vast majority of the applications of evolutionary algorithms (EAs) uses crossover, that is, new solutions (offspring) are generated from two existing solutions (parents). Surprisingly, there is very little rigorous evidence for the usefulness of crossover. This is not only a fundamental open question, but more importantly also indicates that our understanding of the working principles of crossover is very limited, which means that there is very little trustworthy advice how to best employ crossover in evolutionary algorithms.

In this work, we shall analyze crossover via theoretical means, more specifically, via mathematical runtime analyses, which have given many important insights and explanations in the past (Neumann and Witt 2010; Auger and Doerr 2011; Jansen 2013; Zhou, Yu, and Qian 2019; Doerr and Neumann 2020). Unfortunately, when it comes to understanding crossover, the mathematical runtime analysis area was not very successful (but we note that other attempts to understand crossover, e.g., the building block hypothesis (Holland 1975) also struggled to explain crossover (Mitchell, Holland, and Forrest 1993)).

The main difficulty towards understanding crossover via mathematical tools are the usually complicated population dynamics. We note that even when only regarding mutation-based algorithms, still the majority of runtime analyses regards algorithms with trivial populations such as *randomized local search* or the $(1 + 1)$ EA. The particular difficulty when adding crossover is that we need to understand the diversity of the population. Crossover can only be effective when sufficiently different, good solutions are available.

This core problem is easily visible in the existing runtime analyses of crossover-based algorithms. That diversity, more precisely, arriving at a state with a diverse population, is the crucial ingredient for profiting from crossover was already discussed very clearly in the first runtime analysis of a crossover-based algorithm (Jansen and Wegener 2002), which analyzes how the $(\mu + 1)$ genetic algorithm (GA) optimizes the JUMP benchmark. Unfortunately, the positive effect of crossover could be shown here only for unrealistically small crossover rates. Doerr, Happ, and Klein (2012), without taking uncommon assumptions, showed an advantage of crossover when solving the all-pairs-shortest-path problem. This problem, asking for a short path between any two vertices, already in the problem definition contains a strong diversity mechanism that in particular prevents the population from converging to a single genotype. Consequently, it is not clear how the insights gained in that work extend to algorithms without diversity mechanisms.

The maybe most convincing proof for an advantage from crossover was given by Dang et al. (2018), who again regarded how the $(\mu + 1)$ GA optimizes JUMP functions. Interestingly, without particular modifications of the algorithm or non-standard parameter settings, the authors managed to show that the $(\mu + 1)$ GA repeatedly arrives at diverse populations and keeps this diversity for a moderate time. More precisely, once the population is concentrated on the local optima of the JUMP function, it takes an expected time of $O(n\mu + \mu^2 \log \mu)$ iterations to reach a population in which all genotype classes contain at most $\mu/2$ individuals. A diversity of the order of magnitude is kept for an expected number of $O(\mu^2)$ iterations. Given these numbers, the best runtime results are obtained for a relatively large population sizes of order $\tilde{\Theta}(n)$, where a runtime gain of roughly a factor

of $n/\log n$ from crossover can be proven.

The key argument of Dang et al. (2018) is that the size of the largest genotype class, once below μ, is at least as likely to decrease as to increase. This behavior resembles an unbiased random walk in the interval $[\mu/2..\mu]$ and is the reason why the diversity is shown to persist for $\Omega(\mu^2)$ iterations.

In this work, we analyze the random process describing the largest genotype class of the $(\mu+1)$ GA on JUMP more carefully. We observe that there is not an unbiased random walk behavior, as pessimistically assumed in the previous work, but that instead it is more likely that the dominant genotype class reduces than increases (Lemma 5). This advantage is strongest when the largest genotype class covers only a constant fraction of the population. Consequently, we show that once the largest genotype class contains at most $\mu/2$ individuals, a diversity of this order of magnitude is kept for an expected number of $\exp(\Omega(\mu))$ iterations.

This result is interesting in its own right as it shows that diverse parent populations are much easier to obtain than what previous works suggest. For the particular problem of how fast the $(\mu+1)$ GA optimizes JUMP functions, we obtain a stronger advantage from crossover (Theorem 8), namely by a factor of $\Omega(n)$, and this already for population sizes $\Omega(\ln n)$. For smaller values of μ, we still obtain a significant speed-up from crossover, namely by a factor of $\exp(\Omega(\mu))$, again comparing favorably with the speed-up of $\Omega(\mu)$ shown by Dang et al. (2018). In addition, our results hold for any constant crossover and mutation rate, further generalizing the results by Dang et al. (2018).

Overall, our results show that also basic evolutionary algorithms without diversity mechanisms or other adjustments are able to reach and then keep a diverse population for a long time. This is clearly good news for the use of crossover, but this lasting diversity might also lead to other advantages such as a more effective exploration of the search space or a lower risk of getting stuck in local optima.

2 Previous Work

Since the first rigorous result showing a benefit of crossover more than two decades (Jansen and Wegener 2002) ago, a variety of results for genetic algorithms (GAs), i.e., evolutionary algorithms that employ crossover, has been proven. Due to the complex population dynamics in such algorithms, many of the results showing an advantage from crossover had to take uncommon assumptions or apply only to settings where it is not clear how well they generalize.

Non-standard parameter values. The runtime analysis of GAs was initiated by Jansen and Wegener (2002), who regarded the JUMP function benchmark. For JUMP functions with problem size n and gap size k, they proved that the mutation-based $(1+1)$ EA requires an expected number of $\Theta(n^k)$ function evaluations to reach the optimum, whereas the $(\mu+1)$ GA with suitable parameters only requires $O(\mu n^2 k^3 + 4^k/p_c)$ evaluations. To prove their result, however, the authors had to assume an unrealistically low crossover rate of $p_c = O(1/(kn))$.

This result was later improved by Kötzing, Sudholt, and Theile (2011), however still assuming $p_c = O(k/n)$ since the proof relied on techniques similar to the one of Jansen and Wegener (2002).

Constant-factor improvements. For the classic ONEMAX benchmark, Sudholt (2012) proved that a variant of the $(\mu+1)$ GA obtains a constant-factor speed-up compared to any reasonable variant of the $(1+1)$ EA. As this algorithm applies crossover only to the best individuals in the population, $\mu = 2$ is the best population size.

Corus and Oliveto (2018) removed the assumption that the best individual is part of any crossover operation. For this more typical $(\mu+1)$ GA, with a crossover rate of 1, the authors prove that speed-ups of up to $25\,\%$ over the expected runtime of the $(1+1)$ EA can be observed.

In a subsequent work, the same authors improved this result by showing speed-ups of at least $60\,\%$ compared to not using crossover (Corus and Oliveto 2020). The speed-ups increase with growing population size (however, not beyond constant-factor improvements), thus showing a potential benefit of a larger population.

Oliveto, Sudholt, and Witt (2022) proved a lower bound for the $(2+1)$ GA on ONEMAX that matches the previous results (Corus and Oliveto 2020). Together, these results show that a population larger than 2 is truly beneficial.

Diversity mechanisms. A common way to try to make crossover more effective is to employ mechanisms that aim at increasing the diversity in the population of the GA. Lehre and Yao (2011) considered the $(\mu+1)$ GA with deterministic crowding, an algorithm they called SSGA. Deterministic crowding ensures that when a newly created individual is at least as good as its parent(s), one of the parents is removed. Lehre and Yao (2011) showed that using crossover with constant crossover rate improves the expected runtime of the $(\mu+1)$ SSGA from at least exponential to quadratic in the problem size. Although this is an impressive speed-up, the considered algorithm uses a crossover that exchanges exactly one component among two parents, thus making rather local changes, which is uncommon for crossover.

Oliveto, He, and Yao (2008) considered the vertex cover problem and analyzed the $(\mu+1)$ EA (without crossover) as well as $(\mu+1)$ randomized local search (RLS) with crossover, both with deterministic crowding. They show that using deterministic crowding without crossover is already sufficient for finding minimal covers on bipartite graphs.

Neumann et al. (2011) considered the single-receiver island model with 1-point crossover. They designed a problem for which a variant of the $(\mu+1)$ GA (for populations of order at most $n/\log^3(n)$) has for each crossover rate at least an exponential runtime, with high probability. In contrast, they proved that the island model optimizes this problem with high probability efficiently. This rather highlights the benefits of the island model in this scenario than of crossover.

Dang et al. (2016) analyzed the expected runtime of the $(\mu+1)$ GA optimizing JUMP functions, for a variety of different diversity mechanisms. The runtime is always better (for appropriate parameters) than that of the mutation-only $(1+1)$ EA. As before, these results heavily rely on the diversity mechanism for crossover to be useful, and it is hard to predict to what extent the particular diversity mechanisms employed are specific to the JUMP benchmark.

Sutton (2021) proved that a multi-start variant of the $(\mu+1)$ GA optimizes the closest-string problem in randomized fixed-parameter tractable (FPT) time. The article also showed that if crossover is removed, there are instances that are not solved in FPT time.

Problem-specific knowledge. Doerr, Happ, and Klein (2012) analyzed the impact of crossover for the all-pairs-shortest-path problem, a classic combinatorial optimization problem, and proved that the algorithm with crossover improves the expected runtime by a factor of $\Omega(\sqrt{n/\log(n)})$. It has to be noted that the all-pairs shortest path problem has a strong implicit diversity mechanism. Subsequent extensions and improvements of this result include (Doerr and Theile 2009; Neumann and Theile 2010; Doerr et al. 2013).

Non-standard uses of crossover. A decade ago, Doerr, Doerr, and Ebel (2013, 2015) proposed the $(1+(\lambda,\lambda))$ GA – a GA that combines mutation with a high rate with a biased crossover as repair mechanism. Doerr and Doerr (2018) proved that the $(1+(\lambda,\lambda))$ GA has an expected runtime on ONEMAX that is lower by a factor of slightly more than $\sqrt{\ln(n)}$ than many other evolutionary algorithms.

Since then, the $(1+(\lambda,\lambda))$ GA has been studied on a variety of problems (Buzdalov and Doerr 2017; Doerr, Neumann, and Sutton 2017; Antipov, Doerr, and Karavaev 2019, 2020, 2022). Notably, Antipov, Doerr, and Karavaev (2022) proved that the $(1+(\lambda,\lambda))$ GA optimizes JUMP functions with gap size k, with optimal parameters (depending on k), in an expected runtime of order of $(n/k)^{k/2}e^{\Theta(k)}$, which is faster by a factor of $(nk)^{-k/2}e^{\Theta(k)}$ compared to the $(1+1)$ EA.

Other articles determined optimal parameter settings of the $(1+(\lambda,\lambda))$ GA and proposed dynamic choices (Doerr 2016; Doerr and Doerr 2018; Antipov, Buzdalov, and Doerr 2021; Hevia Fajardo and Sudholt 2022). Especially, Antipov, Buzdalov, and Doerr (2021) showed for JUMP functions with gap size k that choosing all parameters of the $(1+(\lambda,\lambda))$ GA according to heavy-tailed distributions (independent of k) achieves a performance close to the best instance-specific parameter choice.

Multi-objective optimization. Although it is not the main focus of this article, we mention that crossover has also been studied for multi-objective problems (Neumann and Theile 2010; Qian, Yu, and Zhou 2013; Huang et al. 2019). Recently, the most-widely used EA for multi-objective optimization, NSGA-II (Deb et al. 2002), which employs crossover, has been studied (Bian and Qian 2022; Dang et al. 2023; Doerr and Qu 2023).

Standard algorithm with standard parameters. To the best of our knowledge, the only result considering a single-objective standard GA with typical parameters is the work by Dang et al. (2018). For JUMP functions with gap size $k = o(n)$, the authors prove that this algorithm with population size $\mu \leq \kappa n$, with κ a sufficiently small constant, optimizes JUMP functions with gap size $k \geq 3$ in time $O(n^k/\mu + n^{k-1}\log n)$. When using a slightly higher mutation rate than normal, an expected runtime of $O(n^{k-1})$ can be achieved. The analysis tracks the diversity of the population, showing that there are phases lasting $\Omega(\mu^2)$ iterations in expectation in which a sufficient diversity in the population emerges naturally. During such a phase, the probability to create the optimum via crossover and mutation is larger by a factor of roughly n compared to using only mutation.

3 Preliminaries

We use the following **notation**. Let $\mathbb{N}$ denote the set of all natural numbers (including 0), and let $\mathbb{R}$ be the set of real numbers. For $m, n \in \mathbb{N}$, let $[m..n] = [m,n] \cap \mathbb{N}$, that is, the discrete interval from m to n. Further, we define the special case $[n] := [1..n]$.

Given a random variable X, a σ-algebra $\mathcal{F}$, and an event A with $\Pr[A] > 0$, let $\mathbb{E}[X \mid \mathcal{F}; A] = \mathbb{E}[X \cdot \mathbb{1}_A \mid \mathcal{F}]/\Pr[A]$. We make use of this notation when we condition on a σ-algebra but also make case distinctions via events.

We consider the **maximization of pseudo-Boolean functions** $f\colon \{0,1\}^n \to \mathbb{R}$. Conforming to the standards in the field of evolutionary computation, such a function will be called a *fitness function*, elements from $\{0,1\}^n$ are *individuals* and $f(x)$ is called their *fitness*. Throughout this paper, whenever we use big-O notation, we assume that the statement is asymptotic in the problem dimension n.

For two individuals $x, y \in \{0,1\}^n$, we define their *Hamming distance* as the number of positions that they differ in. Further, let $|x|_1$ denote the number of 1s in a bit-string x.

The **JUMP functions benchmark** was introduced by Droste, Jansen, and Wegener (2002) and is, together with some variations (Jansen 2015; Bambury, Bultel, and Doerr 2021; Doerr and Zheng 2021; Witt 2023), the most common benchmark to study how randomized search heuristics cope with local optima, see, e.g., Doerr et al. (2017); Corus, Oliveto, and Yazdani (2020); Doerr (2021); Rajabi and Witt (2022); Lissovoi, Oliveto, and Warwicker (2023); Bian et al. (2023); Zheng and Doerr (2024). For $n \in \mathbb{N}_{\geq 1}$ and $k \in [n]$, the JUMP *function* $\text{JUMP}_k\colon \{0,1\}^n \to \mathbb{N}$ is defined by

$$\text{JUMP}_k\colon x \mapsto \begin{cases} k + |x|_1 & \text{if } |x|_1 = n \text{ or } |x|_1 \leq n-k, \\ n - |x|_1 & \text{otherwise.} \end{cases} \tag{1}$$

Apparently, the function value of JUMP_k increases with the number of 1s in the individual until a *plateau* of local optima is reached (all points with exactly $n-k$ 1s). The only global optimum of JUMP_k is the all-1s string. It is separated from the plateau of local optima by a valley of low fitness (all individuals with more than $n-k$ but less than n 1s). Due to this structure, typical elitist EAs easily reach solutions on the plateau, but then have to change k particular bits in a single iteration.

As algorithm we consider the $(\mu+1)$ *genetic algorithm* ($(\mu+1)$ GA), a standard genetic algorithm in the community. See Algorithm 1 for the pseudocode. This algorithm has three parameters, the *population size* $\mu \in \mathbb{N}_{\geq 2}$, the *crossover rate* $p_c \in [0,1]$, and the *mutation rate* $p_m \in [0,1]$.

The $(\mu+1)$ GA maintains a multiset (*population*) of μ individuals, initialized with uniform samples from the search space and afterward updated iteratively. In each iteration, one new individual is created and potentially replaces an existing individual in the population. This is done in the following way, where all random choices are independent.

Algorithm 1: The $(\mu+1)$ GA with parameters $\mu \in \mathbb{N}_{\geq 2}$ and $p_c, p_m \in [0,1]$, maximizing fitness function f

$t \leftarrow 0$;
$P^{(0)} \leftarrow \mu$ individuals, uniformly at random (u.a.r.) from $\{0,1\}^n$;
while termination criterion met **do** `// iter. t`
  $p \leftarrow$ value from $[0,1)$ u.a.r.;
  if $p < p_c$ **then**
    $x^{(t)}, y^{(t)} \leftarrow$ two individuals from $P^{(t)}$ chosen u.a.r. (with replacement);
    $\widetilde{z}^{(t)} \leftarrow$ new individual created by uniform crossover of $x^{(t)}$ and $y^{(t)}$;
    $z^{(t)} \leftarrow \widetilde{z}^{(t)}$ augmented by standard bit mutation;
  else
    $m^{(t)} \leftarrow$ copy of an individual from $P^{(t)}$ chosen u.a.r.;
    $z^{(t)} \leftarrow m^{(t)}$ augmented by standard bit mutation;
  $\ell^{(t)} \leftarrow$ individual with the lowest fitness from $P^{(t)} \cup \{z^{(t)}\}$, breaking ties u.a.r.;
  $P^{(t+1)} \leftarrow (P^{(t)} \cup \{z^{(t)}\}) \setminus \{\ell^{(t)}\}$;
  $t \leftarrow t+1$;

With probability p_c, two individuals from the current population are selected uniformly at random and with repetition (the *parents*). Then, a new individual (the *offspring*) is created by *uniform crossover* of the parents. Here each bit of the offspring is chosen from the respective position of a random one of the two parents. Afterward, *standard bit mutation* with mutation rate p_m is applied to the offspring, that is, each of its bits is flipped independently with probability p_m.

Otherwise, that is, with probability $1-p_c$, a single parent is selected uniformly at random from the current population and produces an offspring only via standard bit mutation.

Finally, in either case, the new individual is added to the population and an individual with the lowest fitness is removed, breaking ties uniformly at random. The resulting multiset, again of μ individuals, forms the population for the next iteration.

As common in this field, we do not specify a termination criterion, but assume that the algorithm runs indefinitely and define its *runtime* on a given instance as the number of fitness function evaluations until an optimal solution is evaluated for the first time.

Tools for Our Analyses

The following elementary lemma computes the probability of constructing the optimum form a diverse population on the plateau of local optima. It is a minimal adaptation of (Dang et al. 2018, Lemma 2), from mutation rate $\frac{1}{n}$ to rate $\Theta(\frac{1}{n})$, so we omit the proof here.

Lemma 1. *Let $n \in \mathbb{N}_{\geq 1}$, $\chi = \Theta(1)$, let $k \in [n/2]$, let $d \in [0..k]$, and let $p_m = \frac{\chi}{n}$. Furthermore, let $x, y \in \{0,1\}^n$ with $|x|_1 = |y|_1 = n-k$ and such that their Hamming distance is $2d$. Then the probability that the result of uniform crossover of x and y, followed independently by standard bit mutation with mutation rate p_m, is the all-1s string is $\Omega\big(4^{-d}\big(\frac{n}{\chi}\big)^{-k+d}\big)$.*

Our proof of the result that the diversity of the population on the plateau persist for an exponential time needs the following negative drift theorem. It is a variant of (Kötzing 2016, Theorem 3) where the drift does not need to be negative for the entire search space, but only in suitable parts of it.

Theorem 2 ((Krejca 2019, Corollary 3.24))**.** *Let $(X_t)_{t\in\mathbb{N}}$ be a random process over $\mathbb{R}$ adapted to a filtration $(\mathcal{F}_t)_{\in\mathbb{N}}$. Further, let $X_0 \leq 0$, let $b \in \mathbb{R}_{>0}$, and let $T = \inf\{t \in \mathbb{N} \mid X(t) \geq b\}$. Suppose that there are constants $a \in \mathbb{R}_{\leq 0}$, $c \in (0,b)$, and $\varepsilon \in \mathbb{R}_{<0}$ such that, for all $t \in \mathbb{N}$, it holds that*

1. $\mathbb{E}[(X_{t+1}-X_t)\cdot \mathbf{1}_{X_t \geq a}\cdot \mathbf{1}_{T<t} \mid \mathcal{F}_t] \leq \varepsilon \cdot \mathbf{1}_{X_t\geq a}\cdot \mathbf{1}_{T<t}$,
2. $|X_{t+1}-X_t|\cdot \mathbf{1}_{X_t\geq a}\cdot \mathbf{1}_{T<t} < c\cdot \mathbf{1}_{X_t\geq a} + \mathbf{1}_{X_t<a}$, *and*
3. $X_{t+1}\cdot \mathbf{1}_{X_t<a}\cdot \mathbf{1}_{T<t} \leq 0$.

Then, for all $t \in \mathbb{N}$, it holds that $\Pr[T \leq t] \leq t^2 \exp\Big(-\frac{b|\varepsilon|}{2c^2}\Big)$.

4 Population Dynamics on the Plateau

In this section, we analyze the population dynamics of the $(\mu+1)$ GA once the population is fully contained on the plateau of local optima. This is the most interesting part of the run of the algorithm, both because it takes longest and because it is here where crossover makes a substantial difference. Due to space restrictions, we omit some proofs, which can be found in the full article (Doerr et al. 2023b).

General assumption: In all what follows, we regard how the $(\mu+1)$ GA optimizes the benchmark $\textsc{Jump}_k$. Our results are asymptotic in the problems size n, hence we can always assume that n is sufficiently large and we shall do so often without explicit notice. We allow k to depend on n. We recall that the most interesting case is that k is small, since otherwise the runtimes are exorbitant. For that reason, we shall occasionally assume (with explicit notice) that $k = o(n)$, as this will ease some proofs. For the algorithm parameters, we regard an arbitrary crossover rate $p_c \in [0,1]$ that is $\Omega(1)$, that is, at least ε for some absolute constant. We allow an arbitrary mutation rate $p_m = \frac{\chi}{n}$ with χ a positive constant. We also allow an arbitrary population size $\mu \geq 2$, and this parameter again may and usually will depend on n. Since our results in particular show that small population sizes suffice to profit from crossover, we often assume that $\mu = o(n)$.

Outline: To allow for a more formal analysis, we refer to identical individuals as a *species*. For a given species, we call an individual belonging to this species a *y-individual*, otherwise we say it is a *non-y-individual*. Since the population is on the plateau, individuals from different species have a Hamming distance of at least 2. Hence, for a constant $c \in (0,1)$, if the largest species has a size of at most $c\mu$

and due to the constant crossover rate, there is a constant probability to select two individuals for crossover that have a Hamming distance of at least 2. By Lemma 1, the probability to create the optimum via such a crossover followed by mutation is at least $\Omega((\frac{n}{\chi})^{-k+1})$, which is roughly n times more likely than if mutation by itself creates the optimum. In the following, we show that the largest species quickly reaches a size of at most $c\mu$ (Lemma 6) and remains there for a time exponential in μ (Lemma 7).

Details: Key to deriving strong bounds on the lasting diversity population is that we show that the majority species has a higher probability of decreasing its cardinality y than increasing it (Lemma 5). This difference in probability is proportional to $(\mu - y)/\mu$. That is, initially, it is not very well pronounced and only in the order of $1/\mu$, but with decreasing y, it becomes more likely, until it is constant. Once it is constant, it is very unlikely to reduce quickly.

To be more formal, for species s (where s is just a symbol), we view its size Y_s as a random process over $[0..\mu]$. In the following, we analyze the transition probabilities of this random process, which we define for all $y \in [\mu]$, all $t \in \mathbb{N}$, and all events A with non-zero probability as

$$p_+^{(s,t)}(y \mid A) \coloneqq \Pr[Y_s(t+1) - Y_s(t) = 1 | Y_s(t) = y, A], \tag{2}$$

$$p_-^{(s,t)}(y \mid A) \coloneqq \Pr[Y_s(t+1) - Y_s(t) = -1 | Y_s(t) = y, A].$$

When analyzing the transition probabilities above, we make a case distinction with respect to how the offspring in an iteration is created. For some of these cases, we use the bounds derived by Dang et al. (2018). However, we improve the following cases: Lemma 3 assumes that crossover (followed by mutation) is performed with parents whose Hamming distance is at most 2. Lemma 4 assumes that only mutation is performed.

The following bound incorporates the constant in the mutation rate more closely than the analysis of Dang et al. (2018).

Lemma 3. *Consider the $(\mu+1)$ GA with the assumptions take at the beginning of this section optimizing $\textsc{Jump}_k$ with $k = o(n)$. Consider a single iteration $t \in \mathbb{N}$. Let A denote the event that the entire population $P^{(t)}$ is on the plateau and that the offspring produced this iteration is created via crossover of two parents whose Hamming distance is at most 2 (followed by mutation). Assume that A has a positive probability of occurring. Last, let s be a species of $P^{(t)}$.*

Then, for the transition probabilities defined in equation (2) *and for all $y \in [\mu - 1]$, it holds that*

$$p_-^{(s,t)}(y \mid A) \geq \frac{y(\mu-y)\big(\mu\big(1+\frac{1}{2}\chi\big) + \frac{1}{2}y\chi\big)}{2(\mu+1)\mu^2}\Big(1-\frac{\chi}{n}\Big)^n.$$

The next lemma considers the transition probabilities of a species under the assumption that only mutation is applied. This case is not present in the original analysis of Dang et al. (2018), as they assume a crossover rate of 1, whereas we allow for constant values less than 1. It is essential to analyze these transition probabilities carefully, as they can occur with constant probability and thus have a great impact on the difference of the transition probabilities. Our bounds show that there is actually a slight bias towards increasing the size of the largest species. However, as this bias is in the order of $1/n$ whereas the probability of decreasing is at least in the order of $1/\mu$, the assumption $\mu = o(n)$ overall guarantees that the largest species is more likely to decrease.

Lemma 4. *Consider the $(\mu+1)$ GA with the assumptions take at the beginning of this section optimizing $\textsc{Jump}_k$ with $k = o(n)$. Consider a single iteration $t \in \mathbb{N}$. Let B denote the event that the entire population $P^{(t)}$ is on the plateau and that the offspring produced this iteration is created via mutation (and no crossover). Assume that B has a positive probability of occurring. Last, let s be a species of $P^{(t)}$.*

Then, for the transition probabilities defined in equation (2) *and for all $y \in [\mu]$, it holds that*

$$p_+^{(s,t)}(y \mid B) = \frac{y(\mu-y)}{\mu(\mu+1)}\Big(1-\frac{\chi}{n}\Big)^n + O\bigg(\frac{(\mu-y)^2}{n\mu^2}\bigg), \text{ and}$$

$$p_-^{(s,t)}(y \mid B) \geq \frac{y(\mu-y)}{\mu(\mu+1)}\Big(1-\frac{\chi}{n}\Big)^n.$$

Using the mainly the transition probabilities from the two lemmas above, we bound the crucial difference in the overall probabilities to increase or decrease a species.

Lemma 5. *Consider the $(\mu+1)$ GA with the assumptions taken at the beginning of this section and $\mu = o(n)$ optimizing $\textsc{Jump}_k$ with $k = o(n)$. Let T be a (random) iteration such that the entire population $P^{(T)}$ is on the plateau. Last, let s be a species of $P^{(t)}$.*

Then, for the transition probabilities defined in equation (2) *and for all $y \in [\frac{\mu}{2}..\mu - 1]$, it holds that*

$$p_-^{(s,T)}(y \mid P^{(T)}) - p_+^{(s,T)}(y \mid P^{(T)}) \geq \frac{\mu - y}{32e\mu} p_c.$$

Utilizing Lemma 5, the next lemma bounds the expected time for a species to reach a size of at most $\frac{\mu}{2}$. We improve the previous result by Dang et al. (2018, Lemma 6), which pessimistically assumed an unbiased random walk, resulting in a bound of $O(\mu n + \mu^2 \log(\mu))$. Our bound improves both terms almost by a factor of μ by carefully considering that the size of the species decreases more likely the smaller it is.

Lemma 6. *Consider the $(\mu+1)$ GA with the assumptions taken at the beginning of this section and $\mu = o(n)$ optimizing $\textsc{Jump}_k$ with $k = o(n)$. Further, let s be a species, and let $t^* \in \mathbb{N}$ be an iteration such $P^{(t^*)}$ is entirely on the plateau. In addition, let $Y_s(t^*)$ be defined as above equation* (2)*.*

Let $T = \inf\{t \in \mathbb{N} \mid Y_s(t^* + t) \leq \frac{\mu}{2}\}$. Then

$$\mathbb{E}[T \mid Y_s(t^*)] = O((\tfrac{n}{k} + \mu)\log(\mu)).$$

Last, the following lemma shows that once a species has a size of at most $\frac{\mu}{2}$, then, for any constant $\lambda \in (\frac{1}{2}, 1)$, its size does not reach $\lambda\mu$ with high probability in a number of iterations exponential in μ. This drastically increases the previously best known probability by Dang et al. (2018, Lemma 7), which is only quadratic in μ.

Lemma 7. *Consider the $(\mu+1)$ GA with the assumptions take at the beginning of this section and $\mu = o(n)$ optimizing $\textsc{Jump}_k$ with $k = o(n)$. Further, let s be a species,*

and let $t^* \in \mathbb{N}$ be an iteration such $P^{(t^*)}$ is entirely on the plateau. Assume that $Y_s(t^*)$ (as defined above equation (2)) is at least $\frac{\mu}{2}$. Let $\lambda \in (\frac{1}{2}, 1)$, and let $T = \inf\{t \in \mathbb{N} \mid Y_s(t + t^*) \geq \lambda\mu\}$. Last, let $C = \frac{(2\lambda-1)(1-\lambda)}{128e} p_c$. Then, for all $t \in \mathbb{N}$, it holds that $\Pr[T \leq t] \leq t^2 \cdot \exp(-C\mu)$.

5 Runtime Analysis

In this section, we prove our main result on the runtime of the $(\mu + 1)$ GA optimizing the JUMP_k benchmark.

Theorem 8. *Let $C = \frac{1}{1024e} p_c$. Under the general assumptions takes at the beginning of Section 4, the $(\mu + 1)$ GA with population size $\mu = o(n)$ optimizes the JUMP_k benchmark with $k = o(n)$ in an expected time (number of iterations or fitness evaluations) of*

$$O\Bigg(n(\mu \log(\mu) + \log(n)) + \frac{(n/k + \mu)\log(\mu)}{(n/\chi)^{-k+1} \min(\exp(\frac{C\mu}{2}), (n/\chi)^{k-1})} + \left(\frac{n}{\chi}\right)^{k-1}\Bigg).$$

To prove this result, we first show how the $(\mu + 1)$ GA reaches a population that is fully contained on the plateau of local optima and then apply our new understanding of the population dynamics. For this first part of the run, we show the following runtime estimate. Since in this part of the optimization we do not profit from crossover, we do not need the requirement that the crossover rate satisfies $p_c = \Omega(1)$.

Theorem 9. *Under the general assumptions taken in Section 4, but allowing an arbitrary crossover rate $p_c \in [0, 1]$, we consider the $(\mu + 1)$ GA optimizing the JUMP_k benchmark (with $k \leq n/2$). Then the expected time until either the optimum is found or the entire population is on the plateau is $O(n(\mu \log(\mu) + \log(n)))$.*

This result improves the time proven in (Dang et al. 2018, Lemma 1) by a factor of $\sqrt{k}$. This factor has no influence on the final runtime, as this is strongly dominated by the time to leave the plateau, but we found it useful to show that the time of this phase is independent of the problem parameter k, which is very natural. We also note that our proof, omitted here for reasons of space, while reusing several arguments of the previous work, avoids the use of the technical population fitness level argument (Dang et al. 2018, Theorem 1).

We are now ready to prove our main runtime result.

Proof sketch of Theorem 8. We split the analysis into two parts. The first part bounds the time it takes until the entire population of the $(\mu + 1)$ GA is on the plateau. By Theorem 9, this expected time is $O(n(\mu \log(\mu) + \log(n)))$.

For the second part, we consider, for appropriately chosen constants c_1 and c_2 and a value $t = \min(\exp(\frac{C\mu-1}{2}), \frac{4}{p_c c_2}(\frac{n}{\chi})^{k-1})$, phases of a fixed length $\ell = 2c_1((\frac{n}{k} + \mu)\log(\mu)) + t$. For each phase, we bound the probability from below that the $(\mu + 1)$ GA generates a sufficiently diverse population such that creating the optimal solution is more likely. For appropriately chosen values, by Lemma 6 and Markov's inequality, with probability at least $\frac{1}{2}$, the $(\mu + 1)$ GA creates a sufficiently diverse population in $2c_1((\frac{n}{k} + \mu)\log(\mu))$ iterations. By Lemma 7, for appropriate values, this diversity remains with constant probability for the remaining t iterations of a phase. Then, by Lemma 1, the $(\mu + 1)$ GA has a probability of $\Omega((\frac{n}{\chi})^{-k+1})$ to create the optimum in a single iteration. This results in an overall probability of at least $\Omega((\frac{n}{\chi})^{-k+1}\frac{t}{2})$ to create the optimum in a single phase. Calculating the expected number of phases yields the result. □

Discussion: In comparison to the results of Dang et al. (2018), our result (Theorem 8) holds for any mutation rate $\frac{\chi}{n}$, not just for $\chi \geq 1$, and for all $\mu = o(n)$. More importantly, we get a better understanding of how μ influences the overall expected run time, which was not possible before to this extent. For constant μ (and $k \geq 2$), we get a bound of $O(\frac{n^k}{\chi^{k-1}k})$, which shows that this term gets closer to the term $(\frac{n}{\chi})^{k-1}$ with increasing k. For $\mu = o(\log(n))$, we see how the dominating term $\frac{n^k}{\chi^{k-1}k}\frac{\log(\mu)}{\exp(\mu)}$ becomes drastically smaller with increasing μ. Once $\mu = \Omega(\log(n))$ and $k \geq 3$, the bound becomes $O((\frac{n}{\chi})^{k-1})$, which is better by a factor of χ^{-k+1} for $\chi > 1$ compared to the results of Dang et al. (2018). This is the regime with the largest speed-up compared to the $\Theta(n^k)$ runtime of the $(1 + 1)$ EA without crossover (Jansen and Wegener 2002).

6 Experimental Evaluation

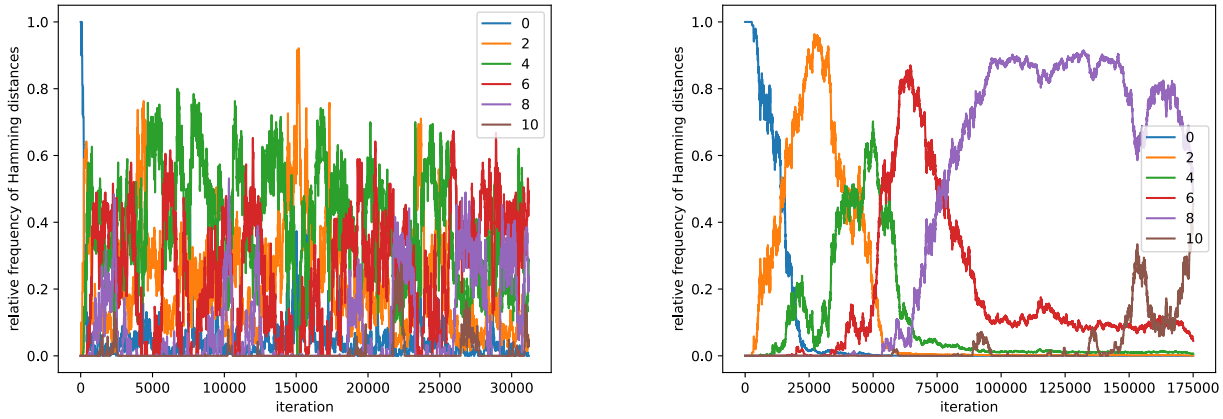
Our runtime analysis in Section 5, especially Lemma 7, shows that once the entire population of the $(\mu + 1)$ GA is on the plateau, there are phases of length exponential in μ during which the largest species (that is, identical individuals) is at most $\frac{3}{4}\mu$. During this time, it is possible that various species get produced. The larger the Hamming distance of two species, the easier it is to find the optimum of JUMP_k (Lemma 1), potentially improving the runtime. In fact, experiments by Dang et al. (2018, Fig. 6) suggest that the expected runtime of the $(\mu + 1)$ GA on JUMP_k is rather in the order of $n \ln(n) + 4^k$ for certain mutation rates.

Since the key to improving our result (Theorem 8) is to better understand how diverse the population on the plateau really is during phases of lasting diversity, we investigate these population dynamics empirically in this section. As Dang et al. (2018) present an extensive empirical analysis of the runtime of different algorithm variants on JUMP_k, we focus on a quantity they did not cover, namely, the relative frequencies of all pairwise Hamming distances in the population. A higher ratio of a larger Hamming distance implies a larger probability to create the optimum. Thus, these numbers provide strong insights into how well the population is spread out in order to create the optimum.

Our code is available on GitHub (Doerr et al. 2023a).

Frequencies of Pairwise Hamming Distances

We analyze and discuss how the relative frequencies of pairwise Hamming distances in the population of the $(\mu + 1)$ GA on the plateau of JUMP_k evolve over the algorithm's iterations. Figure 1 shows our results and explains



(a) A single run of the $(\mu + 1)$ GA on JUMP_k with the parameters $n = 100$, $k = 5$, $\mu = 20$, $p_c = 1$, and $p_m = \frac{1}{n}$.

(b) A single run of the $(\mu + 1)$ GA on JUMP_k with the parameters $n = 1000$, $k = 5$, $\mu = 200$, $p_c = 1$, and $p_m = \frac{1}{n}$.

Figure 1: The relative frequencies of the different Hamming distances of the population of the $(\mu + 1)$ GA (Algorithm 1) on the plateau of JUMP_k (equation (1)), for the respective parameter settings. Each plot depicts a single run of the $(\mu + 1)$ GA on JUMP_k starting with the entire population on the plateau. The initial population is chosen such that it consists of a single species (i.e., μ copies of the same individual), chosen uniformly at random among all individuals with exactly k 0s and $n - k$ 1s. The run is stopped once the algorithm creates the global optimum. The x-axis denotes the number of iterations of the algorithm. The different colors refer to the different Hamming distances in the population. The y-axis depicts the relative frequency of each such Hamming distance among all $\binom{\mu}{2}$ pairs of individuals. Since all individuals are on the plateau, the Hamming distances are always even numbers. The maximum number is twice the gap size, i.e., $2k$. Please refer to Section 6 for more information.

our experimental setup. We see both in Figures 1a and 1b that the different possible Hamming distances (from 0 to 10, in steps of 2) emerge in increasing order of their value (that is, first distance 0, then 2, and so on). This trend is qualitatively the same for both figures, with larger fluctuations for the smaller dimension size (Figures 1a).

Initially, by the setup of the experiment, all individuals have a Hamming distance of 0 to each other. However, after a short time (relative to the entire runtime), the frequency for distance 0 decreases rapidly and the one for distance 2 grows rapidly. From this point on, the frequency for distance 0 remains very low for the rest of the run. That is, the diversity in the population is very high (and remains high). This supports the findings by Dang et al. (2018, Fig. 2), whose experiments also show that the diversity in the population remains very high during a run.

A similar behavior occurs for other neighboring pairs of distances, such as 2 and 4, although this effect is far more pronounced in Figure 1b. After the frequency of a distance $d \in \{2, 4, 6\}$ grows, the frequency of distance $d + 2$ also grows until it overtakes the value of d. From this point on, the frequency of d typically remains below that of $d + 2$, going so far that d is almost not present anymore after some time (Figure 1b). This trend seems to also emerge for distances 8 and 10, although their frequencies barely meet, since the optimum is found by then. These dynamics suggest that the population does not only diversify in the number of species over time but also increases in pairwise distances.

Last, we see that the run ends in both cases once there is a sufficiently high fraction of individuals with maximum Hamming distance in the population. This conforms with the hypothesis that the actual expected runtime is in the order of $n \ln(n) + 4^k$.

7 Conclusion

In this work, we detected and mathematically proved that the diversity occurring when the $(\mu + 1)$ GA optimizes JUMP functions persists much longer than known before, namely for a time exponential in the population size μ instead of quadratic. This result allowed us to prove superior runtime guarantees, in particular, for small population sizes.

Our experiments support our finding that once the population is not anymore dominated by a single genotype, this diversity lasts for a long time. However, our experiments also show that the diversity produced by the $(\mu + 1)$ GA is even stronger than what our proofs show. Not only does no genotype class dominate, but also the typical Hamming distance between individuals grows. If such an effect could be proven, this would immediately lead to much stronger runtime guarantees. This is clearly the most interesting, possibly not very easy, continuation of this work.

From the viewpoint of designing effective genetic algorithms, our work suggest to use crossover with constant crossover rate, and without deviating from standard parameter suggestions such as using a mutation rate of $\frac{1}{n}$. In our results, the greatest speed-up is achieved for population sizes μ logarithmic in n.

Acknowledgments

This research benefited from the support of the FMJH Program PGMO. Some ideas of this paper were the result of discussions at Dagstuhl Seminar 22081—Theory of Randomized Optimization Heuristics.

References

Antipov, D.; Buzdalov, M.; and Doerr, B. 2021. Lazy parameter tuning and control: choosing all parameters randomly from a power-law distribution. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, 1115–1123. ACM.

Antipov, D.; Doerr, B.; and Karavaev, V. 2019. A tight runtime analysis for the $(1 + (\lambda, \lambda))$ GA on LeadingOnes. In *Foundations of Genetic Algorithms, FOGA 2019*, 169–182. ACM.

Antipov, D.; Doerr, B.; and Karavaev, V. 2020. The $(1 + (\lambda, \lambda))$ GA is even faster on multimodal problems. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, 1259–1267. ACM.

Antipov, D.; Doerr, B.; and Karavaev, V. 2022. A rigorous runtime analysis of the $(1 + (\lambda, \lambda))$ GA on Jump functions. *Algorithmica*, 84: 1573–1602.

Auger, A.; and Doerr, B., eds. 2011. *Theory of Randomized Search Heuristics*. World Scientific Publishing.

Bambury, H.; Bultel, A.; and Doerr, B. 2021. Generalized jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, 1124–1132. ACM.

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 *Parallel Problem Solving From Nature, PPSN 2022*, 428–441. Springer.

Bian, C.; Zhou, Y.; Li, M.; and Qian, C. 2023. Stochastic population update can provably be helpful in multi-objective evolutionary algorithms. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, 5513–5521. ijcai.org.

Buzdalov, M.; and Doerr, B. 2017. Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, 1343–1350. ACM.

Corus, D.; and Oliveto, P. S. 2018. Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Transactions on Evolutionary Compututation*, 22: 720–732.

Corus, D.; and Oliveto, P. S. 2020. On the benefits of populations for the exploitation speed of standard steady-state genetic algorithms. *Algorithmica*, 82: 3676–3706.

Corus, D.; Oliveto, P. S.; and Yazdani, D. 2020. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theoretical Computer Science*, 832: 166–185.

Dang, D.; Friedrich, T.; Kötzing, T.; Krejca, M. S.; Lehre, P. K.; Oliveto, P. S.; Sudholt, D.; and Sutton, A. M. 2016. Escaping local optima with diversity mechanisms and crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, 645–652. ACM.

Dang, D.; Friedrich, T.; Kötzing, T.; Krejca, M. S.; Lehre, P. K.; Oliveto, P. S.; Sudholt, D.; and Sutton, A. M. 2018. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22: 484–497.

Dang, D.-C.; Opris, A.; Salehi, B.; and Sudholt, D. 2023. A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In *Conference on Artificial Intelligence, AAAI 2023*, 12390–12398. AAAI Press.

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: 182–197.

Doerr, B. 2016. Optimal Parameter Settings for the $(1 + (\lambda, \lambda))$ Genetic Algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, 1107–1114. ACM.

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

Doerr, B.; and Doerr, C. 2018. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80: 1658–1709.

Doerr, B.; Doerr, C.; and Ebel, F. 2013. Lessons from the black-box: fast crossover-based genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2013*, 781–788. ACM.

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.; Echarghaoui, A.; Jamal, M.; and Krejca, M. S. 2023a. Code repository of this paper. https://github.com/TheMor/ga-jump-diverse-populations.

Doerr, B.; Echarghaoui, A.; Jamal, M.; and Krejca, M. S. 2023b. Lasting Diversity and Superior Runtime Guarantees for the (μ+1) Genetic Algorithm. *CoRR*, abs/2302.12570.

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

Doerr, B.; Johannsen, D.; Kötzing, T.; Neumann, F.; and Theile, M. 2013. More effective crossover operators for the all-pairs shortest path problem. *Theoretical Computer Science*, 471: 12–26.

Doerr, B.; Le, H. P.; Makhmara, R.; and Nguyen, T. D. 2017. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, 777–784. ACM.

Doerr, B.; and Neumann, F., eds. 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.

Doerr, B.; Neumann, F.; and Sutton, A. M. 2017. Time complexity analysis of evolutionary algorithms on random satisfiable k-CNF formulas. *Algorithmica*, 78: 561–586.

Doerr, B.; and Qu, Z. 2023. From understanding the population dynamics of the NSGA-II to the first proven lower

bounds. In *Conference on Artificial Intelligence, AAAI 2023*, 12408–12416. AAAI Press.

Doerr, B.; and Theile, M. 2009. Improved analysis methods for crossover-based algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2009*, 247–254. ACM.

Doerr, B.; and Zheng, W. 2021. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*, 12293–12301. AAAI Press.

Droste, S.; Jansen, T.; and Wegener, I. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276: 51–81.

Hevia Fajardo, M. A.; and Sudholt, D. 2022. Theoretical and empirical analysis of parameter control mechanisms in the $(1 + (\lambda, \lambda))$ genetic algorithm. *ACM Transactions on Evolutionary Learning and Optimization*, 2: 13:1–13:39.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Huang, Z.; Zhou, Y.; Chen, Z.; and He, X. 2019. Running time analysis of MOEA/D with crossover on discrete optimization problem. In *Conference on Artificial Intelligence, AAAI 2019*, 2296–2303. AAAI Press.

Jansen, T. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.

Jansen, T. 2015. On the black-box complexity of example functions: the real jump function. In *Foundations of Genetic Algorithms, FOGA 2015*, 16–24. ACM.

Jansen, T.; and Wegener, I. 2002. The analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica*, 34: 47–66.

Kötzing, T. 2016. Concentration of first hitting times under additive drift. *Algorithmica*, 75: 490–506.

Kötzing, T.; Sudholt, D.; and Theile, M. 2011. How crossover helps in pseudo-Boolean optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2011*, 989–996. ACM.

Krejca, M. S. 2019. *Theoretical Analyses of Univariate Estimation-of-Distribution Algorithms*. Ph.D. thesis, Universität Potsdam.

Lehre, P. K.; and Yao, X. 2011. Crossover can be constructive when computing unique input-output sequences. *Soft Computing*, 15: 1675–1687.

Lissovoi, A.; Oliveto, P. S.; and Warwicker, J. A. 2023. When move acceptance selection hyper-heuristics outperform Metropolis and elitist evolutionary algorithms and when not. *Artificial Intelligence*, 314: 103804.

Mitchell, M.; Holland, J. H.; and Forrest, S. 1993. When will a genetic algorithm outperform hill climbing. In *Advances in Neural Information Processing Systems, NIPS 1993*, 51–58. Morgan Kaufmann.

Neumann, F.; Oliveto, P. S.; Rudolph, G.; and Sudholt, D. 2011. On the effectiveness of crossover for migration in parallel evolutionary algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2011*, 1587–1594. ACM.

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

Neumann, F.; and Witt, C. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.

Oliveto, P. S.; He, J.; and Yao, X. 2008. Analysis of population-based evolutionary algorithms for the vertex cover problem. In *Congress on Evolutionary Computation, CEC 2008*, 1563–1570. IEEE.

Oliveto, P. S.; Sudholt, D.; and Witt, C. 2022. Tight bounds on the expected runtime of a standard steady state genetic algorithm. *Algorithmica*, 84: 1603–1658.

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

Rajabi, A.; and Witt, C. 2022. Self-adjusting evolutionary algorithms for multimodal optimization. *Algorithmica*, 84: 1694–1723.

Sudholt, D. 2012. Crossover speeds up building-block assembly. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, 689–702. ACM.

Sutton, A. M. 2021. Fixed-parameter tractability of crossover: steady-state GAs on the closest string problem. *Algorithmica*, 83: 1138–1163.

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

Zheng, W.; and Doerr, B. 2024. Runtime analysis of the SMS-EMOA for many-objective optimization. In *Conference on Artificial Intelligence, AAAI 2024*. AAAI Press.

Zhou, Z.-H.; Yu, Y.; and Qian, C. 2019. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer.