Comma Selection Outperforms Plus Selection on OneMax with Randomly Planted Optima

Comma Selection Outperforms Plus Selection on OneMax with Randomly Planted Optima[†]

Joost Jorritsma
Eindhoven University of Technology
Eindhoven, Netherlands

Johannes Lengler
ETH Zürich
Zürich, Switzerland

Dirk Sudholt
University of Passau
Passau, Germany

ABSTRACT

It is an ongoing debate whether and how comma selection in evolutionary algorithms helps to escape local optima. We propose a new benchmark function to investigate the benefits of comma selection: OneMax with randomly planted local optima, generated by frozen noise. We show that comma selection (the $(1,\lambda)$ EA) is faster than plus selection (the $(1+\lambda)$ EA) on this benchmark, in a fixed-target scenario, and for offspring population sizes λ for which both algorithms behave differently. For certain parameters, the $(1,\lambda)$ EA finds the target in $\Theta(n \ln n)$ evaluations, with high probability (w.h.p.), while the $(1+\lambda)$ EA w.h.p. requires almost $\Theta((n \ln n)^2)$ evaluations.

We further show that the advantage of comma selection is not arbitrarily large: w.h.p. comma selection outperforms plus selection at most by a factor of $O(n \ln n)$ for most reasonable parameter choices. We develop novel methods for analysing frozen noise and give powerful and general fixed-target results with tail bounds that are of independent interest.

ACM Reference Format:
Joost Jorritsma, Johannes Lengler, and Dirk Sudholt. 2023. Comma Selection Outperforms Plus Selection on OneMax with Randomly Planted Optima[†]. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590488

1 INTRODUCTION

Evolutionary Algorithms (EAs) are optimisation heuristics that are very flexible. An important aspect of EAs are their selection strategies. *Elitist* strategies like plus selection always maintain the best-so-far search point in the population.[1] While elitism is helpful to exploit the best-so-far solution, a common concern is that the algorithm might get stuck in local optima. The *escape hypothesis* posits that non-elitism might help in such cases [6]. Indeed, the disadvantage of elitism can be measured by the *elitist black-box complexity* of a problem, and some (artificial) problems show an exponential penalty of elitism [6, 7, 14].

A popular non-elitist selection strategy is *comma selection*, in which the parent(s) are not allowed to compete for survival. Despite its popularity, it is still unclear how good comma selection is at helping with escaping local optima. Some deceptive landscapes, in which other non-elitist mechanisms can help, can still deceive comma selection [7]. A working group at a 2022 Dagstuhl seminar "came to the conclusion that there is a gap between theory and practice as we are lacking convincing examples (apart from Cliff) where comma selection provably helps, whereas in practice comma strategies seem to be quite popular to escape from local optima" [2].

The main theoretical results for comma strategies at local optima are for the benchmarks Jump and Cliff. It is known that comma selection in the (μ,λ) EA does not work better than plus selection on the Jump function, for arbitrary population sizes μ and λ [9]. For the Cliff function, there is a choice of λ such that the $(1,\lambda)$ EA "only" takes time $O(n^{3.97\cdots})$, while the plus strategy needs exponential time for any λ [19, 22, 34]. However, the optimisation time of the $(1,\lambda)$ EA is still rather high (albeit polynomial), and the dependence on λ is rather tricky. There are some promising efforts to develop self-adjusting mechanisms that can adapt λ during the run of the algorithm, but these come with their own pitfalls [19, 21, 25, 26]. In particular, the optimisation time for Cliff can be reduced to $O(n \ln n)$ by a self-adjusting mechanism which resets λ periodically, but this mechanism needs to be well-aligned with the problem [19].

Arguably, Cliff (and also Jump) captures a rather specific situation that might be atypical for local optima. We will not give the definition of Cliff, but only describe the atypical situation. When the algorithm is in a local optimum x, and accepts a Hamming neighbour y that is closer to the optimum ("down the cliff"), then *most neighbours of y have the same fitness as x* and thus are "back up on the cliff". In particular, a random walk without selective pressure would likely lead back to the local optimum (or an equivalent search point). It is rather hard to imagine this situation in practice, where at or near an optimum (global or local), random walks typically increase the distance from the optimum and decrease fitness. This is not just a coincidental feature of the Cliff function; the analysis of this phenomenon is at the heart of all runtime analyses of Cliff. Thus, Cliff (and Jump) might not be the best test function for understanding local optima.

One issue with the $(1,\lambda)$ EA is that one needs to find a compromise between two trends: if λ is too large, then every generation contains a clone of the parent, so the $(1,\lambda)$ EA just imitates an elitist algorithm. But if λ is too small then the algorithm can not cope with situations in which improvements are hard to find. If the goal is to find a global optimum, then it is hard to balance those two aspects. However, it is less problematic in fixed-target optimisation, which is why we study this setting. We will discuss this point in more detail in Section 1.3.

[†]Extended abstract. Full proofs can be found at [24].
[1]Some authors define elitism in a stronger way, so that the whole population must consist of the best-so-far search points [14].

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590488

1.1 Distorted OneMax

We introduce an alternative model of local optima, which we call DISTORTED ONEMAX or DISOM = $\text{DISOM}_{d,p}$ for short. It could also be called "ONEMAX with planted local optima" or "ONEMAX with frozen Bernoulli noise". We start with the ONEMAX function and two real-valued parameters $p \in [0,1]$ and $d > 0$. Then for each search point x, with probability p we increase its fitness by d, independently of the other search points. Hence, we artificially "plant" a local optimum in x. (It does not always need to be a local optimum since a fitter neighbour of x could also be distorted.) Note that the distortion is part of the fitness function, which is static. Hence, if an algorithm evaluates the same search point x several times, it will always detect the same fitness $f(x)$. This is different from models with noisy fitness evaluations, in which several queries for the same search point can give different fitness values. In independent, concurrent work [15] another frozen noise model was recently studied for the compact Genetic Algorithm cGA on ONEMAX. There, Gaussian noise was added to all search points.

1.2 Main Results

We study the fixed-target performance with fitness target $n - k^*$ of the $(1,\lambda)$ EA and the $(1+\lambda)$ EA on distorted ONEMAX with parameters $p \in [0,1]$ and $d > 1$. We will explain those choices in more detail in Section 1.3 below. We denote by $T^{\text{com}} = T^{\text{com}}_{k^*,\lambda,d,p}$ and $T^{\text{plus}} = T^{\text{plus}}_{k^*,\lambda,d,p}$ the number of function evaluations until the $(1,\lambda)$ EA and the $(1+\lambda)$ EA find a search point of fitness at least $n - k^*$ on $\text{DISOM}_{d,p}$, respectively.

We give matching upper and lower bounds on T^{com} and T^{plus} in Theorem 1.1 below for a wide range of parameters k^*, λ, d, p, which hold with high probability.[2] For those parameters T^{plus} is by a factor $1/p$ larger than T^{com}. In particular, we show that there are parameters for which comma selection reduces the runtime from nearly quadratic to quasi-linear. Since the assumptions on the parameters are a bit technical, we state them together with a discussion in Assumption 1 in Section 1.3 below. Intuitively, the parameters λ and k^* must be chosen such that the $(1,\lambda)$ EA efficiently reaches fitness target $n - k^*$, but it does not create a clone of the parent in each generation. We state our main theorem.

THEOREM 1.1. *Under Assumption 1 on the parameters below, with high probability*

$$T^{\text{com}}_{k^*,\lambda,d,p} = \Theta\big(n \ln n\big), \tag{1}$$

$$T^{\text{plus}}_{k^*,\lambda,d,p} = \Theta(n \ln(n)/p). \tag{2}$$

For any p such that $p = \omega(1/(n \ln n))$ and $p = n^{-\Omega(1)}$ there are k^*, λ and d such that Assumption 1 is satisfied. Thus, (1) *and* (2) *may differ by a factor of almost $n \ln n$.*

It may seem like an unimportant quirk that we have used a w.h.p. statement instead of expectations, but this is not so. Indeed, the DISTORTED ONEMAX function easily leads to regimes in which expectations are completely meaningless, since they are dominated by events of tiny probability which contribute gigantic terms to the expectation. This is a known phenomenon, see [14] for an in-depth discussion in the context of elitist black-box complexity. In our situation, we give the following proposition as example. We remark that those parameters are outside of the regimes of Assumption 1.

PROPOSITION 1.2. *For $p = 2^{-n}$, $d = n - 0.5$, $k^* = 0$ and $\lambda = 3 \ln n$,*

$$\mathbb{E}\big[T^{\text{com}}_{k^*,\lambda,d,p}\big] = O(n \ln n) \quad \text{and} \quad \mathbb{E}\big[T^{\text{plus}}_{k^*,\lambda,d,p}\big] = n^{\Omega(n)}, \tag{3}$$

but with high probability

$$T^{\text{com}}_{k^*,\lambda,d,p} = O(n \ln n) \quad \text{and} \quad T^{\text{plus}}_{k^*,\lambda,d,p} = O(n \ln n). \tag{4}$$

The proof of Proposition 1.2 is omitted. Following the terminology of [14] for black-box complexity, we also call the expected times in (3) the *Las Vegas runtimes*, and for $r \in [0,1]$ we call the *r-Monte Carlo runtime* the time until the algorithm finds the target with probability at least $1 - r$. Proposition 1.2 states that the Las Vegas runtime and the r-Monte Carlo runtime (for any constant r) differ dramatically for the $(1+\lambda)$ EA. In general, Monte Carlo runtimes are more informative since they make a statement about *typical* outcomes.[3]

Theorem 1.1 gives a factor $1/p$ between the comma and the plus strategy that is arbitrarily close to $n \ln n$. The next theorem shows that this is the largest possible factor for Monte Carlo runtimes if the $(1,\lambda)$ EA is efficient on DISOM. Note that the factor for Las Vegas runtimes can be huge by Proposition 1.2.

THEOREM 1.3. *Let $C, \varepsilon > 0$ be constants. Assume that the parameters $\lambda \geq 1$, $k^* \in [n^\varepsilon, n/6]$, $p \in [0,1]$ and $T \in [1, n^C]$ (all possibly depending on n) are such that with high probability*

$$T^{\text{com}}_{k^*,\lambda,d,p} \leq T. \tag{5}$$

Then there exists a constant $C' > 0$ such that with high probability

$$T^{\text{plus}}_{k^*,\lambda,d,p} \leq \begin{cases} T, & \text{if } p \cdot T = o(1) \text{ or } \lambda > C' \ln(n), \\ O\big(n(\lambda + \ln(n/k^*))\big), & \text{if } p \cdot n(\lambda + \ln(n/k^*)) = o(1), \\ O\big(T/p\big), & \text{otherwise, provided } \lambda = O(\ln(n/k^*)). \end{cases}$$

Moreover, in all three cases $T^{\text{plus}} = O(T \cdot n \ln n)$ w.h.p.

We believe that the condition on λ in the third case is not needed, and hope to remove this condition in future work. We emphasize that the conditions in Theorem 1.3 are much more general than Assumption 1 and also cover many degenerate parameter settings. For example, for small λ the $(1,\lambda)$ EA may be inefficient and not have runtime $\Theta(n \ln n)$. We caution that Theorem 1.3 is far from trivial. We do make heavy use of the condition $k^* \geq n^\varepsilon$ in the proof. Moreover, we believe that the statement would be wrong if the $(1+\lambda)$ EA would break fitness ties in favour of the parent. We discuss those matters in more detail in Section 6.

[2] *With high probability* (w.h.p.) means with probability $1 - o(1)$ as $n \to \infty$.

[3] Usually, there is another reason to prefer Monte Carlo runtimes, since with good Monte-Carlo runtimes we can restart the algorithm if a run gets stuck. However, the situation here is a bit more subtle, since DISOM is a randomized function. Thus, there are two forms of randomness: one from the random choice of the fitness function, and one from the random decision of the algorithms. If the long expected runtime comes from an atypical fitness function, the problem is not solved by restarting the algorithm. However, it is easy to find a *fixed* function for which Proposition 1.2 still holds, for example the ONEMAX function with a single planted local optimum of value $n - 1$ at the all-zero string. This is implicitly shown in the proof of Proposition 1.2.

1.3 Parameter Setup

We will now explain which regimes are reasonable to consider for the parameters k^*, λ, d, p. Note that all of $d = d(n)$, $k^* = k^*(n)$, $p = p(n)$ and $\lambda = \lambda(n)$ may depend on n.

We will use the abbreviation $\eta := e/(e-1)$. This is helpful since the probability that a mutation is not identical to the parent (is not a *clone*) is $\approx 1 - 1/e = \eta^{-1}$. We write $q := \eta^{-\lambda}$ for the (approximate) probability to have no clone in the offspring population. This is a central quantity, since it is the probability of escaping from a local optimum in the $(1,\lambda)$ EA. It has been known for decades that if $\lambda \geq C \ln n$ for a large C, then the $(1,\lambda)$ EA mimics an elitist algorithm because then $q = n^{-C\cdot\ln(\eta)} \approx n^{-0.66C}$ [22, 29]. For example, if $C \geq 4$ then $q = o(n^{-2})$ and w.h.p. the parent will be cloned in all of the first n^2 generations. Hence, the $(1,\lambda)$ EA just behaves as the $(1+\lambda)$ EA in this regime. Since we are interested in potential differences between those two algorithms, we will thus consider regimes where $q \geq n^{-1+\varepsilon}$, or equivalently $\lambda \leq (1-\varepsilon)\log_\eta n$, since this is the regime where the two algorithms behave differently [29]. Note that this choice is not wise when the algorithm is supposed to find the optimum, since the $(1,\lambda)$ EA with $\lambda \leq (1-\varepsilon)\log_\eta n$ is inefficient in finding the optimum of any function with unique optimum [34]. However, such a λ is fine for fixed-target optimisation, i.e., if we are interested in finding a search point of fitness at least $n - k^*$, as long as $\lambda \geq (1+\varepsilon)\log_\eta(n/k^*)$ [1], or equivalently $q \leq (k^*/n)^{1+\varepsilon}$. Thus, we require $(1+\varepsilon)\log_\eta(n/k^*) \leq \lambda \leq (1-\varepsilon)\log_\eta n$ for some constant $\varepsilon > 0$. This range is non-empty if $k^* = n^{\Omega(1)}$, so we will make this restriction. This also avoids some complications at the optimum, since the optimum of DISOM may be not at $\vec{1} = (1,\dots,1)$, but at a distorted point whose fitness exceeds n. For this reason we will assume $d \leq k^*$, so that the target fitness $n - k^*$ cannot be achieved by search points at distance larger than $2k^*$ from $\vec{1}$.

Furthermore, the $(1+\lambda)$ EA with $\lambda = O(\ln n)$ needs $O(n \ln n)$ fitness evaluations to optimize ONEMAX. If $p = o(1/(n \ln n))$, then w.h.p. the $(1+\lambda)$ EA will not encounter any distorted search points before finding the optimum. Thus, we may ignore the case $p = o(1/(n \ln n))$. We will restrict ourselves a little bit more, and assume $p = \omega(1/n)$. Finally, we will make two more assumptions for technical simplicity. Firstly, we require $p = o(k^*/n)$. This ensures that close to the target fitness $n-k^*$, the probability $\Theta(k^*/n)$ that an offspring is closer to $(1,\dots,1)$ dominates the probability p that the offspring is distorted. The regime $p = \omega(k^*/n)$ is rather different, and we leave the study of this regime for future work. Secondly, we assume that $q = \omega(p\lambda)$. Note that $p\lambda$ is roughly the probability of sampling a distorted offspring in one generation, while q is the probability of escaping a local optimum in the $(1,\lambda)$ EA. Thus, we assume that escaping is to be more likely than sampling another local optimum. This condition simplifies the analysis in some places, but we don't believe it is actually needed, and we hope that we can remove it in future work. In fact, we will require the slightly stronger condition $q \geq p^{1-\varepsilon}$, or equivalently $\lambda \leq (1-\varepsilon)\log_\eta(1/p)$. This is stronger than $q = \omega(p\lambda)$ since the other conditions already imply $p = n^{-\Omega(1)}$ and $\lambda = \Theta(\ln n)$. Summarizing, we will make the following assumption for our main theorem below.

ASSUMPTION 1. *Let $q := \eta^{-\lambda}$ for $\eta := e/(e-1)$, and let $\varepsilon > 0$ be any constant. We assume $k^* = n^{\Omega(1)}$ and $k^* = n^{1-\Omega(1)}$, $p = \omega(1/(n \ln n))$, and*

$$p^{1-\varepsilon} \leq q \leq (k^*/n)^{1+\varepsilon}. \tag{6}$$

Finally, we assume that $d \in [(1+\varepsilon)\ln(n/p)/\ln(n/k^*), k^*]$.

We have already motivated the assumptions on p, k^* and λ, and the upper bound on d. The lower bound on d will come out of the proof of the lower bound on T^{plus}. By the assumptions on k^* and p we have $\ln(n/p) = \Theta(\ln n)$ and $\ln(n/k^*) = \Theta(\ln n)$, so the lower bound on d is just a constant.

Note that we can write (6) equivalently as a condition on λ:

$$(1+\varepsilon)\log_\eta(n/k^*) \leq \lambda \leq (1-\varepsilon)\log_\eta(1/p). \tag{7}$$

Also, Assumption 1 implies $q = n^{-\Theta(1)}$, and thus for some $\delta > 0$,

$$p \leq q^{1+\varepsilon/(1-\varepsilon)} \leq qn^{-\delta} \leq n^{-2\delta}, \quad \text{and} \quad p \leq q \leq k^*/n^{1+\delta}. \tag{8}$$

We discuss briefly the possible ranges of the parameters. The values of p, q, and k^*/n are coupled by (6), but p can be arbitrarily close to $1/(n \ln n)$ and k^* can take a value n^c for a constant $c < 1$ that is arbitrarily close to 1. Hence, for any constant $0 < c < 1$ we may set any one of the three values p, q, or k^*/n to n^{-c} and still satisfy Assumption 1 by choosing the other two values appropriately. As discussed above, values of p or q much smaller than n^{-1} do not lead to interesting regimes, because respectively the algorithm does not encounter distorted points or it mimics a plus strategy. The restrictions on q always determine λ up to constant factors, where the interval may be more or less narrow depending on p and k^*. This discussion also implies the second statement of Theorem 1.1, that we may choose parameters yielding a factor of more than n between T^{com} and T^{plus}.

PARAMETERS YIELDING QUASI-LINEAR FACTOR IN THEOREM 1.1. Let $\delta > 0$ and $p = p(n)$ such that $p \leq n^{-\delta}$ and $p = \omega(1/(n \ln n))$. Then Assumption 1 is satisfied by setting $q := n^{-\delta/2}$, $k^* := n^{1-\delta/4}$, and d as a sufficiently large constant. □

2 NOTATION AND PRELIMINARIES

General Notation. We write $[n] := \{1,\dots,n\}$. We denote search points by $x = (x_1,\dots,x_n) \in \{0,1\}^n$, and the ONEMAX value of x is $\text{OM}(x) := \sum_{i\in[n]} x_i$. We denote by $\vec{1} = (1,\dots,1)$ and $\vec{0} = (0,\dots,0)$ the unique search points with $\text{OM}(\vec{1}) = n$ and $\text{OM}(\vec{0}) = 0$. For $x, y \in \{0,1\}^n$, the *Hamming distance* $H(x,y)$ of x and y is the number of positions $i \in [n]$ such that $x_i \neq y_i$. We call y a (Hamming) *neighbour* of x if $H(x,y) = 1$. We set $\eta := e/(e-1)$, and we denote by $\log_2 n$ the binary logarithm of n, by $\ln n$ the natural logarithm of n, and by $\log_\eta n := \ln n/\ln\eta$ the logarithm with base η.

For an event $\mathcal{E}$ we denote by $\mathbb{1}\{\mathcal{E}\}$ the indicator variable of $\mathcal{E}$, i.e., $\mathbb{1}\{\mathcal{E}\} = 1$ if $\mathcal{E}$ occurs and $\mathbb{1}\{\mathcal{E}\} = 0$ otherwise.

DISTORTED ONEMAX. We start with a formal definition of the DISTORTED ONEMAX function $\text{DISOM} : \{0,1\}^n \to \mathbb{R}_{\geq 0}$. We partition the search space $\{0,1\}^n$ into two sets $\mathcal{C}$ and $\mathcal{D}$ of "clean" and "distorted" points, respectively, where for each $x \in \{0,1\}^n$ we have

$$x \in \mathcal{D} \text{ with probability } p, \qquad x \in \mathcal{C} \text{ otherwise}, \tag{9}$$

independently of the other points. We define the DISTORTED ONEMAX function $\text{DISOM} = \text{DISOM}_{d,p}$ as

$$\text{DISOM}(x) := \text{OM}(x) + d \cdot \mathbb{1}\{x \in \mathcal{D}\}. \tag{10}$$

Algorithm 1: $(1,\lambda)$ EA for maximizing f to target $n-k^*$.

Initialization: $t = 0$; pick $x^{(0)}$ uniformly at random from $\{0,1\}^n$.
Optimization: **while** $f(x^{(t)}) \le n-k^*$ **do**
  Mutation: **for** $j \in \{1,\dots,\lambda\}$ **do**
    $y^{(t,j)} \leftarrow \text{mutate}(x^{(t)})$ by flipping each bit of $x^{(t)}$ independently with prob. $1/n$;
  Selection: Let $y^{(t)} = \arg\max\{f(y^{(t,1)}),\dots,f(y^{(t,\lambda)})\}$, breaking ties uniformly at random;
  Update: $x^{(t+1)} = y^{(t)}$; $t \leftarrow t+1$;

Algorithm 2: $(1+\lambda)$ EA for maximizing f to target $n-k^*$.

Initialization: $t = 0$; pick $x^{(0)}$ uniformly at random from $\{0,1\}^n$.
Optimization: **while** $f(x^{(t)}) \le n-k^*$ **do**
  Mutation: **for** $j \in \{1,\dots,\lambda\}$ **do**
    $y^{(t,j)} \leftarrow \text{mutate}(x^{(t)})$ by flipping each bit of $x^{(t)}$ independently with prob. $1/n$;
  Selection: Let $y^{(t)} = \arg\max\{f(y^{(t,1)}),\dots,f(y^{(t,\lambda)})\}$, breaking ties uniformly at random;
  Update: **if** $f(y^{(t)}) \ge f(x^{(t)})$ **then** $x^{(t+1)} = y^{(t)}$; **else** $x^{(t+1)} = x^{(t)}$; $t \leftarrow t+1$;

Algorithms. In Algorithms 1 and 2 we give the pseudocode for fixed-target optimisation of a fitness function f with the $(1,\lambda)$ EA and the $(1+\lambda)$ EA respectively. The *running time* is the number of function evaluations until the target is met. Note that in our context of DISOM, the target will always be to reach fitness at least $n-k^*$. Throughout the paper, we assume that the mutation rate is $1/n$.

We recall that $T^{\text{com}} = T^{\text{com}}_{k^*,\lambda,d,p}$ and $T^{\text{plus}} = T^{\text{plus}}_{k^*,\lambda,d,p}$ are the number of function evaluations until the $(1,\lambda)$ EA and the $(1+\lambda)$ EA respectively find a search point of fitness at least $n-k^*$ on $\text{DISOM}_{d,p}$. By the *time* that an algorithm spends, we refer to the number of function evaluations. So when the algorithm runs for time T, then it runs for $\lceil T/\lambda \rceil$ generations.

3 GENERAL TOOLS

In this section we collect some lemmas which we need for our results, but which may be useful in other contexts as well. Section 3.1 collects basics about the $(1,\lambda)$ EA. In Section 3.2, we prove that the $(1+\lambda)$ EA is the most efficient algorithm on ONEMAX (in the sense of stochastic domination) among all $(1+\lambda)$ algorithms with the same fixed mutation rate. This also holds in fixed-target settings. As a corollary, we obtain that the $(1+\lambda)$ EA on ONEMAX is the fastest algorithm to reach a fixed Hamming distance from the optimum among all $(1+\lambda)$ algorithms and all fitness function with a unique global optimum. In Section 3.3 we show high-probability upper and lower runtime bounds for the $(1+\lambda)$ EA and $(1,\lambda)$ EA on ONEMAX with prescribed start and target fitness. The results in this section will not come as a big surprise for experts since similar, but more specific, statements were known before. However, this is the first time they are proven in such generality.

3.1 Properties of the $(1,\lambda)$ EA

The following two lemmas are standard. They confirm that the probability of not creating a clone is $\approx q$, and that it is unlikely to have any mutation with more than $c\ln n$ bit-flips within the first n^2 generations. We remark that the following lemma is numbered 3.2 to maintain consistency with the full version [24].

LEMMA 3.2. *Let $\lambda = o(n)$ and $q = \eta^{-\lambda}$. Then the probability of an iteration of the $(1,\lambda)$ EA and the $(1+\lambda)$ EA of not creating a clone of the current search point is $q(1+o(1))$.*

LEMMA 3.3. *For every constant $c > 0$, the probability of an offspring having a Hamming distance of at least $c\ln(n)$ to its parent is $n^{-\Omega(\ln\ln n)}$. Hence, w.h.p. each offspring generated during the first $n^2\ln(n)$ evaluations in $(1,\lambda)$ EA or $(1+\lambda)$ EA has Hamming distance at most $c\ln(n)$ from its parent.*

3.2 Domination Results

We continue with domination results. It was first shown in [12] that ONEMAX is the easiest function for the $(1+1)$ EA, and these results were extended later in [8, 10, 35, 36]. Here we show that ONEMAX is also the easiest function for the $(1+\lambda)$ EA, and that conversely the $(1+\lambda)$ EA is the fastest mutation-based algorithm on ONEMAX which creates solutions in batches, see Theorem 3.5. Both also hold in fixed-target settings. This result has very powerful implications, and we first give three immediate consequences in Theorem 3.4. We say that a random variable Y *stochastically dominates* by a random variable X if $\mathbb{P}(Y \ge s) \ge \mathbb{P}(X \ge s)$ for all $s \in \mathbb{R}$.

THEOREM 3.4. *Let $a, b \in [0,n]$, $\lambda \in \mathbb{N}$, and consider the $(1+1)$ EA, the $(1,\lambda)$ EA and the $(1+\lambda)$ EA with the same mutation rate $r \le 1/2$ and with starting points x^{one}, x^{com} and x^{plus} respectively. Let $f : \{0,1\}^n \to \mathbb{R}$ be any fitness function.*

(a) Assume $OM(x^{\text{com}}) \le a$ and $OM(x^{\text{plus}}) \ge a$. Let $T^{\text{plus}}(OM)$ be the number of rounds until the $(1+\lambda)$ EA on ONEMAX creates a search point x with $OM(x) \ge b$, and let $T^{\text{com}}(f)$ be the number of rounds until the $(1,\lambda)$ EA on f creates a search point x with $OM(x) \ge b$. Then $T^{\text{com}}(f)$ stochastically dominates $T^{\text{plus}}(OM)$.

(b) Assume $OM(x^{\text{one}}) \ge a$ and $OM(x^{\text{plus}}) \le a$. On ONEMAX, let T^{one} and T^{plus} be the number of function evaluations until the respective algorithm finds a search point x with $OM(x) \ge b$. Then T^{plus} stochastically dominates T^{one}.

(c) Let $x^* \in \{0,1\}^n$ and let $x^f, x^{OM} \in \{0,1\}^n$ be such that $H(x^f, x^*) \ge a$ and $H(x^{OM}, \vec{1}) \le a$. Let T^f be the hitting time of the set $\{x : H(x,x^*) \le b\}$ for the $(1+\lambda)$ EA with $x^{\text{plus}} = x^f$ on f and let T^{OM} be the hitting time of the set $\{x : H(x,\vec{1}) \le b\}$ for the $(1+\lambda)$ EA with $x^{\text{plus}} = x^{OM}$ on ONEMAX. Then T^f stochastically dominates T^{OM}.

Part (b) was already known [23]. For (c), the most natural case is that x^* is the unique global optimum of f, but this is not required.

In fact, all three parts of Theorem 3.4 are just special cases of the following, more general theorem. It says that the $(1+\lambda)$ EA on ONEMAX is faster than any other mutation-based algorithm with the same mutation rate if it creates offspring in batches of size λ. Crucially, this holds for any selection strategy for the parents. Thus, it is also independent of the fitness function, since this "only" decides which individuals may reproduce.

Theorem 3.5. *Let $\lambda \in \mathbb{N}$, $a, b \in [0, n]$, mutation rate $r \leq 1/2$ and let $x^{\mathcal{A}}, x^{\text{plus}} \in \{0,1\}^n$ with $OM(x^{\mathcal{A}}) \leq a$ and $OM(x^{\text{plus}}) \geq a$. Consider any algorithm $\mathcal{A}$ with the following scheme. The algorithm starts by creating $x^{\mathcal{A}}$. In each round, it uses an arbitrary mechanism to select λ (not necessarily distinct) parents among all previously created search points, and creates λ offspring by applying standard bit mutation with mutation probability r to them.*

Let S_t be the set of search points that $\mathcal{A}$ creates in the first t rounds, and let $X^{t,\mathcal{A}} := \max\{OM(x) \mid x \in S_t\}$. Let $X^{t,\text{plus}}$ be the OM-value of the $(1+\lambda)$ EA with standard bit mutation and mutation probability p on OneMax after t rounds if started in x^{plus}. Then $X^{t,\text{plus}}$ stochastically dominates $X^{t,\mathcal{A}}$.

Moreover, let $T^{\mathcal{A}} := \min\{t : X^{t,\mathcal{A}} \geq b\}$ and $T^{\text{plus}} := \min\{t : X^{t,\text{plus}} \geq b\}$. Then $T^{\mathcal{A}}$ stochastically dominates T^{plus}.

The proofs are based on Lemma 6.1 in [36], omitted due to space limitations. We remark that we could have replaced the $(1, \lambda)$ EA in Theorem 3.4(a) by any other algorithm as in Theorem 3.5.

3.3 High-Probability Fixed Target Results

Now we give upper and lower bounds for the time that the $(1+\lambda)$ EA and the $(1, \lambda)$ EA need to reach some target fitness on OneMax, in the regime where the $(1, \lambda)$ EA is efficient. We show that it is exponentially unlikely to deviate from the expectation by more than a constant factor.

The time bounds match known ones for the $(1+\lambda)$ EA [16, 27] and the $(1, \lambda)$ EA [4, 20, 22, 34], albeit that the dependency on λ can be improved slightly, see [3, 13, 31] and we do not have tight leading constants. The strength of our result lies in its generality and exponentially small tail bounds. Related previous work includes upper tail bounds [11] and lower tail bounds for OneMax [32], a review of fixed-target results in [5] and black-box complexity lower bounds with tail bounds for unary unbiased black-box algorithms [31] in a framework similar to ours. The latter work includes a fixed-target scenario of getting close (in Hamming distance) to global or local optima. Our tail bounds are stronger than the previous ones.

Theorem 3.6. *Consider an algorithm $\mathcal{A}$ as in Theorem 3.5 with $r \leq 1/2$ and a fitness function $f\colon \{0,1\}^n \to \mathbb{R}$. Let $a, b \in [0, n]$ with $a > b$ and fix a search point $x^* \in \{0,1\}^n$. Let $T^{\mathcal{A},f}$ denote the number of evaluations made by $\mathcal{A}$ on f, starting with a population of search points that all have Hamming distance at least a to x^*, to reach a search point within Hamming distance at most b of x^*.*

There are positive constants c_1, c_2 such that the following holds.

(1) For every algorithm $\mathcal{A}$, every fitness function f and every target search point x^*,

$$\mathbb{P}\left(T^{\mathcal{A},f} \leq c_1 n \ln(a/b)\right) \leq e^{-\Omega(\min\{a-b,b\})}.$$

(2) For $f = \text{OneMax}$, $x^* = \vec{1}$ and either $\mathcal{A} = (1+\lambda)$ EA with arbitrary λ (including the (1+1) EA) or $\mathcal{A} = (1, \lambda)$ EA with $\lambda \geq \max\{\log_\eta(n/b), 16(e+1)/3\}$,

$$\mathbb{P}\left(T^{\mathcal{A},f} \geq c_2\left((a-b)\lambda + \ln(a/b)n\right)\right) \leq e^{-\Omega(\min\{a-b,b\})}.$$

We omit the proof of Theorem 3.6. It is not based on drift analysis. Instead, one can prove lower tail bounds for the (1+1) EA by arguing that in every step at most a 0-bits may flip and then applying Chernoff bounds to bound the total number of flipping 0-bits, and hence the possible distance decrease, in the stated time. This approach works well if $b = \Omega(a)$. Otherwise, we consider $\approx \log_2(a/b)$ distance intervals $[a \cdot 2^{-1}, a]$, $[a \cdot 2^{-2}, a \cdot 2^{-1}]$, ... and apply the above arguments for each interval, always considering the worst case for the number of ones. By Theorem 3.4, the lower tail bound for the (1+1) EA translates to the stated general setting.

Upper bounds are derived using a similar division into intervals, but we have to account for non-elitism. We use Chernoff bounds to obtain a lower bound for the number of steps in which the distance to the optimum is changed (by positive or negative values). Then we show that, irrespective of the current fitness, the conditional expected progress in distance-changing steps stochastically dominates a random variable with exponentially decaying tail:

$$Z_i := \begin{cases} +1 & \text{with probability } \frac{3}{4} \\ -j & \text{with probability } \frac{3}{4} \cdot 4^{-j}, \text{ for } j \in \mathbb{N}. \end{cases} \tag{11}$$

We prove a Chernoff-type bound for the sum $Z := \sum_{i=1}^{t} Z_i$ and show $\mathbb{P}(Z \leq \mathbb{E}(Z)/2) \leq e^{-7t/8}$. Together, this bounds the total progress in each interval, with the claimed tail bounds.

4 COMMA STRATEGY ESCAPES TRAPS

In this section we prove the upper bound on T^{com} in (1) in Theorem 1.1. The proof consists of three steps: first we introduce dyDisOM, a "dynamic" version of disOM, and study the drift of the $(1, \lambda)$ EA on dyDisOM using drift analysis for a suitable potential function. Studying dyDisOM instead of disOM simplifies the drift analysis, since the "frozen" noise in disOM introduces dependencies with respect to distorted points that are hard to control. Using the bounds on the drift, we compute the expected running time of the $(1, \lambda)$ EA on dyDisOM. Unfortunately, our potential does not satisfy the conditions to employ a standard additive drift theorem with tail bounds. Still, we obtain an "almost matching" bound on the running time which holds w.h.p.

Using the non-sharp upper bound, we develop a 'ratchet' argument (cf. [20]): we show that the algorithm never moves more than $o(\ln^4(n))$ away from the target. This event we use to sharpen our upper bound to obtain concentration around the expected running time. This bootstrapping argument requires a fine control on the drift. In particular, we need to consider steps towards the optimum by more than 1 (contrary to the process in (11)).

Moreover, the ratchet argument allows to argue that no distorted search point is evaluated twice. This will imply that w.h.p. $T^{\text{com,dy}} = T^{\text{com}}$, i.e., the number of function evaluations until the $(1, \lambda)$ EA finds a search point of fitness at least $n - k^*$ on dyDisOM, is the same as the number of function evaluations on disOM.

Dynamic distorted OneMax. We introduce a dynamic version of disOM in which we reveal the sets of distorted and clean points gradually, and in which previously sampled distorted points can become clean later (but not the other way around). Let $s := t\lambda + j$ for $j \in [\lambda]$, write $y^{(s)}$ for the sth sampled search point after initialization, and $x^{(t)}$ for the current search point. With $C_0 = \{x^{(0)}\}$, we iteratively define for $s \geq 1$

$$C_s = \begin{cases} C_{s-1} \cup \{y^{(s)}\}, & \text{w/p } 1-p, \text{if } y^{(s)} \neq x^{(t)}, \\ C_{s-1}, & \text{otherwise.} \end{cases}$$

Note that $C_{s-1} \subseteq C_s$ for all s, reflecting that clean points remain clean forever. Given C_s, we define

$$\textsc{dyDisOM}(y^{(s)}) := \begin{cases} \mathrm{OM}(x), & \text{if } x \in C_s, \\ \mathrm{OM}(x) + d, & \text{otherwise.} \end{cases}$$

We use drift analysis [33] to analyse the $(1,\lambda)$ EA on dyDisOM. For convenience, we use a potential that decreases with the distance from the target, so we use the *ZeroMax* function $\mathrm{ZM}(x) := n - \mathrm{OM}(x)$. We frequently abbreviate $\mathrm{ZM}(x) = k$. We introduce some extra notation to define the potential function. Let $Y_1, \dots, Y_\lambda \sim \mathrm{Bin}(k, 1/n)$ be iid binomial random variables, and define $Y^*(k) := \max(Y_1, \dots, Y_\lambda)$. The random variable Y_i represents the number of 0-bits flipped into a 1 by the ith offspring when the parent has $\mathrm{ZM}(x) = k$. We consider the following potential function for $x \in \{0,1\}^n$ and some suitably chosen constant $\delta > 0$:

$$P(x) := \mathrm{ZM}(x) + \mathbb{1}\{x \notin C_s\}\frac{\delta}{\lambda p}\mathbb{E}\big[Y^*(\mathrm{ZM}(x))\big]. \tag{12}$$

We will compute bounds on the drift

$$\Delta(x) := \mathbb{E}\big[P(x^{(t)}) - P(x^{(t+1)}) \mid x^{(t)} = x\big]. \tag{13}$$

Note that by our sign convention, a positive drift corresponds to progress towards smaller potentials, and thus we want to compute a positive lower bound on $\Delta(x)$ since we aim to establish an upper bound on the running time. We comment briefly on the second term in the potential $P(x)$, which balances two effects: on the one hand it is sufficiently small so that the event that a distorted offspring is found from a clean point (which happens with probability at most λp) yields a small negative contribution to the drift; on the other hand it is sufficiently large so that the drift from a distorted point is of the same order as the drift from a clean point, even though the probability of making a jump is much smaller.

We state four lemmas, then show that an upper bound on T^{com} follows, and eventually give a sketch of one of the new lemmas. The next lemma is due to Gießen and Witt [16].

Lemma 4.1 ([16, Lemma 4]). *Let, for some $k \geq 1$, $Y_1, \dots, Y_\lambda \sim \mathrm{Bin}(k, 1/n)$ be iid binomial random variables, and define $Y^*(k) := \max(Y_1, \dots, Y_\lambda)$. It follows that*

(1) if there exists $\alpha > 0$ with $\alpha = O(1)$ such that $k = n/(\ln^\alpha \lambda)$, and $\lambda = \omega(1)$, then

$$\mathbb{E}[Y^*(k)] = (1 \pm o(1))\frac{\ln(\lambda)}{\ln\ln(\lambda) + \ln(n/k)},$$

(2) for all k, we have $\mathbb{E}[Y^*(k)] \geq \lambda k/(\lambda k + n)$,
(3) if $k = o(n/\lambda)$, then $\mathbb{E}[Y^*(k)] = (1 - o(1))\lambda k/n$.

The following lemma provides useful bounds for the drift analysis. We omit the proof.

Lemma 4.2. *Consider the setting of Lemma 4.1 under Assumption 1. For $k \geq k^*$ it holds that*

(1) $\mathbb{E}[Y^*(k + \ln n)] = (1 + o(1))\mathbb{E}[Y^*(k)]$;
(2) $\ln(n)\lambda p = o(\mathbb{E}[Y^*(k)])$;
(3) $\ln(n)q = o(\mathbb{E}[Y^*(k)])$;
(4) $1/n = o(\mathbb{E}[Y^*(k)])$;
(5) for all $k \leq n/\lambda$, we have $\mathbb{E}[Y^*(k)] = \Omega\big(P(x)\frac{\lambda}{n}\big)$.

The next lemma obtains bounds on the drifts and a non-sharp upper bound on T^{com}, i.e., it contains an additional ln-factor compared to Theorem 1.1. Our lower bound on the drift matches the drift of the $(1,\lambda)$ EA and OneMax with the same parameters (using the ZeroMax potential for those cases).

Lemma 4.3. *Under Assumption 1, we have $\Delta(x) = \Omega(\mathbb{E}[Y^*(k)])$ for all x with $k \geq k^*$. Moreover, $T^{\mathrm{com,dy}}_{k^*,\lambda,d,p} \leq n\ln^2 n$ w.h.p.*

We omit the proof. It is based on a careful case analysis (distinguishing whether there is a clone of the parent and whether there is a distorted offspring) to bound the drift and obtain a lower bound on $\mathbb{E}[T^{\mathrm{com,dy}}]$, and Markov's inequality to obtain a tail bound.

The following lemma shows that even though the noise in dyDisOM is dynamic, these dynamics are not seen by the algorithm since each sampled point always returns the same function value w.h.p. Moreover, it shows that for all $t \leq T^{\mathrm{com}}$, the algorithm never jumps to a search point with much smaller OM-value than the current search point $x^{(t)}$.

Lemma 4.4. *Under Assumption 1, for dyDisOM it holds w.h.p. that if there exists $s < t \leq T^{\mathrm{com,dy}}$ such that $y^{(s)} = y^{(t)}$, then dyDisOM$(y^{(s)})$ = dyDisOM$(y^{(t)})$. Moreover, for any $\varepsilon > 0$ and n sufficiently large, w.h.p.*

$$\big\{\forall t < t' \leq \lceil T^{\mathrm{com}}/\lambda\rceil : OM(x^{(t')}) \geq OM(x^{(t)}) - \varepsilon\ln^4(n)\big\}.$$

We postpone a sketch of the proof and first prove the upper bound on T^{com}, assuming Lemma 4.4.

Proof of Theorem 1.1, upper bound on T^{com}.
We consider a run of the algorithm on dyDisOM, and make the connection to disOM at the end of the proof. We split the run of the $(1,\lambda)$ EA on dyDisOM into two phases. Let T_1 denote the number of function evaluations until the first time that the $(1,\lambda)$ EA moves to a point X_{T_1} that has ZM-value at most $n/\lambda - \varepsilon\ln^4(n)$, where ε is as in Lemma 4.4. Let T_2 denote the number of function evaluations after moving to X_{T_1} until moving to a point with ZM-value at most k^*. Then $T^{\mathrm{com,dy}} \leq T_1 + T_2$, since a distorted point with ZM-value at most $k^* + d$ (which has fitness at least $n - k^*$) may have been found before $T_1 + T_2$. We will argue that w.h.p. $T_1, T_2 = O(n\ln n)$.

We first consider T_1, for which we apply a variable drift theorem and Markov's inequality (the presence of the indicator in the potential in (12) prevents direct use of drift theorems with tail bounds, since the decay on the jump size distribution does not decay exponentially). By Lemmas 4.1 and 4.3, we obtain that

$$\Delta(x) \geq \begin{cases} \Omega\Big(\dfrac{\ln\lambda}{2\ln\ln\lambda}\Big), & \text{if } k \geq n/\ln\lambda, \\ \Omega(1), & \text{if } k \geq n/\lambda - \varepsilon\ln^4 n. \end{cases}$$

By the variable drift theorem as stated in [33, Theorem 2.3.3], the expected number of generations $\mathbb{E}[\lceil T_1/\lambda\rceil]$ is of order at most

$$n\frac{\ln\ln\lambda}{\ln\lambda} + \frac{n}{\ln\lambda} = O\Big(n\frac{\ln\ln\lambda}{\ln\lambda}\Big). \tag{14}$$

Since the number of function evaluations is a factor λ larger than the number of generations, it follows by Markov's inequality that

$$\mathbb{P}\big(T_1 \geq n\ln n\big) \leq \frac{O\Big(n\frac{\ln\ln\lambda}{\ln\lambda}\lambda\Big)}{n\ln n} = O\Big(\frac{\ln\ln\ln n}{\ln\ln n}\Big) = o(1), \tag{15}$$

using that $\lambda = \Theta(\ln(n))$ by Assumption 1.

Recall that at time T_1 the current search point x satisfies $\mathrm{ZM}(x) \leq n/\lambda - \varepsilon \ln^4 n$. By Lemma 4.4, w.h.p. the $(1,\lambda)$ EA never visits a search point x' with $\mathrm{ZM}(x') > n/\lambda$ after time T_1. By Lemma 4.3 and Lemma 4.1(5), we have for all x with $\mathrm{ZM}(x) \leq n/\lambda$ that

$$\Delta(x) = \Omega\big(\mathbb{E}[Y^*(k)]\big) = \Omega(P(x)\cdot\lambda/n),$$

which corresponds to a multiplicative drift for all the visited search points after T_1 w.h.p. By the multiplicative drift theorem with tail bounds [11, Theorem 5], w.h.p. the number of generations to reach a point with $\mathrm{ZM}(x) \leq k^*$ from X_{T_1} is at most $O(\ln(n)\cdot n/\lambda)$. The number of function evaluations in this phase is by a factor λ larger, namely $O(n\ln n)$, as required. Combined with (14) this implies that $T^{\mathrm{com,dy}} = O(n\ln n)$ w.h.p.

We will now translate this bound on DYDISOM (with dynamic noise) to a bound on DISOM (with frozen noise). For DYDISOM and DISOM, each point is distorted at the first time it is sampled with probability p independently of other points. Consequently, $x^{(0)}$ is clean w.h.p. Moreover, by Lemma 4.4 w.h.p. the function values of points sampled in DYDISOM never change, so the runs on the dynamic and the 'frozen' model are identical. Hence, the upper bound in (1) on $T^{\mathrm{com}}_{k^*,\lambda,d,p}$ follows from $T^{\mathrm{com,dy}}_{k^*,\lambda,d,p} = O(n\ln n)$. □

We omit the proofs of Lemmas 4.2–4.3, but provide a sketch of the proof of Lemma 4.4.

Proof sketch of Lemma 4.4. We verify the first statement of Lemma 4.4, and omit the proof of the second statement. The only way there could exist $t' < t$ such that $y^{(t')} = y^{(t)}$, but $\textsc{dyDisOM}(y^{(t')}) \neq \textsc{dyDisOM}(y^{(t)})$, is when $y^{(t')}$ is distorted and resampled at a later time t at which it is clean (clean points remain clean). Hence, it suffices to argue that w.h.p. the algorithm never resamples a point that was distorted at the first time it was sampled. To do so, we will analyse the positive and negative jumps of the $(1,\lambda)$ EA on DYDISOM in polynomial time intervals.

Let $s := C(n/k^*)\ln^4(n)$ for some large constant $C > 0$, and define for the t-th function evaluation the random times $t_1(t),\dots,t_s(t)$, which are the first s unique evaluations after the $(t-1)$-st evaluation from a clean parent (for convenience we assume $s \in \mathbb{N}$). It can be shown that there exists $C', \varepsilon > 0$ such that w.h.p. the following three events hold:

(i) $\forall t \leq n^2$, the total time in the interval $[t, t_s(t)]$ at which the parent is distorted is at most $O(\ln^3(n)/q)$.
(ii) $\forall t \leq n^2, r \in [t_s(t), T^{\mathrm{com,dy}}]$, each sampled point $y^{(r)}$ satisfies $\mathrm{OM}(y^{(r)}) > \mathrm{OM}(x^{(\lceil t/\lambda\rceil-1)}) + C'\ln(n)$.
(iii) the number of sampled distorted points until the fixed target is reached is $O(pn\ln(n))$.

We show here that the statement follows under the assumption that these events hold w.h.p. Recall that it is sufficient to bound the probability that a distorted point is resampled. If an offspring $y^{(t)}$ with $\mathrm{OM}(y^{(t)}) = k$ is distorted, then by event (i) its parent (which is $x^{(\lceil t/\lambda\rceil-1)}$) has OM-value at least $k - C'\ln(n)$. By event (iii), the only times at which the distorted point $y^{(t)}$ can be resampled is during the interval $[t, t_s(t)]$. At each time $r \in [t, t_s(t)]$ the probability that $y^{(t)}$ is resampled is at most $1/n$. By a union bound over the precisely $s = \Theta((n/k^*)\ln^4(n))$ times in $[t, t_s(t)]$ at which the parent is clean (definition of $t_1(t),\dots,t_s(t)$), and the $O(\ln^3(n)/q)$ times at which the parent is distorted (event (i)), the probability that the point $y^{(r)}$ is resampled in this interval is at most

$$O(\max\{\ln^4(n)/k^*, \ln^3(n)/(nq)\}).$$

By a union bound over the at most $O(pn\ln(n))$ distorted points visited (event (iii)), no distorted point is resampled with probability at least

$$1 - O(pn\ln(n)\cdot\max\{\ln^4(n)/k^*, \ln^3(n)/(nq)\})$$
$$= 1 - O(\max\{\ln^5(n)\cdot p\cdot n/k^*, \ln^4(n)p/q\}) = 1 - o(1),$$

since $p \leq k^*/n^{1+\delta} = o(k^*/(n\ln^5(n)))$ and $p \leq qn^{-\delta} = o(q/\ln^4(n))$ by Assumption 1, see (8). Thus, the first lemma statement follows if the three events hold with high probability. □

5 TRAPS SLOW DOWN PLUS STRATEGY

Proof sketch of Theorem 1.1, lower bound on T^{plus}. Let us consider distances from $\vec{1}$ in the two intervals $I := (2k^*, 2k^*n^\delta]$ and $I' := [k^*n^\delta, k^*n^{2\delta}] \subset I$ for some sufficiently small $\delta > 0$ so that $k^*n^{2\delta}\lambda = o(n)$ (which exists by Assumption 1). Note that all points in I have fitness less than $n - 2k^* + d \leq n - k^*$, so it is necessary to traverse I. We claim that w.h.p., with $\mathcal{D}$ the set of distorted points,

(i) we visit a distorted point with ZM-value in I',
(ii) afterwards the algorithm does not leave $\mathcal{D}$ for time $n\ln(n)/p$, and
(iii) it takes time $\Omega(n\ln(n)/p)$ to traverse I (that has size $|I| = n^{\Omega(1)}$) within $\mathcal{D}$ when starting in I'.

The three items imply the lower bound (2). We omit the proofs. □

6 PLUS STRATEGY STILL ESCAPES

The main goal of this section is to sketch the proof of Theorem 1.3. We remark that the results obtained here immediately imply the upper bound on T^{plus} in Theorem 1.1. We split the proof of Theorem 1.3 into two cases. We start with a rather simple lemma for the case which covers the first two settings in Theorem 1.3.

Lemma 6.1. *Consider the setting of Theorem 1.3. There exists a constant $C' > 0$ such that w.h.p.*

$$T^{\mathrm{plus}} = \begin{cases} T, & \text{if } p\cdot T = o(1), \text{ or } \lambda \geq C'\ln(n), \\ O\big(n\cdot(\lambda + \ln(n/k^*))\big), & \text{if } p\cdot n\cdot(\lambda+\ln(n/k^*)) = o(1). \end{cases}$$

For small values of p we argue that the $(1+\lambda)$ EA never encounters a distorted point w.h.p., while for $\lambda \geq C'\ln(n)$ the $(1,\lambda)$ EA mimics the $(1+\lambda)$ EA. We omit the details.

For other values of p and λ, the key step is to show that w.h.p. the $(1+\lambda)$ EA is at most a factor $O(1/p)$ slower on DISOM than on OM, which is shown by the following lemma. After a discussion we will show that the upper bounds on T^{plus} follows from the domination results in Section 3.2, using that the $(1+\lambda)$ EA on OM is at least as fast as the $(1,\lambda)$ EA on DISOM.

Lemma 6.2. *Consider the setting of Theorem 1.3, and assume additionally that $p\cdot n\cdot(\lambda+\ln(n/k^*)) = \Omega(1)$. Let $T^{OM} = T^{OM,\mathrm{plus}}$ be the fixed-target hitting time of the $(1+\lambda)$ EA on OneMax for target fitness $n-k^*$, and similarly for T^{DISOM}. If $T^{OM} \leq n^{C'}$ holds w.h.p. for some constant $C' > 0$, then w.h.p.*

$$T^{DISOM} = O\big(T^{OM}/p\big).$$

This statement seems very intuitive: the algorithm can always make progress by staying within the distorted points. If the probability of making an improving step on OM is p_{imp}, then the probability of making an improving step on DISOM should be $p_{\text{imp}} \cdot p$ since we need to find a OM-improving step to a distorted offspring.

However, this intuition can be misleading. The problem is that the distortions are fixed, and that we do not get fresh randomness each time. Let us focus on single-bit flips for illustration. In distance k from the optimum, since k of the n neighbours are improving, each single-bit flip has a chance of k/n to be improving, so on ONEMAX we need to wait for n/k single-bit flips in expectation.

For DISOM, if we are in a distorted point x at distance k from the optimum, then the probability that a single-bit flip is distorted *and* OM-improving (i.e., closer to $\vec{1}$ than the parent) is naively pk/n, so it is tempting to assume that one simply needs to wait for $n/(pk)$ rounds in expectation. However, this is not true! Once we have queried the fitness of a OM-improving neighbour and it was clean, the chance is gone for good. It could happen that all k OM-improving neighbours of x are clean. In fact, this is *likely* since the expected number of OM-improving distorted neighbours is pk, and $pk = o(1)$ is a perfectly normal situation.[4] In this case, we can never escape from x by a single-bit flip. (In other words: DISOM has a local optimum, which is the whole point of this new benchmark after all.) So in this case, we need at least *two-bit flips* to escape.

This could potentially be very costly, but fortunately we can profit from two-bit flips to *the same OM-level*, i.e., to search points in the same Hamming distance from the optimum. Those two-bit flips are much cheaper than *OM-improving* two-bit flips and provide fresh randomness. Mind that this is a real and important issue, and the above lemma would simply be wrong if the $(1+\lambda)$ EA would break ties in favour of the parent. This is also why we are uncertain whether Theorem 1.3 transfers similarly to other functions. In fact, we conjecture that it is false for other linear functions.

In the proof, we call a distorted search point *good* if at most half of its OM-improving neighbours have been queried before. In this case, the naive intuition is still correct, since each OM-improving neighbour has probability at least $1/2$ to be unqueried, in which case it is distorted with probability p. Thus progress within distorted points is only slowed down by a factor of $O(1/p)$. To deal with bad points, we define $N^{(i)}(x)$ as the set of all search points in distance $2i$ from x which can be reached via i consecutive 2-bit flips, all of which stay within distorted points. Informally, we then show for some constant i_0 that w.h.p. *(i)* at least half of the search points in $N^{(i_0)}(x)$ are good, and *(ii)* the algorithm moves from x to an almost uniform random point in $N^{(i_0)}(x)$ in expected time $O(1/(pp_{\text{imp}}) + \lambda)$, conditionally on not leaving $\mathcal{D}_k$. This allows us to prove that the algorithm visits good search points frequently, on which it then has good improvement chances. We omit the details. Now we have all ingredients to prove Theorem 1.3.

Proof of Theorem 1.3. Since we switch between the fitness functions $f \in \{\text{OM}, \text{DISOM}\}$ and several target fitnesses k^*, we include the indices f and k^* in the notation.

The first two cases are implied by Lemma 6.1. For the third case, we observe that a necessary condition for reaching fitness level $n - k^*$ is to reach Hamming distance $k^* + d$ from $\vec{1}$, so $T^{\text{DISOM,com}}_{k^*+d} \le T$. By Theorem 3.5(a) the $(1,\lambda)$ EA on DISOM is at most as fast as the $(1+\lambda)$ EA on OM, so we also have $T^{\text{OM,plus}}_{k^*+d} \le T$ w.h.p. By Theorem 3.6 we have $T^{\text{OM,plus}}_{k^*} = \Theta(T^{\text{OM,plus}}_{k^*+d})$ w.h.p., where we use $\lambda = O(\ln(n/k^*))$ and $k^* \le k^* + d \le 2k^* \le n/3$. Hence, w.h.p. $T^{\text{OM,plus}}_{k^*} = O(T)$. Then we use Lemma 6.2 to conclude $T^{\text{DISOM,plus}}_{k^*} = O(T/p)$.

We show now that $T^{\text{plus}} = O(T \cdot n \ln n)$. For parameters as in the first case this is trivial. Assume otherwise, so that $\lambda = O(\ln n)$. If $p \cdot n(\lambda + \ln(n/k^*)) = o(1)$, then the second case yields $T^{\text{plus}} = O(n \ln n) = O(T \cdot n \ln n)$. If $p \cdot n(\lambda + \ln(n/k^*)) = \Omega(1)$, then the third case yields $T = O(T \cdot n(\lambda + \ln(n/k^*))) = O(T \cdot n \ln n)$. □

7 COMBINING ALL RESULTS

We verify that the previous sections combined prove Theorem 1.1.

Proof of Theorem 1.1. The upper bound on T^{com} is proven in Section 4. We verify now the lower bound on T^{com}. Observe that T^{com} stochastically dominates the time T' until the $(1,\lambda)$ EA finds an offspring y with $\text{OM}(y) \ge k^* + d$. By Theorem 3.6(1) it follows that $T' = \Omega(n \ln n)$ (setting $\mathcal{A}$ =$(1,\lambda)$ EA, $a = \Theta(n)$ and $b = k^* + d$).

We turn to T^{plus}. The upper bound follows immediately from the upper bound on T^{com}, since by Theorem 1.3 (which holds for a wider range of parameters than assumed in Assumption 1 in Theorem 1.1) the runtime of the $(1+\lambda)$ EA on DISOM is at most a factor $O(1/p)$ slower than on OM when $\lambda = \Theta(\ln n)$ and $k^* = n^{1-\Omega(1)}$. The lower bound on T^{plus} is given in Section 5. □

8 CONCLUSION

We have shown that a comma strategy can indeed help for dealing with local optima. To this end, we have introduced the new theoretical benchmark DISOM. We believe that this benchmark is of wider interest for studying local optima. As discussed in the introduction, arguably the popular benchmarks JUMP and CLIFF have rather atypical local optima, and DISOM is a very simple way of adding local optima to the simple OM function. Thus, it would be very interesting to investigate how other non-elitist selection mechanisms like tournament selection [30], linear ranking selection, or fitness-proportionate selection [18] perform (see [17, 29] for overviews on non-elitist selection), and whether this can be phrased more generally in terms of selective pressure [28].

Our proof also gives insights into how the $(1,\lambda)$ EA escapes local optima. In particular, in the DISOM landscape under Assumption 1, our proof shows that the $(1,\lambda)$ EA escapes local optima for good: after escaping, it never hits the same local optimum a second time.

Of course, ONEMAX is not the only function that can be distorted. The same process can be applied to any other function, for example to any linear function. As discussed after Lemma 6.2, we suspect that for the $(1+\lambda)$ EA there is a real difference between ONEMAX and other linear functions, and that the huge fitness plateaus of ONEMAX are important for the $(1+\lambda)$ EA to be efficient.

ACKNOWLEDGMENTS

We are thankful for the fruitful discussions at the Dagstuhl seminar 22081 "Theory of Randomized Optimization Heuristics", which triggered this research, as well as the Dagstuhl seminar 22182 "Estimation-of-Distribution Algorithms: Theory and Applications".

[4] Recall that p can be arbitrarily close to $1/n$. In fact, this is a particularly interesting case since it gives the largest factor between T^{com} and T^{plus} in Theorem 1.1.

REFERENCES

[1] Denis Antipov, Benjamin Doerr, and Quentin Yang. 2019. The efficiency threshold for the offspring population size of the (μ, λ) EA. In *Genetic and Evolutionary Computation Conference (GECCO 2019)*. 1461–1469.

[2] Anne Auger, Carlos M. Fonseca, Tobias Friedrich, Johannes Lengler, and Armand Gissler. 2022. Theory of Randomized Optimization Heuristics (Dagstuhl Seminar 22081). *Dagstuhl Reports* 12, 2 (2022), 87–102.

[3] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. 2014. Unbiased Black-Box Complexity of Parallel Search. In *Parallel Problem Solving from Nature (PPSN 2014)*. Springer, 892–901.

[4] Jakob Bossek and Dirk Sudholt. 2021. Do additional optima speed up evolutionary algorithms?. In *Foundations of Genetic Algorithms (FOGA 2021)*, Vol. 8. 1–11.

[5] Maxim Buzdalov, Benjamin Doerr, Carola Doerr, and Dmitry Vinokurov. 2022. Fixed-Target Runtime Analysis. *Algorithmica* 84, 6 (2022), 1762–1793.

[6] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. 2021. Escaping local optima with non-elitist evolutionary algorithms. In *AAAI Conference on Artificial Intelligence (AAAI 2021)*, Vol. 35. 12275–12283.

[7] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. 2021. Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys. In *Genetic and Evolutionary Computation Conference (GECCO 2021)*. 1133–1141.

[8] Benjamin Doerr. 2019. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science* 773 (2019), 115–137.

[9] Benjamin Doerr. 2020. Does comma selection help to cope with local optima?. In *Genetic and Evolutionary Computation Conference (GECCO 2020)*. 1304–1313.

[10] Benjamin Doerr. 2021. The runtime of the compact genetic algorithm on jump functions. *Algorithmica* 83 (2021), 3059–3107.

[11] Benjamin Doerr and Leslie Ann Goldberg. 2013. Adaptive Drift Analysis. *Algorithmica* 65, 1 (2013), 224–250.

[12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 4, 64 (2012), 673–697.

[13] Benjamin Doerr and Marvin Künnemann. 2013. How the (1+λ) Evolutionary Algorithm Optimizes Linear Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013)*. ACM, 1589–1596.

[14] Carola Doerr and Johannes Lengler. 2017. Introducing elitist black-box models: When does elitist behavior weaken the performance of evolutionary algorithms? *Evolutionary Computation* 25, 4 (2017), 587–606.

[15] Tobias Friedrich, Timo Kötzing, Frank Neumann, and Aishwarya Radhakrishnan. 2022. Theoretical Study of Optimizing Rugged Landscapes with the cGA. In *Parallel Problem Solving from Nature (PPSN 2022)*. Springer, 586–599.

[16] Christian Gießen and Carsten Witt. 2017. The Interplay of Population Size and Mutation Probability in the (1+λ) EA on OneMax. *Algorithmica* 78, 2 (2017), 587–609.

[17] David E Goldberg and Kalyanmoy Deb. 1991. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms (FOGA 1991)*. Vol. 1. 69–93.

[18] Edda Happ, Daniel Johannsen, Christian Klein, and Frank Neumann. 2008. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*. 953–960.

[19] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2021. Self-Adjusting Offspring Population Sizes Outperform Fixed Parameters on the Cliff Function. In *Foundations of Genetic Algorithms (FOGA 2021)*, Vol. 5. 1–5.

[20] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2021. Self-Adjusting Population Sizes for Non-Elitist Evolutionary Algorithms: Why Success Rates Matter. In *Genetic and Evolutionary Computation Conference (GECCO 2021)*. 1151–1159.

[21] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2022. Hard Problems Are Easier for Success-Based Parameter Control. In *Genetic and Evolutionary Computation Conference (GECCO 2022)*. 796–804.

[22] Jens Jägerskupper and Tobias Storch. 2007. When the plus strategy outperforms the comma strategyand when not. In *Foundations of Computational Intelligence (FOCI 2007)*. 25–32.

[23] Thomas Jansen, Kenneth A De Jong, and Ingo Wegener. 2005. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation* 13, 4 (2005), 413–440.

[24] Joost Jorritsma, Johannes Lengler, and Dirk Sudholt. 2023. Comma Selection Outperforms Plus Selection on OneMax with Randomly Planted Optima. *arXiv preprint arXiv:2304.09712* (2023).

[25] Marc Kaufmann, Maxime Larcher, Johannes Lengler, and Xun Zou. 2022. Self-adjusting Population Sizes for the (1, λ)-EA on Monotone Functions. In *Parallel Problem Solving from Nature (PPSN 2022)*. Springer, 569–585.

[26] Marc Kaufmann, Maxime Larcher, Johannes Lengler, and Xun Zou. 2023. OneMax Is Not the Easiest Function for Fitness Improvements. In *23rd European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2023)*. Springer, 162–178.

[27] Jörg Lässig and Dirk Sudholt. 2011. Adaptive Population Models for Offspring Populations and Parallel Evolutionary Algorithms. In *Foundations of Genetic Algorithms (FOGA 2011)*. ACM, 181–192.

[28] Per Kristian Lehre. 2010. Negative drift in populations. In *Parallel Problem Solving from Nature (PPSN 2010)*. Springer, 244–253.

[29] Per Kristian Lehre. 2011. Fitness-levels for non-elitist populations. In *Genetic and Evolutionary Computation Conference (GECCO 2011)*. 2075–2082.

[30] Per Kristian Lehre and Xiaoyu Qin. 2022. More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments. *Algorithmica* (2022), 1–46.

[31] Per Kristian Lehre and Dirk Sudholt. 2020. Parallel Black-Box Complexity with Tail Bounds. *IEEE Transactions on Evolutionary Computation* 24, 6 (2020), 1010–1024.

[32] Per Kristian Lehre and Carsten Witt. 2021. Tail bounds on hitting times of randomized search heuristics using variable drift analysis. *Combinatorics, Probability and Computing* 30, 4 (2021), 550–569.

[33] Johannes Lengler. 2020. Drift analysis. In *Theory of Evolutionary Computation*. Springer, 89–131.

[34] Jonathan E Rowe and Dirk Sudholt. 2014. The choice of the offspring population size in the (1, λ) evolutionary algorithm. *Theoretical Computer Science* 545 (2014), 20–38.

[35] Dirk Sudholt. 2013. A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 418–435.

[36] Carsten Witt. 2013. Tight Bounds on the Optimization Time of a Randomized Search Heuristic on Linear Functions. *Combinatorics, Probability and Computing* 22, 2 (2013), 294–318.