Optimal Mutation Rates for the $(1 + \lambda)$ EA on OneMax

Maxim Buzdalov[1] and Carola Doerr[2]

[1] ITMO University, Saint Petersburg, Russia, mbuzdalov@gmail.com
[2] Sorbonne Université, CNRS, LIP6, Paris, France, Carola.Doerr@lip6.fr

Abstract. The OneMax problem, alternatively known as the Hamming distance problem, is often referred to as the "drosophila of evolutionary computation (EC)", because of its high relevance in theoretical and empirical analyses of EC approaches. It is therefore surprising that even for the simplest of all mutation-based algorithms, Randomized Local Search and the (1+1) EA, the optimal mutation rates were determined only very recently, in a GECCO 2019 poster.
In this work, we extend the analysis of optimal mutation rates to two variants of the $(1 + \lambda)$ EA and to the $(1 + \lambda)$ RLS. To do this, we use dynamic programming and, for the $(1 + \lambda)$ EA, numeric optimization, both requiring $\Theta(n^3)$ time for problem dimension n. With this in hand, we compute for all population sizes $\lambda \in \{2^i \mid 0 \leq i \leq 18\}$ and for problem dimension $n \in \{1000, 2000, 5000\}$ which mutation rates minimize the expected running time and which ones maximize the expected progress. Our results do not only provide a lower bound against which we can measure common evolutionary approaches, but we also obtain insight into the structure of these optimal parameter choices. For example, we show that, for large population sizes, the best number of bits to flip is not monotone in the distance to the optimum. We also observe that the expected remaining running times are not necessarily unimodal for the $(1 + \lambda)$ EA$_{0 \rightarrow 1}$ with shifted mutation.

Keywords: Parameter Control · Optimal Mutation Rates · Population-based Algorithms · OneMax

1 Introduction

Evolutionary algorithms (EAs) are particularly useful for the optimization of problems for which algorithms with proven performance guarantee are not known; e.g., due to a lack of knowledge, time, computational power, or access to problem data. It is therefore not surprising that we observe a considerable gap between the problems on which EAs are applied, and those for which rigorously proven analyses are available [12].

If there is a single problem that stands out in the EA theory literature, this is the OneMax problem, which is considered to be "the drosophila of evolutionary computation" [16]. The OneMax problem asks to maximize the simple linear function that counts the number of ones in a bit string, i.e., $\mathrm{OM}(x) = \sum_{i=1}^{n} x_i$.

This function is, of course, easily optimized by sampling the unique optimum $(1, \ldots, 1)$. However, most EAs show identical performance on OneMax as on any problem asking to minimize the Hamming distance $H(z, \cdot)$ to an unknown string z, i.e., $f_z(x) = n - H(z, x)$, which is a classical problem studied in various fields of Computer Science, starting in the early 60s [15]. In the analysis of EAs, OneMax typically plays the role of a benchmark problem that is easy to understand, and on which one can easily test the hill-climbing capabilities of the considered algorithm; very similar to the role of the sphere function in derivative-free numerical optimization [1, 18].

Despite its popularity, and numerous deep results on the OneMax problem (see [12] for examples), there are still a number of open questions, and this even for the simplest settings in which the problem is static and noise-free, and the algorithms under consideration can be described in a few lines of pseudo-code. One of these questions concerns the optimal mutation rates of the $(1 + \lambda)$ EA, i.e., the algorithm which always keeps in memory a best-so-far solution x, and which samples in each iteration λ "offspring" by applying standard bit mutation to x. By optimal mutation rates we refer to the values that minimize the expected optimization time, i.e., the average number of function evaluations needed until the algorithm evaluates for the first time an optimal solution. It is not very difficult to see that the optimal mutation rate of this algorithm as well as of its Randomized Local Search (RLS) analog (i.e., the algorithm applying a deterministic mutation strength rather than a randomly sampled one) depend only on the function value $\mathrm{OM}(x)$ of the current incumbent [2, 3, 10]. However, even for $\lambda = 1$ the optimal mutation rates were numerically computed only in the recent work [4]. Prior to [4], only the rates that maximize the expected progress and those that yield asymptotically optimal running times (in terms of big-Oh notation) were known, see discussion below. It was shown in [4] that the optimal mutation rates are not identical to those maximizing the expected progress, and that the differences can be significant when the current Hamming distance to the optimum is large. In terms of running time, however, the *drift-maximizing* mutation rates are known to yield almost optimal performance, which is another result that was proven only recently [10] (more precisely, it was proven there for Randomized Local Search (RLS), but the result is likely to extend to the (1+1) EA and its $(1 + \lambda)$ variants).

Our Contribution. We extend in this work the results from [4] to the case $\lambda \in \{2^i \mid i \in [0..18]\}$. As in [4] we do not only focus on the standard $(1 + \lambda)$ EA, but we also consider the $(1+\lambda)$ equivalent of RLS and we consider the $(1+\lambda)$ EA with the "shift" mutation operator suggested in [7]. The shift mutation operator $0 \to 1$ flips exactly one randomly chosen bit when the sampled mutation strength of the standard bit mutation operator equals zero.

Differently from [4] we do not only store the optimal and the drift-maximizing parameter settings for the three different algorithms, but we also store the expected remaining running time of the algorithm that always applies the same fixed mutation rate as long as the incumbent has distance d to the optimum and that applies the optimal mutation rate at all distances $d' < d$. With these values at hand, we can compute the *regret* of each mutation rate, and summing these regrets for a given $(1+\lambda)$-type algorithm gives the exact expected running time, as well as the cumulative regret, which is the expected performance loss of the considered algorithm against the optimal strategy.

Our results extend the main observation shared in [4], which states that, for the (1+1) EA, the drift-maximizing mutation rates are not always also optimal, to the $(1 + \lambda)$ RLS and to both considered $(1 + \lambda)$ EAs. We also show that the drift-maximizing and the optimal mutation rates are almost identical across different dimensions, when compared against the normalized distance d/n.

We also show that, for large population sizes, the optimal number of bits to flip is not monotone in the distance to the optimum. Moreover, we observe that the expected remaining running time is not necessarily unimodal for the $(1+\lambda)$ $\text{EA}_{0\to 1}$ with shifted mutation. Another interesting finding is that some of the drift-maximizing mutation strengths of the $(1+\lambda)$ RLS with $\lambda > 1$ are even, whereas it was proven in [10] that for the (1+1) EA the drift-maximizing mutation strength must always be uneven. The distance d at which we observe even drift-maximizing mutation strengths decreases with λ, whereas its frequency increases with λ.

Applications of Our Results in the Analysis of Parameter Control Mechanisms. Apart from providing several data-driven conjectures about the formal relationship between the optimal and the drift-maximizing parameter settings of the investigated $(1+\lambda)$ algorithms, our results have immediate impact on the analysis of parameter control techniques. Not only do we provide an accurate lower bound against which we can measure the performance of other algorithms, but we can also very easily identify where potential performance losses originate from. We demonstrate such an example in Sec. 6, and recall here only that, despite its discussed simplicity, OneMax is a very commonly used test case for all types of parameter control mechanisms – not only for theoretical studies [9], but also in purely empirical works [22, 25].

OneMax Does Not Require Offspring Population. It is well known that, for the optimization of OneMax, the $(1 + 1)$ EA is the most efficient among the $(1 + \lambda)$ EAs [21] when measuring performance by fitness evaluations. In practice, however, the λ offspring can be evaluated in parallel, so that – apart from mathematical curiosity – the influence of the population size, the problem size, and the distance to the optimum on the optimal (and on the drift-maximizing) mutation rates also has practical relevance.

Related Work. Tight running time bounds for the $(1 + \lambda)$ EA with *static* mutation rate $p = c/n$ are proven in [19]. For constant λ, these bounds were further refined in [20]. The latter also presents optimal static mutation rates for selected combinations of population size λ and problem size n.

For the here-considered *dynamic* mutation rates, the following works are most relevant to ours. Bäck [2] studied, by numerical means, the drift-maximizing mutation rates of the classic $(1+\lambda)$ EA with standard bit mutation, for problem size $n = 100$ and for $\lambda \in \{1, 5, 10, 20\}$. Mutation rates which minimize the expected optimization time in big-Oh terms were derived in [3, Theorem 4]. More

Algorithm 1: Blueprint of an elitist $(1+\lambda)$ unbiased black-box algorithm maximizing a function $f : \{0,1\}^n \to \mathbb{R}$.

```
1 Initialization: Sample $x \in \{0,1\}^n$ uniformly at random and evaluate $f(x)$;
2 Optimization: for $t = 1, 2, 3, \dots$ do
3     for $i = 1, \dots, \lambda$ do
4         Sample $k(i) \sim D(n, f(x))$;
5         $y^{(i)} \leftarrow \texttt{flip}_{k(i)}(x)$;
6         evaluate $f(y^{(i)})$;
7     $y \leftarrow \text{select}\left(\arg\max\{f(y^{(i)}) \mid i \in [\lambda]\}\right)$;
8     if $f(y) \geq f(x)$ then $x \leftarrow y$;
```

precisely, it was shown there that the $(1+\lambda)$ EA using mutation rate $p(\lambda, n, d) = \max\{1/n, \ln(\lambda)/(n \ln(en/d))\}$ needs $O\left(\frac{n}{\ln \lambda} + \frac{n \log n}{\lambda}\right)$ function evaluations, on average, to find an optimal solution. This is asymptotically optimal among all λ-parallel mutation-only black-box algorithms [3, Theorem 3]. Self-adjusting and self-adaptive $(1+\lambda)$ EAs achieving this running time were presented in [11] and [13], respectively.

2 OneMax and $(1+\lambda)$ Mutation-Only Algorithms

As mentioned, the classical OneMax function OM simply counts the number of ones in the string, i.e., $\text{OM} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^n x_i$. For all algorithms discussed in this work, the behavior on OM is identical to that on any of the problems $\text{OM}_z : \{0,1\}^n \to \mathbb{R}, x \mapsto n - H(z,x) := |\{i \in [n] \mid x_i \neq z_i\}|$. We study the maximization of these problems.

Algorithm 1 summarizes the structure of the algorithms studied in this work. All algorithms start by sampling a uniformly chosen point x. In each iteration, λ offspring $y^{(1)}, \dots, y^{(\lambda)}$ are sampled from x, independently of each other. Each $y^{(i)}$ is created from the incumbent x by flipping some $k(i)$ bits, which are pairwise different, independently and uniformly chosen (this is the operator `flip` in line 4). The best of these λ offspring replaces the incumbent if it is at least as good as it (line 8). When $\arg\max\{f(y^{(i)}) \mid i \in [\lambda]\}$ contains more than one point, the selection operator select chooses one of them, e.g., uniformly at random or via some other rule. As a consequence of the symmetry of OneMax, all results shown in this work apply regardless of the chosen tie-breaking rule.

What is left to be specified is the distribution $D(n, f(x))$ from which the *mutation strengths* $k(i)$ are chosen in line 3. This is the only difference between the algorithms studied in this work.

Deterministic vs. Random Sampling: The Randomized Local Search variants (RLS) use a deterministic mutation strength $k(i)$, i.e., the distributions $D(n, f(x))$ are one-point Dirac distributions. We distinguish two EA variants: the one using standard bit mutation, denoted $(1+\lambda)$ EA_{sbm}, and the one using the shift mutation suggested in [7], which we refer to as $(1+\lambda)$ EA$_{0\to1}$. *Standard bit mutation* uses the binomial distribution $\mathrm{Bin}(n,p)$ with n trials and success probability p. The *shift mutation* operator uses $\mathrm{Bin}_{0\to1}(n,p)$, which differs from $\mathrm{Bin}(n,p)$ only in that all probability mass for $k=0$ is moved to $k=1$. That is, with shift mutation we are guaranteed to flip at least one bit, and the probability to flip exactly one bit equals $(1-p)^n + np(1-p)^{n-1}$. In both cases we refer to p as the *mutation rate*.

Optimal vs. Drift-maximizing Rates: Our main interest is in the *optimal* mutation rates, which minimize the expected time needed to optimize OneMax. Much easier to compute than the optimal mutation rates are the *drift-maximizing ones*, i.e., the values which maximize the expected gain $\mathbb{E}[f(y)-f(x) \mid y \leftarrow \mathtt{flip}_k(x), k \sim D(n,f(x))]$, see Sec. 3.

Notational Convention. We omit the explicit mention of $(1+\lambda)$ when the value of λ is clear from the context. Also, formally, we should distinguish between the *mutation rate* (used by the EAs, see above) and the *mutation strengths* (i.e., the number of bits that are flipped). However, to ease presentation, we will just speak of *mutation rates* even when referring to the parameter setting for RLS.

3 Computation of Optimal Parameter Configurations

We compute the optimal parameters using the similar flavor of dynamic programming that has already been exploited in [4]. Namely, we compute the optimal parameters and the corresponding remaining time expectations for Hamming distance d to the optimum after we have computed them for all smaller distances $d' < d$. We denote by $T^*_{D,O}(n,\lambda,d)$ the minimal expected remaining time of a $(1+\lambda)$ algorithm with mutation rate distribution $D \in \{\mathrm{RLS}, \mathrm{sbm}, 0\to1\}$, optimality criterion $O \in \{\mathrm{opt}, \mathrm{drift}\}$, and population size λ on a problem size $n \in \mathbb{N}$ when at distance $d \in [0..n]$. We also denote the distribution parameter (mutation strength or rate) by ρ, and the optimal distribution parameter for the current context as $\rho^*_{D,O}(n,\lambda,d)$.

Let $P_{n,D}(d,d',\rho)$ be the probability of sampling an offspring at distance d' to the optimum, provided the parent is at distance d, the problem size is n, the distribution function is D, and the distribution parameter is ρ. The expected remaining time $T_{D,O}(n,\lambda,d,\rho)$, which assumes that at distance d the algorithm consistently uses parameter ρ and at all smaller distances it uses the optimal (time-minimizing or drift-maximizing, respectively) parameter for that distance, is then computed as follows:

$$T_{D,O}(n,\lambda,d,\rho) = \frac{1}{(1-P_{n,D}(d,d,\rho))^\lambda} + \sum\nolimits_{d'=1}^{d-1} T^*_{D,O}(n,\lambda,d') \cdot P^\lambda_{n,D}(d,d',\rho), \tag{1}$$

where $P^\lambda_{n,D}(d,d',\rho) = \left(\sum_{t=d'}^{d} P_{n,D}(d,t,\rho)\right)^\lambda - \left(\sum_{t=d'+1}^{d} P_{n,D}(d,t,\rho)\right)^\lambda$.

To compute $T^*_{D,O}(n,\lambda,d)$, Eq. (1) is used, where direct minimization of ρ is performed when $O=\mathrm{opt}$, and the following drift-maximizing value of ρ is substituted when $O=\mathrm{drift}$: $\rho_{n,D}(d) = \arg\max_\rho \sum_{d'=0}^{d-1}(d-d') \cdot P^\lambda_{n,D}(d,d',\rho)$.

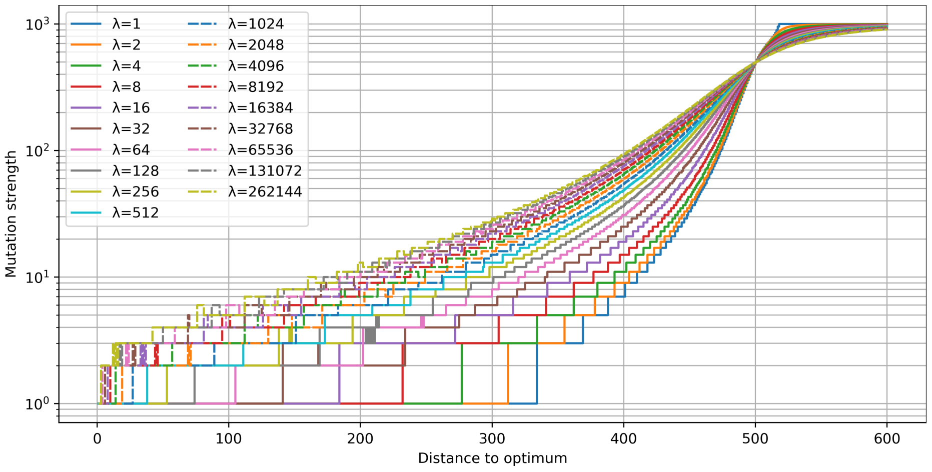

Fig. 1. Optimal parameters $\rho^*_{\text{RLS,opt}}(n, \lambda, d)$ for different values of λ and $n = 1000$ as a function of d, the distance to the optimum

Another difference to the work of [4] is in that we do not only compute the expected remaining running times $T^*_{D,O}(n, \lambda, d)$ when using the optimal mutation rates $\rho^*_{D,O}(n, \lambda, d)$, but we also compute and store $T_{D,O}(n, \lambda, d, \rho)$ for suboptimal values of ρ. For RLS we do that for all possible values of ρ, which are integers not exceeding n, while for the $(1 + \lambda)$ EA we consider $\rho = 2^{i/5-10}/n$ for all $i \in [0; 150]$. We do this not only because it gives us additional insight into the sensitivity of $T_{D,O}(n, \lambda, d, \rho)$ with respect to ρ, but it also offers a convenient way to detect deficits of parameter control mechanism; see Sec. 6 for an illustrated example. Since our data base is hence much more detailed than that of [4], we also re-consider the case $\lambda = 1$.

Our code has the $\Theta(n^3)$ runtime and $\Theta(n^2)$ memory complexity. The code is available on GitHub [6], whereas the generated data is available on Zenodo [5].

4 Optimal Mutation Rates and Optimal Running Times

Fig. 1 plots the optimal parameter settings $\rho^*_{\text{RLS,opt}}(n, \lambda, d)$ for fixed dimension $n = 10^3$ and for different values of λ, in dependence of the Hamming distance d to the optimum. We observe that the mutation strengths $\rho^*_{\text{RLS,opt}}(n, \lambda, d)$ are nearly monotonically increasing in λ, as a result of having more trials to generate an offspring with large fitness gain. We also see that, for some values of λ, the curves are not monotonically decreasing in d, but show small “bumps”. Similar non-monotonic behavior can also be observed for drift-maximizing mutation strengths $\rho^*_{\text{RLS,drift}}(n, \lambda, d)$, as can be seen in Fig. 2.

We show now that these “bumps” are not just numeric precision artifacts, but rather a (quite surprising) feature of the parameter landscape. For a small example that can be computed by a human we consider $n = 30$ and $\lambda = 512$. For

Table 1. Drifts for $n = 30$, $\lambda = 512$, $d = 7, 8$, $\rho \in [1..10]$.

d	$\rho = 1$	2	3	4	5	6	7	8	9	10
7	0.5000	2.0000	2.9762	2.9604	**3.0434**	2.7009	2.5766	2.2292	1.7457	1.3854
8	0.5000	2.0000	2.9984	**3.4601**	3.3583	3.3737	3.2292	2.9124	2.7323	2.3445

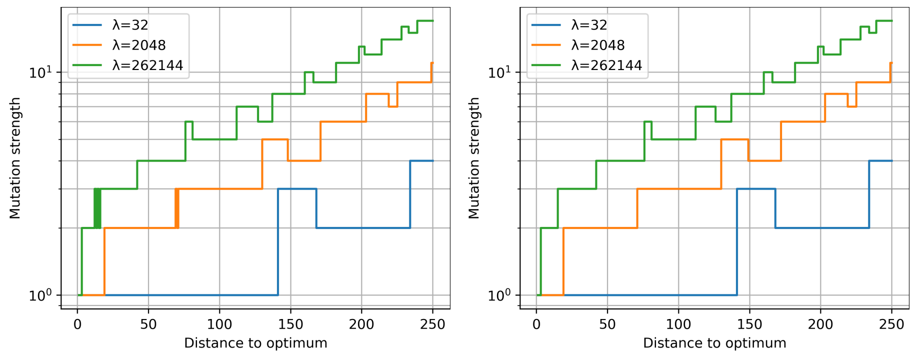

Fig. 2. Non-monotonicity in optimal (left) and drift-optimal (right) mutation strengths for $n = 1000$ and selected λ

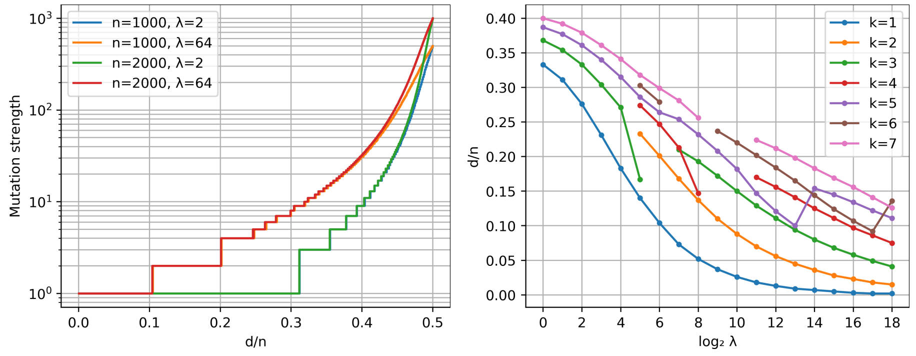

Fig. 3. Left: $\rho^*_{\mathrm{RLS,opt}}(n, \lambda, d)$ for $\lambda \in \{2, 64\}$ and $n \in \{1k, 2k\}$, in dependence of d/n. **Right:** Normalized maximal distance d/n at which flipping $k \in [1..7]$ bits is optimal for RLS, for $n = 10^3$ and $\lambda \in \{2^i \mid 0 \le i \le 18\}$.

$d = 7$ and 8, we compute the drifts for mutation strengths in [1..10]. These values are summarized in Table 1. Here we see that the drift-maximizing mutation for $d = 7$ is 5, whereas for $d = 8$ it is 4. This example, in fact, serves two purposes: first, it shows that even the drift-maximizing strengths can be non-monotone, and second, that the drift-maximizing strengths can be even for non-trivial problem sizes, which – as mentioned in the introduction – cannot be the case when $\lambda = 1$ [10].

In the left chart of Fig. 3 we show that at least small $\rho^*_{\mathrm{RLS,opt}}(n, \lambda, d)$ are quite robust with respect to the problem dimension $n \in \{1, 2\} \cdot 10^3$, if the Hamming

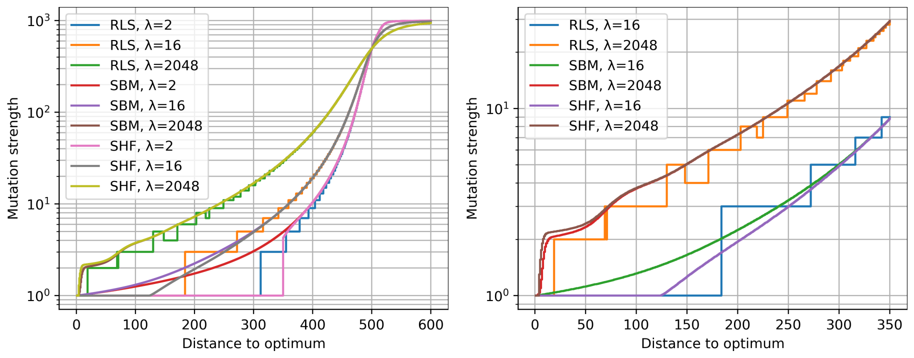

Fig. 4. Left: Expected mutation strengths of the time-minimizing parameter settings for the $(1+\lambda)$ RLS and two $(1+\lambda)$ EAs, $\lambda \in \{2, 16, 2048\}$, using standard bit mutation (SBM) and shift mutation (SHF), respectively. Values are for $n = 1000$ and plotted against the Hamming distance to the optimal solution. **Right:** Same for $\lambda \in \{16, 2048\}$ with an emphasis on small distances.

distance d to the optimum is appropriately scaled as d/n. The chart plots the curves for $\lambda \in \{2, 64\}$ only, but the observation applies to all tested values of λ. In accordance to our previous notes, we also see that for $\lambda = 64$ there is a regime for which flipping two bits is optimal. For small population sizes λ, we also obtain even numbers for certain regimes, but only for much larger distances.

The maximal distances at which flipping k bits is optimal are summarized in the chart on the right of Fig. 3. Note here that the curves are less smooth than one might have expected. For instance, for $n = 10^3$, flipping three bits is never optimal for $\lambda = 64$, and flipping seven bits is never optimal for $\lambda = 2^9$ and 2^{10}.

In Fig. 4 we compare the optimal (i.e., time-minimizing) parameter settings of the $(1+\lambda)$ variants of RLS, the $\text{EA}_{0\to 1}$, and the EA_{sbm}. To obtain a proper comparison, we compare the mutation strength $\rho^*_{\text{RLS,opt}}(n, \lambda, d)$ with the expected number of bits that flip in the two EA variants, i.e., $n\rho^*_{\text{sbm,opt}}(n, \lambda, d)$ for the EA using standard bit mutation and $n\rho^*_{\text{sbm,opt}}(n, \lambda, d) + (1 - \rho^*_{\text{sbm,opt}}(n, \lambda, d))^n$ for the EA using the shift mutation operator. We show here only values for $\lambda \in \{2, 16, 1024\}$, but the picture is similar for all evaluated λ.
We observe that, for each λ, the curves are close together. While for $\lambda = 1$ the curves for standard bit mutation were always below that of RLS, we see here that this picture changes with increasing λ. We also see a sudden decrease in the expected mutation strength of the shift operator when λ is small. In fact, it is surprising to see that, for $\lambda = 2$, the value drops from around 5.9 at distance 373 to 1 at distance 372. This is particularly interesting in light of a common tendency in state-of-the-art parameter control mechanisms to allow only for small parameter updates between two consecutive iterations. This is the case, for example, in the well-known one-fifth success rule [8, 23, 24]. Parameter control techniques building on the family of reinforcement learning algorithms (see [17] for examples) might catch such drastic changes more efficiently.

Non-surprisingly, the expected mutation strengths of the optimal standard bit mutation rate and the optimal shift mutation rate converge as the distance to the optimum increases.

5 Sensitivity of the Optimization Time w.r.t the Parameter Settings

In this section, we present our findings on the sensitivity of the considered $(1+\lambda)$ algorithms to their mutation parameters. To do this, we use not only the expected remaining times $T^*_{D,O}(n, \lambda, d)$ that correspond to optimal parameter values, but also $T_{D,O}(n, \lambda, d, \rho)$ for various parameter values ρ, which correspond to the situation when an algorithm uses the parameter ρ while it remains at distance d, and switches to using the optimal parameter values (time-minimizing for $O = \text{opt}$ and drift-maximizing for $O = \text{drift}$, respectively) once the distance is improved. For reasons of space we focus on $O = \text{opt}$.

We use distance-versus-parameter heatmaps as a means to show which parameter values are efficient. As the non-optimality regret $\delta_{D,O}(n, \lambda, d, \rho) = T_{D,O}(n, \lambda, d, \rho) - T^*_{D,O}(n, \lambda, d)$ is asymptotically smaller than the remaining time, we derive the color from the value $\tau(\rho) = \exp(-\delta_{D,O}(n, \lambda, d, \rho))$. Note that $\tau(\rho) \in (0; 1]$, and the values close to one represent parameters that are almost optimal by their effect. The parameters where $\tau(\rho) \approx 0.5$, on the other hand, correspond to a regret of roughly 0.7, that is, if the parameters satisfy $\tau(\rho) \geq 0.5$ throughout the entire optimization, the total expected running time is greater by at most $0.7n/2$ than the optimal time for this type of algorithms.

Fig. 5 depicts these regrets for RLS_{opt} on $n = 10^3$ and $\lambda \in \{1, 512\}$. The stripes on the fine-grained plot for $\lambda = 1$ expectedly indicate, as in [4], that flipping an even number of bits is generally non-optimal when the distance to the optimum is small, which is the most pronounced for $\rho = 2$. This also indicates that the parameter landscape of RLS is multimodal, posing another difficulty to parameter control methods. The parameter-time landscape remains multimodal for $\lambda = 512$, but the picture is now much smoother around the optimal parameter values.

Fig. 6 plots the regret for the $(1 + \lambda)$ EA_{sbm} (top) and the $(1 + \lambda)$ $\text{EA}_{0\rightarrow 1}$ (bottom) with $\lambda = 1$ (left) and $\lambda = 512$ (right). The pictures for standard and shift mutations are very similar until the distance is so small that one-bit flips become nearly optimal. We also see that bigger population sizes result in a lower sensitivity of the expected remaining optimization time with respect to the mutation rate. In fact, we see that, even for standard bit mutation, parameter settings that are much smaller than the typically recommended mutation rates (e.g., $\rho = 1/(10n)$) are also good enough when the distance is $\Omega(n)$, as the probability to flip at least one bit at least once is still quite large.

The plot for the $(1+1)$ $\text{EA}_{0\rightarrow 1}$ deserves separate attention. Unlike other plots in Fig. 6, it shows a bimodal behavior with respect to the mutation probability ρ even for quite large distances $d < n/2$. We zoom into this effect by displaying in Fig. 7 the expected remaining optimization times for $d \in \{370, 376\}$.

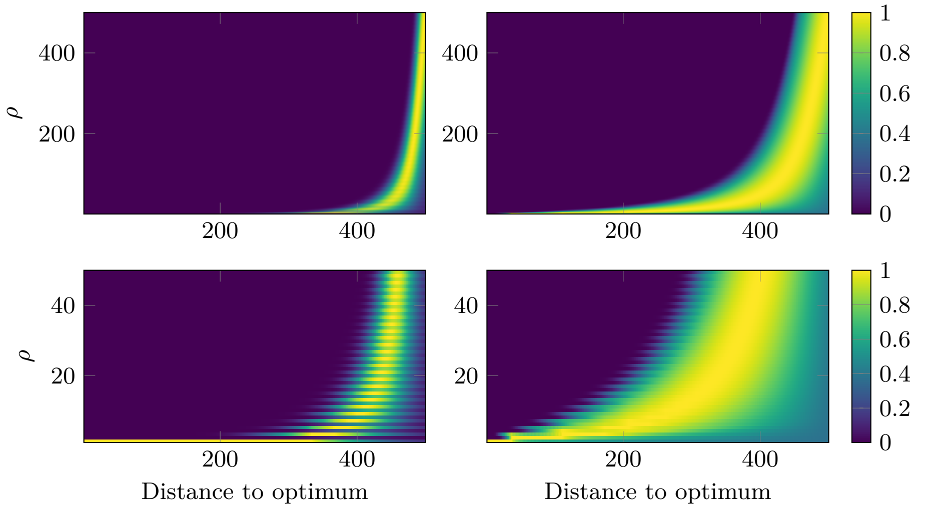

Fig. 5. Relative expected remaining optimization times for the $(1+\lambda)$ RLS$_{\text{opt}}$ with parameters $n = 10^3$, $\lambda = 1$ (left) and $\lambda = 512$ (right). The first row displays the general picture, the second row focuses on small mutation strengths

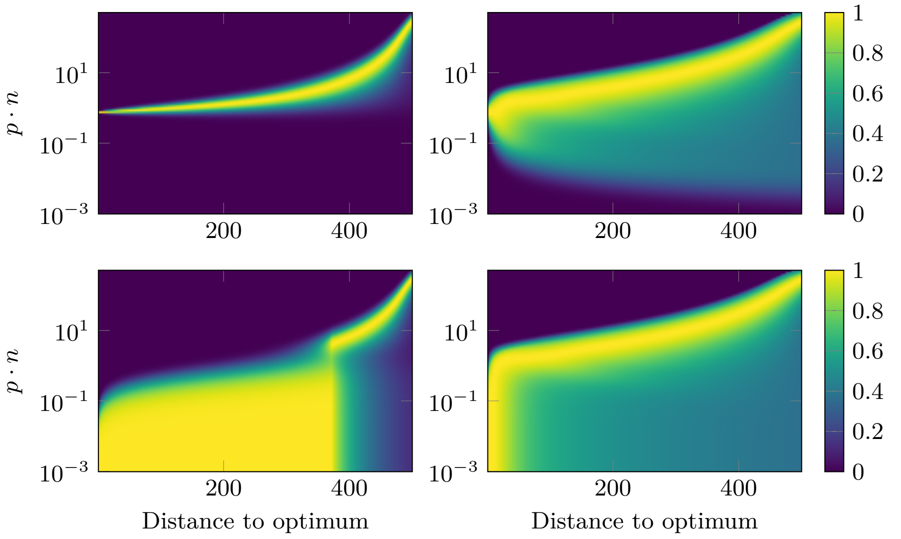

Fig. 6. Relative expected remaining optimization times for the $(1+\lambda)$ EA$_{\text{sbm,opt}}$ (top) and the $(1+\lambda)$ EA$_{0\rightarrow 1,\text{opt}}$ (bottom) with $\lambda = 1$ (left) and $\lambda = 512$ (right)

Drift-Maximization vs. Time-Minimization. We note, without diving into the details, that the observation that the optimal mutation parameters are not identical to the drift maximizing ones, made in [4] for $(1+1)$ algorithms,

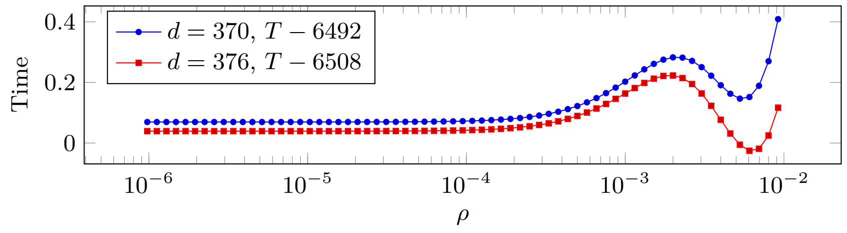

Fig. 7. Expected remaining optimization time of the $(1+\lambda)$ $\text{EA}_{0\to 1,\text{opt}}$ as a function of the mutation probability ρ

extends to $(1+\lambda)$-type algorithms with $\lambda > 1$. More precisely, it applies to all tested dimensions and population sizes λ. We note, though, that the disadvantage $T^*_{\text{RLS,drift}}(n,\lambda,d) - T^*_{\text{RLS,opt}}(n,\lambda,d)$ decreases with increasing λ. Since the difference is already quite small for the case $\lambda = 1$ (e.g., for $n = 1000$, it is 0.242), we conclude that this difference, albeit interesting from a mathematical perspective, has very limited relevance in empirical evaluations. This is good news for automated algorithm configuration techniques, as it implies that simple regret (e.g., in the terms of one-step progress) is sufficient to derive reasonable parameter values – as opposed to requiring cumulative regret, which, as Sec. 3 shows, is much more difficult to track.

6 Applications in Parameter Control

Fig. 8 displays the experimentally measured mean optimization times, averaged over 100 runs, of (1) the standard $(1+\lambda)$ EA with static mutation rate $\rho = 1/n$, (2) RLS_{opt}, (3) the $(1+\lambda)$ $EA_{0\to 1,\text{opt}}$, and of (4-5) the “two-rate” parameter control mechanism suggested in [11], superposed here to the $(1+\lambda)$ $\text{EA}_{0\to 1}$ with two different lower bounds $\rho_{\min}$ at which the mutation rate is capped.

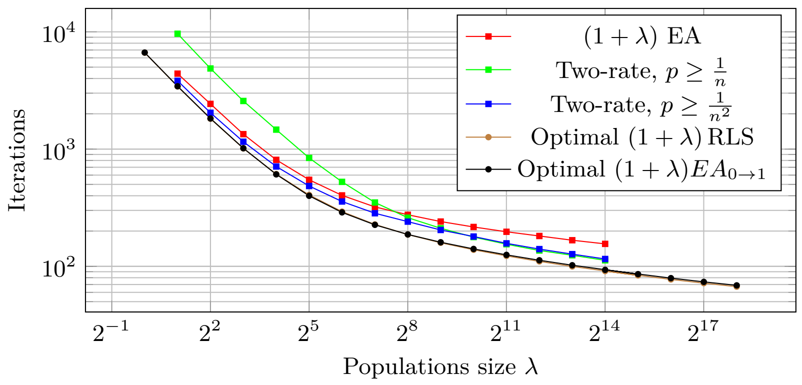

Fig. 8. Mean number of iterations of different $(1+\lambda)$ EAs vs. the expected number of iterations of RLS_{opt} and $EA_{\text{opt},0\to 1}$ for $n = 10^3$, as a function of the population size λ

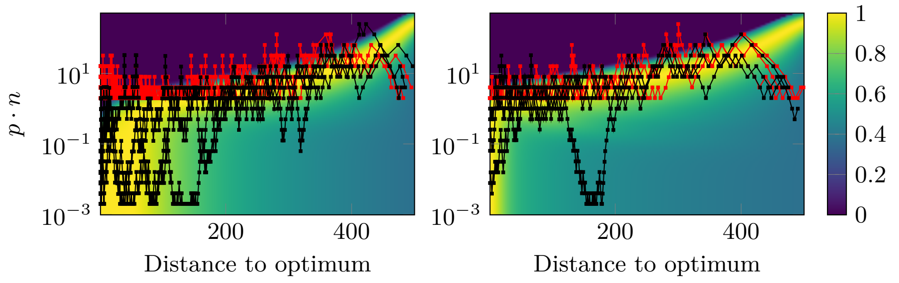

Fig. 9. Parameter control plots of the two-rate method atop parameter efficiency heatmaps, $n = 10^3$, $\lambda = 64$ (left) and $\lambda = 2048$ (right). Red traces are for the mutation rate lower bound of $\rho_{\min} = 1/n$, black traces are for the lower bound of $\rho_{\min} = 1/n^2$

With such pictures, we can infer how far a certain algorithm is from an optimally tuned algorithm with the same structure, which can highlight its strengths and weaknesses. However, it is difficult to derive insights from just expected times. To get more information, one can record the parameter values produced by the investigated parameter control method and draw them atop the heatmaps produced as in Sec. 5. An example of this is shown in Fig. 9. An insight from this figure, that may be relevant to the analysis of strengths and weaknesses of this method, would be that the version using $\rho_{\min} = 1/n$ cannot use very small probabilities and is thus suboptimal at distances close to the optimum, whereas the version using $\rho_{\min} = 1/n^2$ falls down from the optimal parameter region too frequently and too deep.

7 Conclusions

Extending the work [4], we have presented in this work optimal and drift-maximizing mutation rates for two different $(1+\lambda)$ EAs and for the $(1+\lambda)$ RLS. We have demonstrated how our data can be used to detect weak spots of parameter control mechanisms. We have also described two unexpected effects of the dependency of the expected remaining optimization time on the mutation rates: non-monotonicity in d (Sec. 4) and non-unimodality (Sec. 5). We plan on exploring these effects in more detail, and with mathematical rigor. Likewise, we plan on analyzing the formal relationship of the optimal mutation rates with the normalized distance d/n. As a first step towards this goal, we will use the numerical data presented above to derive close-form expressions for the expected remaining optimization times $T_{D,O}(n, \lambda, d, \rho)$ as well as for the optimal configurations $\rho^*_{D,O}(n, \lambda, d)$. Finally, we also plan on applying similar analyses to more sophisticated benchmark problems.

The extended version of the paper is available on arXiv [14].

Acknowledgments. The work was supported by RFBR and CNRS, project no. 20-51-15009, by the Paris Ile-de-France Region, and by ANR-11-LABX-0056-LMH.

References

1. Auger, A., Hansen, N.: Theory of evolution strategies: A new perspective. In: Theory of Randomized Search Heuristics: Foundations and Recent Developments, pp. 289–325. World Scientific (2011).
2. Bäck, T.: The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In: Proc. of Parallel Problem Solving from Nature (PPSN II). pp. 87–96. Elsevier (1992)
3. Badkobeh, G., Lehre, P.K., Sudholt, D.: Unbiased black-box complexity of parallel search. In: Proc. of Parallel Problem Solving from Nature (PPSN XIII). LNCS, vol. 8672, pp. 892–901. Springer (2014).
4. Buskulic, N., Doerr, C.: Maximizing drift is not optimal for solving OneMax. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), Companion Material. pp. 425–426. ACM (2019). , full version available at https://arxiv.org/abs/1904.07818
5. Buzdalov, M.: Data for "Optimal mutation rates for the $(1 + \lambda)$ EA on OneMax". https://doi.org/10.5281/zenodo.3897351 (2020)
6. Buzdalov, M.: Repository with the code to compute the optimal rates and the expected remaining optimization times. https://github.com/mbuzdalov/one-plus-lambda-on-onemax/releases/tag/v1.0 (2020)
7. Carvalho Pinto, E., Doerr, C.: Towards a more practice-aware runtime analysis of evolutionary algorithms (2018), https://arxiv.org/abs/1812.00493
8. Devroye, L.: The compound random search. Ph.D. dissertation, Purdue Univ., West Lafayette, IN (1972)
9. Doerr, B., Doerr, C.: Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices. In: Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 271–321. Springer (2020), also available online at https://arxiv.org/abs/1804.05650v2
10. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. Theor. Comput. Sci. **801**, 1–34 (2020).
11. Doerr, B., Gießen, C., Witt, C., Yang, J.: The $(1+\lambda)$ evolutionary algorithm with self-adjusting mutation rate. Algorithmica **81**(2), 593–631 (2019)
12. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation—Recent Developments in Discrete Optimization. Springer (2020)
13. Doerr, B., Witt, C., Yang, J.: Runtime analysis for self-adaptive mutation rates. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'18). pp. 1475–1482. ACM (2018).
14. Doerr, C., Buzdalov, M.: Optimal mutation rates for the $(1 + \lambda)$ EA on OneMax. https://arxiv.org/abs/2006.11457 (2020)
15. Erdős, P., Rényi, A.: On two problems of information theory. Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei **8**, 229–243 (1963)
16. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In: Proc. of Learning and Intelligent Optimization (LION'09). LNCS, vol. 5851, pp. 176–190. Springer (2009).
17. Fialho, Á., Costa, L.D., Schoenauer, M., Sebag, M.: Analyzing bandit-based adaptive operator selection mechanisms. Ann. Math. Artif. Intell. **60**(1-2), 25–64 (2010).

18. Finck, S., Hansen, N., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions. http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf (2010)
19. Gießen, C., Witt, C.: The interplay of population size and mutation probability in the $(1 + \lambda)$ EA on OneMax. Algorithmica **78**(2), 587–609 (2017)
20. Gießen, C., Witt, C.: Optimal mutation rates for the $(1 + \lambda)$ EA on OneMax through asymptotically tight drift analysis. Algorithmica **80**(5), 1710–1731 (2018)
21. Jansen, T., Jong, K.A.D., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. Evolutionary Computation **13**(4), 413–440 (2005)
22. Karafotias, G., Hoogendoorn, M., Eiben, Á.E.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Transactions on Evolutionary Computation **19**(2), 167–187 (2015)
23. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboorg Verlag, Stuttgart (1973)
24. Schumer, M.A., Steiglitz, K.: Adaptive step size random search. IEEE Transactions on Automatic Control **13**, 270–276 (1968)
25. Thierens, D.: On benchmark properties for adaptive operator selection. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'09), Companion Material. pp. 2217–2218. ACM (2009).