Sensitivity of Parameter Control Mechanisms with Respect to Their Initialization

Carola Doerr[1] and Markus Wagner[2]

[1] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, 75005 Paris, France
[2] Optimisation and Logistics, University of Adelaide, Adelaide, SA 5005, Australia

Abstract. The parameter setting problem constitutes one of the major challenges in evolutionary computation, and is subject to considerable research effort. It is widely acknowledged that the optimal parameter values can change during the optimization process. Efficient parameter control techniques that automatically identify and track reasonable parameter values can therefore bring substantial performance gains.
One complication of dynamic parameter selection is the fact that every control mechanism itself introduces a new set of *hyper-parameters*, which needs to be tuned for the problem at hand. The general hope is that the performance of an algorithm is much less sensitive with respect to its hyper-parameters than with respect to its original parameters. For the parameter update rules this belief is backed up by a number of empirical and theoretical results. What is less understood in discrete black-box optimization, however, is the influence of the initial parameter value. We contribute with this work an empirical sensitivity analysis for three selected algorithms with self-adjusting parameter choices: the (1+1) EA$_\alpha$, the 2-rate $(1+\lambda)$ EA$_{2r,r/2}$, and the $(1+(\lambda,\lambda))$ GA.

Keywords: Parameter Control · Evolutionary Algorithms · Discrete Black-Box Optimization · Initialization.

1 Introduction

Every evolutionary algorithm (EA) and, more generally, every discrete black-box optimization heuristic, comes with a set of parameters that needs to be set in order to run it. Among the most influential parameters are the population sizes, the mutation rates, the crossover probabilities, and the selective pressure. The choice of any of these parameters can have a significant impact on the performance of the EA under consideration. It is therefore not surprising that the parameter selection question has evolved into an important research stream within the evolutionary computation community, cf. [17] for detailed discussions.

The last forty years of research on the parameter setting problem have contributed to a significant gain in performance, and have been a major building block for the success of evolutionary computation methods. According to the seminal work of Eiben, Hinterding, and Michalewicz [12] the parameter setting literature can be classified into two main research streams:

- *Parameter tuning* addresses the question how to efficiently identify good parameter values through an initial set of experiments. After their identification, these parameter values are not further adjusted during the optimization process, but remain fixed instead. Among the most-widely applied tools for parameter tuning are irace [18], SPOT [3], SMAC [14], ParamILS [15], and GGA [2].
- *Parameter control,* in contrast, studies ways to adjust ("control") the parameter values during the run of the optimization, to benefit from an adaptation to the different stages of the optimization process. Using such non-static parameter values, the EAs can, for example, evolve from a rather exploratory globally acting heuristic to a more and more locally exploiting one. Among the best-known parameter control techniques are the step size and covariance matrix adaptation in the CMA-ES [13] and variants of the 1/5-th success rule [5, 19, 20].

The focus of our work is on parameter control for discrete black-box optimization, a topic that has been somewhat neglected in the evolutionary computation community: a quote of [16, Section VIII] says that "controlling EA parameters on-the-fly is still a rather esoteric option". A potential reason for this situation may be the common critique that parameter control mechanisms add yet another level of complexity to the algorithms. The influence of the parameter control mechanisms are indeed difficult to grasp analytically, so that only few theoretical works addressing the parameter control question exist [7]. A related critique of parameter control is the fact that on-the-fly parameter selection techniques come with their own *hyper-parameters*, which need to be set to determine the exact update rules. From a high-level perspective one may feel that not much can be gained by replacing a parameter by one or more hyper-parameters, but the general hope is that the influence of these hyper-parameters is much less important than that of the original parameter values. Several studies confirm this hope for some specific settings, empirically as well as in rigorous mathematical terms, cf. the surveys [1, 7, 12, 16] and references therein.

1.1 Our Contribution

Complementing our recent work on the sensitivity of parameter control mechanisms with respect to their choice of the hyper-parameters that determine the update strength [11], we consider in this study their sensitivity with respect to initialization. More precisely, we analyze for three different EAs with self-adjusting parameter selection the influence of the initial value on the performance: the (1+1) EA$_\alpha$ proposed in [11], the 2-rate $(1+\lambda)$ EA$_{2r,r/2}$ from [10], and the $(1+(\lambda,\lambda))$ GA [6, 8]. In the first two algorithms the mutation rate is controlled by a success-based update rule. In the $(1+(\lambda,\lambda))$ GA the adaptation of λ influences the offspring population sizes, the mutation rate, and the crossover bias, cf. Section 3. For all three algorithms we test the influence of extreme initialization, i.e., $p = 1/n$ vs. $p = 1/2$ for the $(1+1)$ EA$_\alpha$ and the $(1+\lambda)$ EA$_{2r,r/2}$, and $\lambda = 1$ vs. $\lambda = n$ for the $(1+(\lambda,\lambda))$ GA. Our selection is clearly theory-biased, i.e., we favor those algorithms for which mathematical analyses of their running time behavior are available. This allows us to chose update mechanisms which are known to be (close to) optimal, so that our sensitivity analysis of the initial parameter values is not biased by a non-sensible choice of hyper-parameters.

Our testbed are the well-known OneMax and LeadingOnes benchmark functions, again with the motivation to not bias the result by a non-suitable control mechanism, and to allow for a comparison with known optimal dynamic parameter values. The OneMax problem is the problem of maximizing a function of the type $\text{Om}_z : \{0,1\}^n \to [0..n], x \mapsto |\{i \in [n] \mid x_i = z_i\}|$, where $z \in \{0,1\}^n$ is of course unknown to the algorithm. While OneMax is a separable function, and thus in general easy to hill climb for greedy algorithms, the LeadingOnes problem is non-separable, and requires a quadratic number of function evaluations, on average, by standard evolutionary algorithms. The LeadingOnes problem is the problem of optimizing functions of the type $\text{Lo}_{z,\sigma} : \{0,1\}^n \to \mathbb{N}, x \mapsto \max\{i \in [0..n] \mid \forall j \in [i] : x_{\sigma(j)} = z_{\sigma(j)}\}$, where z is a length-n bit string and $\sigma : [n] \to [n]$ a permutation of the index set $[n]$. We acknowledge that these simplified benchmark problems may not be very representative for real-world optimization challenges. In accordance with [21] we nevertheless believe that they can test several important features of reasonable parameter control mechanisms, and give first indications into which update schemes to favor under which conditions.

Our results indicate quite stable performances for the $(1+1)$ EA_α and the $(1+\lambda)$ $\text{EA}_{2r,r/2}$. Even when initialized with extreme mutation rates, the dynamic choice very quickly converges to optimal mutation rates and the incurred performance loss of a sub-optimal initialization is small. The situation is different for the $(1+(\lambda,\lambda))$ GA. The number of function evaluations grows linearly with the value of λ (more precisely, up to 2λ offspring are evaluated per iteration), a cost that the additional drift towards the optimum cannot compensate for. This situation of a too large λ value does not last very long, as we observe again fast convergence of the parameter towards its optimal (dynamic) setting. It nevertheless causes significant and non-negligible performance losses: for the 1000-dimensional OneMax problem, for example, the worst initialization $\lambda = n$ yields a performance that is around 69% worse compared to that of the optimal initial parameter choice $\lambda = 1$.

2 Sensitivity Analysis for the (1+1) EA_α

In [11] we have presented a $(1+1)$ EA variant with success-based multiplicative mutation rate updates, the $(1+1)$ EA_α. This algorithm starts the optimization process with a random initial solution and an initial mutation rate $p = p_0 \in (0, 1/2]$. In every iteration one new solution candidate is created from the current-best solution through a conditional standard bit mutation with mutation rate p. The condition requires that at least one bit is changed, to avoid useless function evaluations. In practice, this conditional mutation operator can be implemented by first sampling a number ℓ from the conditional binomial distribution $\text{Bin}_{>0}(n,p)$ and then choosing uniformly at random and without replacement the ℓ positions in which the bits are flipped. If the so-created offspring is at least as good as its predecessor, it replaces the latter. In this case the mutation rate is increased to $\min\{Ap, 1/2\}$, where $A > 1$ is a constant that remains fixed during the execution of the algorithm. If, on the other hand, the offspring is strictly worse than its parent, it is discarded and the mutation rate decreased to $\max\{bp, 1/n^2\}$, where $0 < b < 1$ is another constant.

Altogether, the (1+1) EA_α has three *hyper-parameters*: the update strengths A and b as well as the initial mutation rate p_0. It was demonstrated in [11] that the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$ performs very well on the classic benchmark functions OneMax and LeadingOnes for a broad choice of values for A and b. For example, in 78% of all tested combinations of $A \in (1, 2.5]$ and $b \in [0.4, 1)$ the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$ achieved a better average running time than RLS (!) on the 250-dimensional LeadingOnes function. About 90% of the configurations outperform the (1+1) $\text{EA}_{>0}$ (which is the $(1+1)$ $\text{EA}_\alpha(1, 1, 1/n)$) on the 1000-dimensional OneMax function. We will analyze how sensitive this performance is with respect to the choice of the initial mutation rate p_0.

2.1 Optimal Mutation Rates for OneMax and LeadingOnes

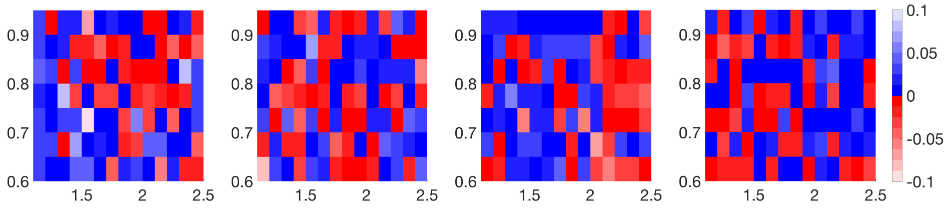
Before we present our empirical findings, we summarize in this section what is known on the optimal mutation rates for OneMax and LeadingOnes.

OneMax. In [9] it was shown that the RLS variant flipping in every step the number of bits that maximizes the expected progress cannot be significantly worse than the best unary unbiased algorithm, which is the one minimizing in every step the expected remaining running time. Denoting by $k_{\text{opt,OM}}(n, \text{Om}(x))$ the choice that maximizes the expected Om-progress $\mathbb{E}\Big[\max\{\text{Om}(\text{mut}_\ell(x)) - \text{Om}(x), 0\}\Big] := \sum_{i=\lceil \ell/2 \rceil}^{\ell} \frac{\binom{n-\text{Om}(x)}{i}\binom{\text{Om}(x)}{\ell-i}(2i-\ell)}{\binom{n}{\ell}}$ of flipping ℓ bits in bit string x, the following is known. $k_{\text{opt,OM}}(n, \text{Om}(x))$ decreases monotonically with increasing function value $\text{Om}(x)$. It is $n/2$ for $\text{Om}(x) = n/2$ and converges to 1. Flipping one bit per iteration is optimal for all x with $\text{Om}(x) \geq n/3$.

The expected OneMax value of a random initial solution x is $n/2$ for OneMax, and with high probability $\text{Om}(x)$ lies in the interval $[n/2 \pm \sqrt{n}]$. The exact average optimal mutation rate is $\sum_{i=1}^{n} \mathbb{P}[\text{Om}(x) = i] k_{\text{opt,OM}}(n, i)$. We do not have any closed form for the drift maximizing value $k_{\text{opt,OM}}(n, i)$, but we can evaluate this expression numerically. For $n = 1000$ the sum evaluates to 500.0252.

LeadingOnes. For LeadingOnes the situation is much better understood. The optimal mutation rate of the classic (non-resampling) $(1+1)$ EA is $1/(\text{Lo}(x)+1)$ [4] and the optimal number of bits to flip is $k_{\text{opt,LO}}(n, \text{Lo}(x)) := \lfloor n/(\text{Lo}(x)+1) \rfloor$ [11, Lemma 1].

The expected LeadingOnes value of the random initial solution x is $\sum_{i=0}^{n} i\mathbb{P}[\text{Lo}(x) = i] = \sum_{i=0}^{n} i2^{-(i+1)} = 1 - 1/2^n$, and the average optimal initial



(a) OM $n = 1000$ (b) OM $n = 1500$ (c) LO $n = 100$ (d) LO $n = 250$

Fig. 1: Relative difference $(T(A,b,1/n)-T(A,b,1/2))/T(A,b,1/n)$ of the average running time for 120 configurations of the $(1+1)$ EA$_\alpha$ with $1 < A \leq 2.5$ and $0.6 \leq b < 1$

mutation rate equals $\sum_{i=0}^{n} k_{\text{opt,LO}}(n,i)\mathbb{P}[\text{LO}(x) = i] = \sum_{i=0}^{n} \lfloor n/(i+1)\rfloor 2^{-(i+1)}$. For $n = 100$ (250, 1,000) this value is around 69 (173, 693).

2.2 Evaluating the Relative Average Improvement

In light of the discussion in Section 2.1, one might wonder if significant gains are possible for the $(1+1)$ EA$_\alpha$ when the mutation rate is initialized as $p_0 = 1/2$. As a first step towards analyzing the sensitivity of the $(1+1)$ EA$_\alpha(A,b,p_0)$ with respect to this initialization, we compute for each of the 120 configurations with $A \in \{1.1, 1.2, \ldots, 2.5\}$ and $b \in \{0.6, 0.65, \ldots, 0.95\}$ the average optimization time of 101 independent runs of the $(1+1)$ EA$_\alpha(A,b,1/2)$. We compare this average value to that of the same configuration (A,b) for $p_0 = 1/n$, and we compute the relative gain of the $p_0 = 1/2$ initialization. That is, denoting by $T(A,b,p_0)$ the average optimization time of the $(1+1)$ EA$_\alpha(A,b)$ with initialization p_0, we calculate for each configuration (A,b) the value $(T(A,b,1/n)-T(A,b,1/2))/T(A,b,1/n)$. This data is displayed in the heatmaps of Figure 1 for 1000- and the 1500-dimensional OneMax problem and the 100- and 250-dimensional LeadingOnes problem, respectively. We observe that the data is rather unstructured, and that a good relative gain in one dimension does typically not apply to the other.

The relative gains range from a negative -10% (-8%) to a positive 8% (7%) improvement for OneMax of dimension $n = 1000$ ($n = 1,500$), and from -7% (-4%) to 5% (4%) for the 100-(250-)dimensional LeadingOnes problem. Note that here the relatively low number of repetitions has to be taken into account. The average gain of the $p_0 = 1/2$ initialization over the $p_0 = 1$ initialization in all 120 (A,b) configurations is about 0.17% (0.21%) for the OneMax problem of dimension $n = 1000$ ($n = 1500$) and is about -0.13% (-0.05%) for LeadingOnes in dimension $n = 100$ ($n = 250$). These small values indicate that the influence of the initial parameter value is not very important. It may be surprising that the average gain is negative for the LeadingOnes problem, but we suspect that this is an effect of the problem size that may vanish in larger dimension.

2.3 Testing for Statistical Significance

While the results displayed in the heatmaps do not suggest that we should expect *important* performance gains from a better initialization, this data does not answer the question whether the (dis-)advantages are *statistically significant*. We therefore investigate a few selected configurations in more detail, and use the Wilcoxon rank-sum tests to test for significance. Precisely, we run each of the four selected configurations $(A = 1.2, b = 0.85)$, $(1.3, 0.75)$, $(2.0, 0.5)$, and $(1.11, 0.66)$ investigated in [11] $1,001$ independent times on the OneMax problem of dimension $n \in \{500, 1000, 2000\}$ and on the LeadingOnes problem of dimensions $n \in \{100, 250, 500\}$. For each (configuration, function, dimension) triple we test whether there is a significant difference between the optimization times of the $(1+1)$ $\text{EA}_\alpha(A, b, 1/2)$ and the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$. The results are summarized in Table 1 and 2. The reported p-values are for the test "$T(A, b, 1/2) < T(A, b, 1/n)$?"; i.e., small p-values indicate a strong support for the null hypothesis that the running time distribution of the $(1+1)$ $\text{EA}_\alpha(A, b, 1/2)$ is dominated by that of the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$. Put differently, a small p-value is a strong evidence for the hypothesis that the $(1+1)$ $\text{EA}_\alpha(A, b, 1/2)$ is faster than the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$. We recall that the result of the Wilcoxon rank-sum test for the other one-sided null hypothesis (i.e., the hypothesis that $T(A, b, 1/2) > T(A, b, 1/n)$) is $1 - p$. We therefore highlight in Tables 1 and 2 p-values that are smaller than 5% or larger than 95%.

n	A	b	$T(A,b,1/n)$	$T(A,b,1/2)$	$(T_{1/n} - T_{1/2})/T_{1/n}$	$p(1/2 < 1/n)$
500	1.11	0.66	3,045	3,019	0.9%	0.096
500	1.2	0.85	3,063	2,994	2.3%	**0.028**
500	1.3	0.75	3,039	2,998	1.3%	0.092
500	2	0.5	3,035	2,980	1.8%	**0.005**
1000	1.11	0.66	6,780	6,788	-0.1%	0.231
1000	1.2	0.85	6,787	6,645	2.1%	**0.009**
1000	1.3	0.75	6,802	6,595	3.0%	**0.001**
1000	2	0.5	6,752	6,682	1.0%	0.086
2000	1.11	0.66	14,962	14,895	0.4%	0.112
2000	1.2	0.85	14,834	14,854	-0.1%	0.478
2000	1.3	0.75	14,839	14,768	0.5%	0.369
2000	2	0.5	15,297	15,133	1.1%	0.238

Table 1: Average running times of the $(1+1)$ $\text{EA}_\alpha(A, b, p_0)$ on OneMax for 1,001 independent repetitions and results of the one-sided Wilcoxon rank-sum tests for the null hypothesis that $T(A, b, 1/2) < T(A, b, 1/n)$.

We observe that for OneMax the p-values for the one-sided Wilcoxon rank-sum test are smaller than 0.5 for all tested configurations and problem dimensions, indicating that, if at all, there is a bias towards rejecting the null hypothesis and towards supporting that the $(1+1)$ $\text{EA}_\alpha(A, b, 1/2)$ is faster than the $(1+1)$ $\text{EA}_\alpha(A, b, 1/n)$. For three of the four configurations the p-values are much larger for problem dimension $n = 2000$ than for the smaller dimensions. For the 1/5-th success rule configuration ($A = 1.11, b = 0.66$) the p-value is largest for $n = 1000$. We do not have an explanation for this, but did not investigate further as the value does not indicate a statistically significant difference.

For LeadingOnes, the situation is different. Some p-values are rather large, and one value even larger then 95%, which might indicate that in this setting the initialization with $p_0 = 1/n$ may be more suitable than the initialization $p_0 = 1/2$. We recall, however, from Section 2.1 that the average optimal initial value is rather around 69/100. The absolute and relative differences in running time are all very small.

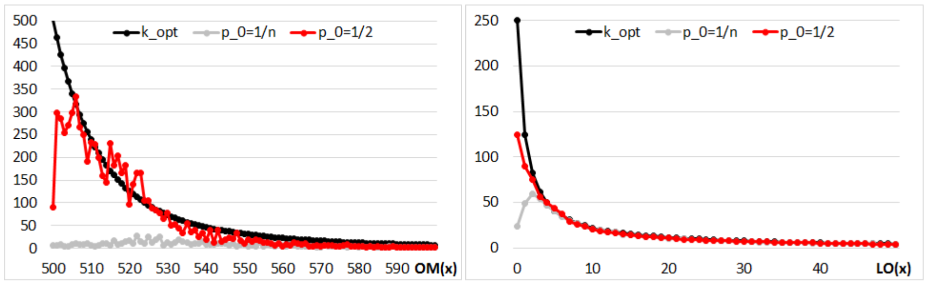
n	A	b	$T(A,b,1/n)$	$T(A,b,1/2)$	$(T_{1/n} - T_{1/2})/T_{1/n}$	$p(1/2 < 1/n)$
100	1.11	0.66	4,493	4,508	-0.3%	0.602
100	1.2	0.85	4,125	4,105	0.5%	0.183
100	1.3	0.75	4,141	4,144	-0.1%	0.574
100	2	0.5	4,182	4,245	-1.5%	**0.954**
250	1.11	0.66	28,348	28,130	0.8%	0.081
250	1.2	0.85	25,386	25,513	-0.5%	0.708
250	1.3	0.75	25,720	25,954	-0.9%	0.884
250	2	0.5	26,142	26,302	-0.6%	0.796
500	1.11	0.66	112,583	113,135	-0.5%	0.882
500	1.2	0.85	102,018	101,605	0.4%	0.082
500	1.3	0.75	102,862	102,903	0.0%	0.528
500	2	0.5	105,329	105,129	0.2%	0.375

Table 2: Average running times of the (1+1) $\mathrm{EA}_\alpha(A, b, p_0)$ on LeadingOnes for 1,001 independent repetitions and results of the one-sided Wilcoxon rank-sum tests for the null hypothesis that $T(A, b, 1/2) < T(A, b, 1/n)$.

2.4 Visualizing the Mutation Rate Adaptation

Finally, we investigate the evolution of the mutation rate. To this end, we have tracked for 100 independent runs the number of bits that have been flipped in each iteration, along with the function value of the corresponding parent. From this data we compute the average number of bit flips per function value. This data is plotted against the optimal mutation rates $k_{\text{opt,f}}(n, f(x))$ described in Section 2.1. Figure 2 summarizes this data. Note that we zoom in both plots into the interesting initial part of the optimization process.

We observe that the curves for $p_0 = 1/2$ have a better fit with k_{opt} than those for $p_0 = 1/n$. We also see that for the 1000-dimensional OneMax problem it is around $\text{Om}(x) = 560$ that the two curves converge. They are indistinguishable thereafter since the underlying adaptation rule is the same. For LeadingOnes the two curves do not differ by more than one for all $\text{Lo}(x)$-values greater than 11.



(a) OneMax $n = 1,000$ (b) LeadingOnes $n = 250$

Fig. 2: Average number of bit flips of the $(1 + 1)$ EA$_\alpha(A = 2, b = 0.5, p_0)$ in iterations starting with a parent individual of fitness $f(x)$

3 Sensitivity of the Self-Adjusting $(1 + (\lambda, \lambda))$ GA

We also test the relevance of the initial parameter value for the self-adjusting $(1 + (\lambda, \lambda))$ GA. This algorithm had been presented in [8] and has later been analyzed in [6]. It stores in the memory a current-best solution, creates from it λ offspring by mutation, and another λ offspring by a biased recombination of the best of the mutated offspring with its parent. The best recombined offspring replaces the parent individual if its function value is at least as good.

Using the recommended parametrization $p = \lambda/n$ and $c = 1/\lambda$ for the mutation rate and the crossover bias, respectively, the only parameter of the $(1+(\lambda, \lambda))$ GA becomes the population size λ. In [8] the following multiplicative update rule was suggested to control λ: If an iteration was successful, i.e., if at the end of the iteration we have identified a strictly better search point, we decrease λ to λ/F. We increase λ to $\lambda F^{1/4}$ otherwise. According to experiments reported in [8] the influence of the update strength F is not very pronounced. In line with common implementations of the 1/5-th success rule and the recommendations given in [6,8], we set F equal to 3/2. It was proven in [6] that the self-adjusting $(1 + (\lambda, \lambda))$ GA achieves a linear expected running time on OneMax [6]. This is asymptotically optimal among all possible parameter settings, and strictly better than what any static parameter choice can achieve [6].

We note that as in the (1+1) EA$_\alpha$, and unlike the experiments reported in [8], we enforce that at least one bit is flipped in the mutation phase. In addition, we evaluate a recombined offspring only if it is different from both of its parents. This can be tested efficiently and avoids useless function evaluations.

To test the influence of the initialization of λ, we perform 1,001 runs of the algorithm on OneMax instances of dimension $n = 500$, $n = 1000$, and $n = 2000$ with three different initialization rules: $\lambda_0 = 1$, $\lambda_0 = \ln n$, and $\lambda_0 = n$.

Quite surprisingly, the average optimization times vary drastically. To test for statistical significance, we first employ the Kruskal-Wallis test, which is an extension of the Wilcoxon rank-sum test for more than two data sets[3]. The

[3] We remark that a one-way ANOVA is not applicable as the Shapiro-Wilk normality test returns that the data is not normally distributed.

outcomes of the Kruskal-Wallis test of zero (or effectively zero) provide strong evidence that the outcomes are not identically distributed. This is confirmed by the pairwise Wilcoxon rank-sum tests, whose values are also reported in Table 3.

n	λ_0	T	KW test	$p(1 < \ln n)$	$p(1 < n)$	$p(\ln n < n)$
500	1	3,293				
500	$\ln n$	3,309	**0**	0.178	**0**	**0**
500	n	5,562				
1000	1	6,715				
1000	$\ln n$	6,6780	**0**	**0.004**	**0**	**0**
1000	n	11,366				
2000	1	13,716				
2000	$\ln n$	13,736	**2.29E-155**	0.556	**2.49E-105**	**9.11E-106**
2000	n	18,357				

Table 3: Results for the self-adjusting $(1+(\lambda,\lambda))$ GA with different initialization. Nearly all differences are statistically significant.

To visualize the adaptation of λ, we plot in Figure 3 its evolution in dependence of the $\text{OM}(x)$-value against the asymptotically optimal choice of $\lambda_{\text{opt}} = \lceil\sqrt{n/(n - \text{OM}(x))}\rceil$ for the $n = 1,000$-dimensional ONEMAX instance. The reported values are averages of 100 independent runs. In the middle range $650 < \text{OM}(x) < 850$ the average parameter values are all very close to the optimal ones. We therefore plot only the averages for the beginning of the optimization process, $n/2 = 500 < \text{OM}(x) \leq 650$, and its end, $850 \leq \text{OM}(x) \leq n = 1000$, respectively. We observe that for values $\text{OM}(x) > 624$ the curves are almost not distinguishable. In line with the empirical observations made in [6, 8] we also see that all curves track the increase of the optimal λ-value towards the end of the optimization process very well. This confirms that our modification of the $(1 + (\lambda, \lambda))$ GA does not harm, but rather improves its performance.

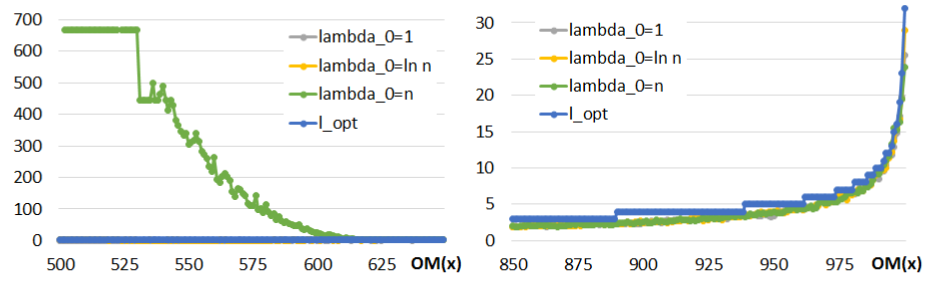


Fig. 3: Average value of λ per $\text{OM}(x)$-value for the self-adjusting $(1+(\lambda,\lambda))$ GA with update strength $F = 3/2$ and different initial parameter values values λ_0

n	λ	$G(r=1)$	$G(r=n/4)$	$(G_{n/4}-G_1)/G_1$	$p(1>n/4)$
5000	100	2,234	2,217	-0.74%	0.1144
5000	500	1,056	1,037	-1.73%	1.97E-22
5000	1000	852	834	-2.04%	3.16E-10
50000	100	63,627	62,666	-1.51%	0.6737
50000	500	65,139	65,722	0.90%	0.6833
50000	1000	62,814	61,567	-1.99%	0.2120

Table 4: Results for the average of 1001 independent runs of the $(1+\lambda)$ EA$_{r/2,2r}$ on OneMax

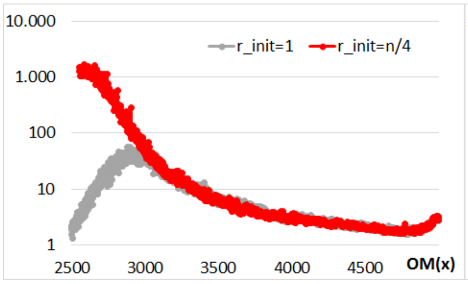
4 The $(1+\lambda)$ EA$_{r/2,2r}$ with 2-Rate Standard Bit Mutation

In [10] a novel idea how to control the mutation rate in a $(1+\lambda)$ EA has been presented. Their $(1+\lambda)$ EA$_{r/2,2r}$ stores a parameter r and creates in every iteration half of the offspring by standard bit mutation with mutation rate $r/(2n)$, while the other offspring are created with mutation rate $2r/n$. At the end of the iteration the value of r is updated as follows. With probability $1/2$ it is replaced randomly by either $r/2$ or $2r$ and with the remaining $1/2$ probability it is set to the value that the winning individual of the last iteration has been created with. Finally, the value r is capped at 1 if it is smaller, and at $n/4$, if it exceeds this value. As before, we implement this algorithm with the conditional standard bit mutation that enforces to flip at least one bit.

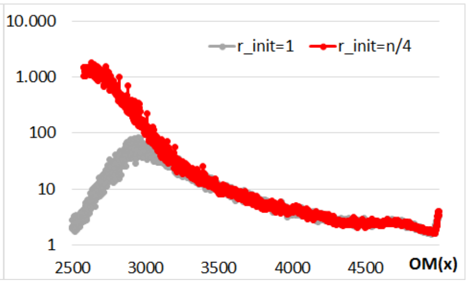
For the $(1+\lambda)$ EA$_{r/2,2r}$ we test two different initializations: $r_0 = 1$ and $r_0 = n/4$. Because of an efficient implementation, which samples waiting times instead of actually running the problem on the OneMax function, we can test the influence of these initial values for the $(1+\lambda)$ EA$_{r/2,2r}$ on OneMax instances of much larger dimensions $n = 5,000$ and $n = 50,000$. We perform tests for different values of λ: $\lambda = 100$, $\lambda = 500$, and $\lambda = 1000$. The results are summarized in Table 4. Note here that in contrast to all results presented above we report the average number of *generations* until an optimal solution has been evaluated for the first time, not the number of function evaluations. To obtain the latter, the $G(r)$-values need to be multiplied by λ.

The Wilcoxon rank-sum single-sided test for $G(A,b,r=1) < G(A,b,r=n/4)$ shows a small but significant difference between the two distributions when $n = 5,000$ for two values of λ. However, it vanishes for $n = 50,000$, and while there appear to be differences, they are not significant anymore as the optimization time distributions now overlap more.

We plot again the evolution of the r-values in Figure 4 and observe that the curves are quite similar for the two settings.



(a) $\lambda = 100$



(b) $\lambda = 500$

Fig. 4: Average value of r per $\textsc{Om}(x)$-value for the $(1+\lambda)$ $\text{EA}_{r/2,2r}$ on the 5000-dimensional OneMax problem

5 Conclusions and Future Work

We have analyzed the influence of the initialization of success-based multiplicative update schemes on the performance of three different evolutionary algorithms. For all tested settings, we could observe that the parameter values converge very quickly, even if initialized in their extreme points. The different initialization could nevertheless lead to statistically significant performance gaps. In the case of the $(1+1)$ EA_α and the $(1+\lambda)$ $\text{EA}_{r/2,2r}$ the relative performance losses of non-optimal initial parameter values are, however, rather small. In the case of the $(1+(\lambda,\lambda))$ GA, however, the performance loss could be as large as 69%, suggesting that more care needs to be taken when controlling population sizes.

We hope that our work encourages researchers and practitioners to experiment with parameter control schemes. Extending our results to more complex combinatorial optimization problems could be a reasonable next step towards a better understanding of which parameter control schemes to use under which conditions.

Acknowledgments

The authors would like to thank Eduardo Carvalho Pinto and Christian Giessen for providing their implementations of the $(1+1)$ EA_α and the $(1+(\lambda,\lambda))$ GA and the $(1+\lambda)$ $\text{EA}_{r/2,2r}$, respectively.

Our work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH and by the Australian Research Council project DE160100850.

References

1. Aleti, A., Moser, I.: A systematic literature review of adaptive parameter control methods for evolutionary algorithms. ACM Computing Surveys **49**, 56:1–56:35 (2016)

2. Ansótegui, C., Malitsky, Y., Samulowitz, H., Sellmann, M., Tierney, K.: Model-based genetic algorithms for algorithm configuration. In: Proc. of International Conference on Artificial Intelligence (IJCAI'15). pp. 733–739. AAAI Press (2015)
3. Bartz-Beielstein, T., Flasch, O., Koch, P., Konen, W.: SPOT: A Toolbox for Interactive and Automatic Tuning in the R Environment. In: Hoffmann, F., Hüllermeier, E. (eds.) Proceedings 20. Workshop Computational Intelligence. pp. 264–273. Universitätsverlag Karlsruhe (2010)
4. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Proc. of Parallel Problem Solving from Nature (PPSN'10). LNCS, vol. 6238, pp. 1–10. Springer (2010)
5. Devroye, L.: The compound random search. Ph.D. dissertation, Purdue Univ., West Lafayette, IN (1972)
6. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. Algorithmica **80**, 1658–1709 (2018)
7. Doerr, B., Doerr, C.: Theory of parameter control mechanisms for discrete black-box optimization: Provable performance gains through dynamic parameter choices. In: Doerr, B., Neumann, F. (eds.) Theory of Randomized Search Heuristics in Discrete Search Spaces. Springer (2018), to appear
8. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theoretical Computer Science **567**, 87–104 (2015)
9. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'16). pp. 1123–1130. ACM (2016)
10. Doerr, B., Gießen, C., Witt, C., Yang, J.: The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'17). pp. 1351–1358. ACM (2017)
11. Doerr, C., Wagner, M.: On the effectiveness of simple success-based parameter selection mechanisms for two classical discrete black-box optimization benchmark problems. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'18). ACM (2018), to appear. https://arxiv.org/abs/1803.01425
12. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation **3**, 124–141 (1999)
13. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**, 159–195 (2001)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential Model-Based Optimization for General Algorithm Configuration, pp. 507–523. Springer (2011)
15. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research **36**, 267–306 (2009)
16. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Transactions on Evolutionary Computation **19**, 167–187 (2015)
17. Lobo, F.G., Lima, C.F., Michalewicz, Z. (eds.): Parameter Setting in Evolutionary Algorithms, Studies in Computational Intelligence, vol. 54. Springer (2007)
18. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
19. Rechenberg, I.: Evolutionsstrategie. Friedrich Fromman Verlag (Günther Holzboog KG), Stuttgart (1973)
20. Schumer, M.A., Steiglitz, K.: Adaptive step size random search. IEEE Transactions on Automatic Control **13**, 270–276 (1968)

21. Thierens, D.: Adaptive mutation rate control schemes in genetic algorithms. In: Proc. of the 2002 Congress on Evolutionary Computation (CEC'02). vol. 1, pp. 980–985. IEEE (2002)