Evolutionary Algorithms with Self-adjusting Asymmetric Mutation

Amirhossein Rajabi(✉) and Carsten Witt

Technical University of Denmark, Kgs. Lyngby, Denmark
{amraj,cawi}@dtu.dk

Abstract. Evolutionary Algorithms (EAs) and other randomized search heuristics are often considered as unbiased algorithms that are invariant with respect to different transformations of the underlying search space. However, if a certain amount of domain knowledge is available the use of biased search operators in EAs becomes viable. We consider a simple (1+1) EA for binary search spaces and analyze an asymmetric mutation operator that can treat zero- and one-bits differently. This operator extends previous work by Jansen and Sudholt (ECJ 18(1), 2010) by allowing the operator asymmetry to vary according to the success rate of the algorithm. Using a self-adjusting scheme that learns an appropriate degree of asymmetry, we show improved runtime results on the class of functions OneMax$_a$ describing the number of matching bits with a fixed target $a \in \{0,1\}^n$.

Keywords: Evolutionary algorithms · Runtime analysis · Self-adjusting algorithms · Parameter control · Asymmetric mutations

1 Introduction

The rigorous runtime analysis of randomized search heuristics, in particular evolutionary algorithms (EAs), is a vivid research area [6,13,20] that has provided proven results on the efficiency of different EAs in various optimization scenarios and has given theoretically guided advice on the choice of algorithms and their parameters. A common viewpoint is that EAs in the absence of problem-specific knowledge should satisfy some *invariance* properties; e. g., for the space $\{0,1\}^n$ of bit strings of length n it is desirable that the stochastic behavior of the algorithm does not change if the bit positions are renamed or the meaning of zeros and ones at certain bit positions is exchanged (formally, these properties can be described by automorphisms on the search space). Black-box complexity theory of randomized search heuristics (e. g., [8]) usually assumes such invariances, also known under the name of unbiasedness [16].

In a given optimization scenario, a certain amount of domain knowledge might be available that invalidates the unbiasedness assumption. For example,

Supported by a grant from the Danish Council for Independent Research (DFF-FNU 8021-00260B).

on bit strings it might be known beforehand that zero-bits have a different interpretation than one-bits and that the total number of one-bits in high-quality solutions should lie within a certain interval. A prominent example is the minimum spanning tree (MST) problem [19], where, when modelled as a problem on bit strings of length m, m being the number of edges, all valid solutions should have exactly $n-1$ one-bits, where n is the number of vertices. In fact, the paper [19] is probably the first to describe a theoretical runtime analysis of an EA with a biased (also called asymmetric) mutation operator: they investigate, in the context of a simple (1+1) EA, an operator that flips each zero-bit of the current bit string x with probability $1/|x|_0$, where $|x|_0$ is the number of zero-bits in x, and each one-bit with probability $1/|x|_1$, where $|x|_1$ is the number of one-bits. As a consequence, the expected number of zero- and one-bits is not changed by this operator. Different biased mutation operators have been studied, both experimentally and theoretically, in, e. g., [18,21,25].

The (1+1) EA with asymmetric mutation operator (for short, asymmetric (1+1) EA) proposed in [19] was revisited in depth by Jansen and Sudholt [14] who investigated its effectiveness on a number of problems on bit strings, including the famous OneMax problem. In particular, they showed the surprising result that the asymmetric (1+1) EA optimizes OneMax in expected time $O(n)$, while still being able to optimize in expected time $O(n \log n)$ all functions from the generalized class $\text{OneMax}_a(x) := n - H(x, a)$, where $a \in \{0, 1\}^n$ is an arbitrary target and $H(\cdot, \cdot)$ denotes the Hamming distance. However, the question is open whether this speed-up by a factor $\Theta(\log n)$ for the all-ones target (and the all-zeros target) compared to general targets a is the best possible that can be achieved with an asymmetric mutation. In principle, since the operator knows which bits are ones and which are zeros it is not unreasonable to assume that it could be modified to let the algorithm approach the all-ones string faster than $\Theta(n)$ – however, this must not result in an algorithm that is tailored to the all-ones string and fails badly for other targets.

In this paper, we investigate whether the bias of the asymmetric operator from [14] can be adjusted to put more emphasis on one-bits when it is working on the OneMax_{1^n} function with all-ones string as target (and accordingly for zero-bits with OneMax_{0^n}) while still being competitive with the original operator on general OneMax_a. Our approach is to introduce a *self-adjusting bias*: the probability of flipping a one-bit is allowed to decrease or increase by a certain amount, which is upper bounded by $r/|x|_1$ for a parameter r; the probability of flipping a zero-bit is accordingly adjusted in the opposite direction. A promising setting for this bias is learned through a self-adjusting scheme being similar in style with the 1/5-rule [2] and related techniques: in a certain observation phase of length N two different parameter values are each tried $N/2$ times and the value that is relatively more successful is used in the next phase. Hence, this approach is in line with a recent line of theoretical research of self-adjusting algorithms where the concrete implementation of self-adjustment is an ongoing debate [4,5,7,9–11,15,23,24]. See also the recent survey article [3] for an in-depth coverage of parameter control, self-adjusting algorithms, and theoretical runtime results.

We call our algorithm the *self-adjusting (1+1) EA with asymmetric mutation* (Asym-SA-(1+1) EA) and conduct a rigorous runtime analysis on the OneMax$_a$ problem. Since the above-mentioned parameter r also determines the expected number of flipping bits of any type, we allow the bias of zero- resp. one-bits only to change by a small constant. Nevertheless, we can prove a speed-up on the OneMax function with target 1^n (and analogously for ZeroMax with target 0^n) by a factor of at least 2 compared to the previous asymmetric (1+1) EA since the bias is adjusted towards the "right" type of bits. On general OneMax$_a$, where a both contains zeroes and ones, we prove that the bias is not strongly adjusted to one type of bits such that we recover the same asymptotic bound $O(n \log(\min\{|a|_0, |a|_1\}))$ as in [14]. These results represent, to the best of our knowledge, the first runtime analysis of a self-adjusting asymmetric EA and pave the way for the study of more advanced such operators.

This paper is structured as follows: in Sect. 2, we introduce the studied algorithms and fitness function. In Sect. 3, we prove for the function OneMax with the all-ones string as target that the Asym-SA-(1+1) EA is by a factor of roughly 2 faster than the original asymmetric (1+1) EA from [14]. For targets a containing both zero- and one-bits, we show in Sect. 4 that our algorithm is asymptotically not slowed down and in fact insensitive to the second parameter α of the algorithm that controls its learning speed. Experiments in Sect. 5 demonstrate that the speedup by a factor of at least 2 already can be observed for small problem sizes while our algorithm is not noticeably slower than the original asymmetric one on the mixed target with half ones and zeros. We finish with some conclusions.

2 Preliminaries

2.1 Algorithms

We consider asymmetric mutation in the context of a (1+1) EA for the maximization of pseudo-boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$, which is arguably the most commonly investigated setting in the runtime analysis of EAs. The framework is given in Algorithm 1, where we consider the following two choices for the mutation operator:

Standard bit mutation Flip each bit in a copy of x independently with probability $\frac{1}{n}$.

Asymmetric mutation Flip each 0-bit in a copy of x with probability $\frac{1}{2|x|_0}$ and each 1-bit with probability $\frac{1}{2|x|_1}$ (independently for all bits).

With standard bit mutation, we call the algorithm the *classical (1+1) EA* and the one with asymmetric mutation the (static) *asymmetric (1+1) EA*. The latter algorithm stems from [14] and differs from the one from [19] by introducing the two factors 1/2 to avoid mutation probabilities above 1/2.

Algorithm 1. (1+1) EA

```
Select $x$ uniformly at random from $\{0,1\}^n$
for $t \leftarrow 1, 2, \ldots$ do
    Create $y$ by applying a mutation operator to $x$.
    if $f(y) \geq f(x)$ then
        $x \leftarrow y$.
```

The *runtime* (synonymously, *optimization time*), of Algorithm 1, is the smallest t where a search point of maximal fitness (i. e., f-value) has been created; this coincides with the number of f-evaluations until that time. We are mostly interested in the expected value of the runtime and call this the *expected runtime.*

As motivated above, we investigate a biased mutation operator that can vary the individual probabilities of flipping 0- and 1-bits. We propose a rather conservative setting that flips 0-bits of the underlying string x with probability $\frac{r_0}{|x|_0}$ for some value $0 \leq r_0 \leq 1$ and 1-bits with probability $\frac{1-r_0}{|x|_1}$; hence increasing the probability of one type decreases the probability of the other type. Formally, we also allow other ranges $0 \leq r_0 \leq r$ for some $r \geq 1$, which could make sense for problems where more than one bit must flip for an improvement. However, in the context of this paper, we fix $r = 1$. To ease notation, we use $r_1 \coloneqq r - r_0$, i. e., $r_1 = 1 - r_0$. Since r_i describes the expected number of flipping i-bits, $i \in \{0, 1\}$, we call r_0 the 0-strength and accordingly r_1 the 1-strength.

We now formally define the self-adjusting (1+1) EA with asymmetric mutation (Asym-SA-(1+1) EA), see Algorithm 2. Initially, $r_0 = r_1 = \frac{1}{2}$ so that the operator coincides with the static asymmetric mutation. In an observation phase of length N, which is a parameter to be chosen beforehand, two different pairs of 0- and 1-strengths are tried alternatingly, which differ from each other by an additive term of 2α that controls the speed of change. The pair that leads to relatively more successes (i. e., strict improvements) is used as the ground pair for the next phase. In case of a tie (including a completely unsuccessful phase) a uniform random choice is made. Also, we ensure that the strengths are confined to $[\alpha, 1-\alpha]$. Hereinafter, we say that we *apply asymmetric mutation with probability pair* (p_0, p_1) if we flip every 0-bit of the underlying string x with probability p_0 and every 1-bit with probability p_1 (independently for all bits). Note that the asymmetric (1+1) EA from [14] applies the probability pair $(\frac{1}{2|x|_0}, \frac{1}{2|x|_1})$.

So far, apart from the obvious restriction $\alpha < 1/4$, the choice of the parameter α is open. We mostly set it to small constant values but also allow it to converge to 0 slowly. In experiments, we set the parameter α to 0.1. Note that if the extreme pair of strengths $(\alpha, 1-\alpha)$ is used then 0-bits are only rarely flipped and 1-bits with roughly twice the probability of the static asymmetric operator.

In this paper, we are exclusively concerned with the maximization of the pseudo-Boolean function

$$\textsc{OneMax}_a(x) \coloneqq n - H(x, a)$$

Algorithm 2. Self-adjusting (1+1) EA with Asymmetric Mutation (Asym-SA-(1+1) EA); parameters: N = length of observation phase, α = learning rate

```
Select $x$ uniformly at random from $\{0,1\}^n$.
$r \leftarrow 1$. // parameter $r$ fixed to 1 in this paper
$r_0 \leftarrow \frac{1}{2}$, $r_1 \leftarrow r - r_0$.
$b \leftarrow 0$.
for $t \leftarrow 1, 2, \ldots$ do
    $p_- \leftarrow (\frac{r_0-\alpha}{|x|_0}, \frac{r_1+\alpha}{|x|_1})$, $p_+ \leftarrow (\frac{r_0+\alpha}{|x|_0}, \frac{r_1-\alpha}{|x|_1})$.
    Create $y$ by applying asymmetric mutation with pair $p_-$ if $t$ is odd and with pair $p_+$ otherwise.
    if $f(y) \geq f(x)$ then
        $x \leftarrow y$.
        if $f(y) > f(x)$ then
            if $t$ is odd then
                $b = b - 1$.
            else
                $b = b + 1$.
    if $t \equiv 0 \pmod N$ then
        if $b < 0$ then
            Replace $r_0$ with $\max\{r_0 - \alpha, 2\alpha\}$ and $r_1$ with $\min\{r_1 + \alpha, 1 - 2\alpha\}$.
        else if $b > 0$ then
            Replace $r_0$ with $\min\{r_0 + \alpha, 1 - 2\alpha\}$ and $r_1$ with $\max\{r_1 - \alpha, 2\alpha\}$.
        else
            Perform one of the following two actions with prob. 1/2:
              – Replace $r_0$ with $\max\{r_0 - \alpha, 2\alpha\}$ and $r_1$ with $\min\{r_1 + \alpha, 1 - 2\alpha\}$.
              – Replace $r_0$ with $\min\{r_0 + \alpha, 1 - 2\alpha\}$ and $r_1$ with $\max\{r_1 - \alpha, 2\alpha\}$.
        $b \leftarrow 0$.
```

for an unknown target $a \in \{0,1\}^n$, where $H(\cdot,\cdot)$ denotes the Hamming distance. Hence, $\text{OneMax}_a(x)$ returns the number of matching bits in x and a. In unbiased algorithms, the target a is usually and without loss of generality assumed as the all-ones string 1^n, so that we denote $\text{OneMax} = \text{OneMax}_{1^n}$. In the considered asymmetric setting, the choice a makes a difference. Note, however, that only the number $|a|_1$ influences the runtime behavior of the considered asymmetric algorithms and not the absolute positions of these $|a|_1$ 1-bits.

3 Analysis of Self-adjusting Asymmetric Mutation on OneMax

In this section, we show in Theorem 1 that the Asym-SA-(1+1) EA is by a factor of roughly 2 faster on OneMax, i. e., when the target a is the all-ones string, than the static asymmetric (1+1) EA from [14]. The proof shows that the self-adjusting scheme is likely to set the 0-strength to its maximum $1 - 2\alpha$ which makes improvements more likely than with the initial strength of 1/2. On the way to the proof, we develop two helper results, stated in Lemmas 1 and 2 below.

In the following lemma, we show that the algorithm is likely to observe more improvements with larger 0-strength and smaller 1-strength on OneMax.

Lemma 1. *Consider the Asym-SA-(1+1) EA on* $\textsc{OneMax} = \textsc{OneMax}_{1^n}$ *and let* x *denote its current search point. For* $0 \le \beta \le 1 - r_0$ *we have*

$$\Pr\left(S_{(\frac{r_0+\beta}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right) \ge \Pr\left(S_{(\frac{r_0}{|x|_0}, \frac{r_1}{|x|_1})}\right) + r_1 r_0 (1 - e^{-\beta}),$$

where $S_{(p_0,p_1)}$ *is the event of observing a strict improvement when in each iteration zero-bits and one-bits are flipped with probability* p_0 *and* p_1 *respectively.*

Proof. Consider the following random experiment which is formally known as a coupling (see [1] for more information on this concept). Assume that we flip bits in two phases: in the first phase, we flip all one-bits with probability $(r_1 - \beta)/|x|_1$ and all zero-bits with probability $r_0/|x|_0$. In the second phase, we only flip zero-bits with probability $\beta/(|x|_0 - r_0)$. We fixed $|x|_0$ and $|x|_1$ before the experiment. At the end of the second phase, the probability of a bit being flipped equals $(r_1 - \beta)/|x|_1$ if the value of the bit is one and $(r_0 + \beta)/|x|_0$ if the value of the bit is zero. The former is trivial (the bit is flipped only once with that probability) but for the latter, the probability of flipping each zero-bit equals $1 - (1 - r_0/|x|_0)(1 - \beta/(|x|_0 - r_0)) = (r_0 + \beta)/|x|_0$. Therefore, the probability of observing an improvement at the end of the second phase is

$$\Pr\left(S_{(\frac{r_0+\beta}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right). \tag{1}$$

Let us now calculate the probability of observing an improvement differently by computing the probability of success in each phase separately. At the end of the first phase, the probability of an improvement is $\Pr(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})})$. It fails with probability $1 - \Pr(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})})$ and the probability of success in the second phase is at least $(1 - (r_1 - \beta)/|x|_1)^{|x|_1} \Pr(S_{(\frac{\beta}{|x|_0 - r_0}, 0)})$, where no one-bits are flipped in the first phase and the algorithm finds a better search point when it only flips zero-bits with probability $\beta/(|x|_0 - r_0)$. Altogether we can say that the probability of observing a success is at least

$$\Pr\left(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right) + \left(1 - \Pr\left(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right)\right)\left(1 - \frac{r_1 - \beta}{|x|_1}\right)^{|x|_1} \Pr\left(S_{(\frac{\beta}{|x|_0 - r_0}, 0)}\right). \tag{2}$$

By considering (1) and (2), we obtain

$$\Pr\left(S_{(\frac{r_0+\beta}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right) \ge \Pr\left(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right) + \left(1 - \Pr\left(S_{(\frac{r_0}{|x|_0}, \frac{r_1-\beta}{|x|_1})}\right)\right)\left(1 - \frac{r_1 - \beta}{|x|_1}\right)^{|x|_1} \Pr\left(S_{(\frac{\beta}{|x|_0 - r_0}, 0)}\right).$$

Note that $\Pr(S_{(\frac{r_0}{|x|_0},\frac{r_1-\beta}{|x|_1})}) \leq (r_0/|x|_0)|x|_0 = r_0$. Also, we have $\Pr(S_{(\frac{r_0}{|x|_0},\frac{r_1-\beta}{|x|_1})}) \geq \Pr(S_{(\frac{r_0}{|x|_0},\frac{r_1}{|x|_1})})$ since in the second setting, the probability of flipping one-bits is larger, with the same probability of flipping zero-bits, which results in flipping more one-bits.

Finally, we have

$$\begin{aligned}\Pr\Big(S_{(\frac{r_0+\beta}{|x|_0},\frac{r_1-\beta}{|x|_1})}\Big) &\geq \Pr\Big(S_{(\frac{r_0}{|x|_0},\frac{r_1-\beta}{|x|_1})}\Big)\\ &\quad+\Big(1-\Pr\Big(S_{(\frac{r_0}{|x|_0},\frac{r_1-\beta}{|x|_1})}\Big)\Big)\Big(1-\frac{r_1-\beta}{|x|_1}\Big)^{|x|_1}\Pr\Big(S_{(\frac{\beta}{|x|_0-r_0},0)}\Big)\\ &\geq \Pr\Big(S_{(\frac{r_0}{|x|_0},\frac{r_1}{|x|_1})}\Big)+(1-r_0)(1-r_1+\beta)\Big(1-(1-\frac{\beta}{|x|_0})^{|x|_0}\Big)\\ &\geq \Pr\Big(S_{(\frac{r_0}{|x|_0},\frac{r_1}{|x|_1})}\Big)+r_1r_0(1-e^{-\beta}),\end{aligned}$$

since $r_1 + r_0 = 1$ and $\beta \geq 0$ as well as for $t \geq 1$ and $0 \leq s \leq t$, we have the inequality $1 - s \leq (1 - s/t)^t \leq e^{-s}$. □

We can apply concentration inequalities to show that the larger success probability with the pair p_+ with high probability makes the algorithm to move to the larger 0-strength at the end of an observation phase. This requires a careful analysis since the success probabilities themselves change in the observation phase of length N. Due to space restrictions, the proof of the following lemma has been omitted; it can be found in the preprint [22].

Lemma 2. *Assume that* $\min\{|x|_0, |x|_1\} = \Omega(n^{5/6})$ *for all search points* x *in an observation phase of length* N *of the Asym-SA-(1+1) EA on* OneMax. *With probability at least* $1-\epsilon$, *if* $\alpha = \omega(n^{-1/12}) \cap (0, 1/4)$, *and* $N \geq 8\alpha^{-8}\ln(4/\epsilon)$, *for an arbitrarily small constant* $\epsilon \in (0, 1/2)$, *the algorithm chooses the probability pair* p_+ *on at the end of the phase.*

We can now state and prove our main result from this section.

Theorem 1. *The expected optimization time* $E(T)$ *of the Asym-SA-(1+1) EA with* $\alpha = \omega(n^{-1/12}) \cap (0, 1/4)$, $N \geq \lceil 8\alpha^{-8}\ln(4/\epsilon)\rceil$, *for an arbitrarily small constant* $\epsilon \in (0, 1/2)$, *and* $N = o(n)$ *on* OneMax *and* ZeroMax *satisfies*

$$\frac{n}{2}(1-\alpha)^{-1} \leq E(T) \leq \frac{n}{2}(1-4\alpha)^{-1} + o(n).$$

This theorem rigorously shows that the presented algorithm outperforms the original asymmetric (1+1) EA from [14] by a constant factor of roughly 2 on OneMax. Indeed, the number of iterations needed for the asymmetric (1+1) EA is at least n since by drift analysis [17], the drift is at most $|x|_0 \cdot 1/(2|x|_0) = 1/2$ (recalling that each 0-bit flips with probability $1/(2|x|_0)$) and $E[H(x^*, 1^n)] = n/2$ for the initial search point x^* while Theorem 1 proves that the Asym-SA-(1+1) EA finds the optimum point within $(1 - \Theta(\alpha))^{-1}n/2 + o(n)$ iterations.

We even believe that the speedup of our algorithm is close to $2e^{1/2}$; namely, the original asymmetric (1+1) EA in an improvement usually must preserve existing one-bits, which happens with probability $(1-1/(2|x|_1))^{|x|_1} \approx e^{-1/2}$. This probability is close to $1-2\alpha$ in our algorithm when the self-adaptation has increased the 0-strength to $1-2\alpha$ and thereby decreased the 1-strength to 2α.

Proof (of Theorem 1). By Chernoff's bound, we have $H(x^*, 1^n) \le n/2 + n^{3/4}$ with probability $1-e^{-\Omega(n)}$, where x^* is the initial search point. Having the elitist selection mechanism results in keeping that condition for all future search points.

Now, we divide the run of the algorithm into two epochs according to the number of zero-bits. In the first epoch, we assume that $|x|_0 > n^{5/6}$. Since $|x|_1 > n^{5/6}$ also holds (proved by Chernoff), according to Lemma 2 with $\epsilon < 1/2$, the algorithm sets p_+ as its current probability pair in each observation phase with probability at least $1-\epsilon$. Hence, we have a random walk so the number of phases to reach $p^* := ((1-\alpha)/|x|_0, \alpha/|x|_1)$, by using the Gambler's Ruin Theorem [12], is at most $(1/(2\alpha))/(1-\epsilon-\epsilon)$. Since each phase contains N iterations, the expected number of iterations to reach p^* for the first time is at most $N/(2\alpha(1-2\epsilon))$.

When p_+ is equal to p^*, the algorithm uses the probability pairs

$$p_+ = \left(\frac{1-\alpha}{|x|_0}, \frac{\alpha}{|x|_1}\right) \text{ and } p_- = \left(\frac{1-2\alpha}{|x|_0}, \frac{2\alpha}{|x|_1}\right)$$

in the iterations so the drift is greater than

$$s\frac{1-2\alpha}{s}\left(1-\frac{2\alpha}{n-s}\right)^{n-s} \ge (1-2\alpha)^2 \ge 1-4\alpha.$$

Consequently, via the additive drift theorem [17], we need at most $(n/2)(1-4\alpha)^{-1} + o(n)$ iterations where p_+ equals p^*.

After the first time where p_+ gets equal to p^*, there is a possibility of getting p_+ away from p^*. With the probability of ϵ, the algorithm chooses probability pair p_-. The expected number of phases to reach p^* again is $(1-2\epsilon)^{-1}$ from the Gambler's Ruin Theorem [12], resulting in $N\epsilon(1-2\epsilon)^{-1}$ extra iterations for each phase where p^* is p_+. The expected number of steps needed until we have one step with pair p^* is $\epsilon(1-2\epsilon)^{-1}$ and by linearity of expectation this factor also holds for the expected number of such steps in the drift analysis.

Overall, the expected number of iterations of the first epoch is at most

$$\frac{n}{2}(1-4\alpha)^{-1}\frac{\epsilon}{1-2\epsilon} + \frac{N}{2\alpha(1-2\epsilon)} + o(n) = \frac{n}{2}(1-4\alpha)^{-1}\frac{\epsilon}{1-2\epsilon} + O(\alpha^{-9}) + o(n),$$

where we used $N = o(n)$.

In the second epoch where $|x|_0 \le n^{5/6}$, the expected number of iterations is at most $O(n^{5/6}/\alpha)$ since in the worst case with the probability pair $p_- = (\alpha/|x|_0, (1-\alpha)/|x|_1)$ the drift is

$$\frac{s\alpha}{s(1-\frac{1-\alpha}{n-s})^{n-s}} \ge 2\alpha.$$

Altogether and by our assumption that $\alpha = \omega(n^{-1/12}) \cap (0, 1/4)$, we have

$$E[T] \leq (1-4\alpha)^{-1}n/2 + o(n).$$

Moreover, in order to compute the lower bound, we have the upper bound $s(1-\alpha)/s = 1-\alpha$ on the drift, so through the additive drift theorem [17], we have

$$E[T] \geq (1-\alpha)^{-1}n/2.$$

□

4 Different Targets Than All-Ones and All-Zeros

After we have seen that the self-adjusting asymmetry is beneficial on the usual OneMax and ZeroMax function, we now demonstrate that this self-adaptation does not considerably harm its optimization time on OneMax_a for targets a different from the all-ones and all-zeros string. A worst case bound, obtained by assuming a pessimistic strength of α for the bits that still have to be flipped, would be $O((n/\alpha)\log z)$ following the ideas from [14]. We show that the factor $1/\alpha$ stemming from this pessimistic assumption does not actually appear since it becomes rare that the strength takes such a worst-case value on OneMax_a if a contains both many zeros and ones. Hence, we obtain the same asymptotic bound as for the original asymmetric (1+1) EA in this case.

Theorem 2. *Assume $z = \min\{|a|_1, |a|_0\} = \omega(\ln n)$, $1/\alpha = O(\log z/\log\log n)$ and $N = O(1)$. Then the expected optimization time of the Asym-SA-(1+1) EA on OneMax_a is $O(n \log z)$, where the implicit constant in the O does not depend on α.*

Proof. W.l.o.g. $|a|_1 \leq n/2$ and a has the form 0^z1^{n-z}, where we refer to the first z bits as the prefix and the final $n-z$ bits as the suffix.

We divide the run of the Asym-SA-(1+1) EA into three epochs according to the Hamming distance $h_t = n - H(x_t, a)$ of the current search point at time t to a, where the division is partially inspired by the analysis in [14]. If $h_t \geq 2z$ there are at least $h_t - z \geq h_t/2$ 0-bits in the suffix that can be flipped to improve and we also have $|x_t|_0 \leq h_t + z \leq (3/2)h_t$. Then the probability of an improvement is at least $((h_t/2)(\alpha/|x|_0))(1-|x|_1/n)^{|x|_1} \geq (\alpha/3)e^{-1-o(1)}$. Hence, the expected length of the first epoch is $O(n/\alpha) = O(n\log z)$ since $1/\alpha = O(\log z/\log\log n)$.

In the second epoch, we have $h_t \leq 2z$. This epoch ends when $h_t \leq n/\log^3 n$ and may therefore be empty. Since nothing is to show otherwise, we assume $z \geq n/\log^3 n$ in this part of the analysis. Since the probability of flipping an incorrect bit and not flipping any other bit is always at least $(\alpha/n)e^{-1+o(1)}$ the expected length of this epoch is at most $\sum_{i=n/\log^3 n}^{z} \frac{e^2 n}{\alpha i} = O((n/\alpha)\ln\ln n)$, which is $O(n \ln z)$ since $1/\alpha = O(\log z/\log\log n)$.

At the start of the third epoch, we have $h_t \leq z^*$, where $z^* = \min\{2z, n/\log^3 n\}$. The epoch ends when the optimum is found and is divided into z^* phases of varying length. Phase i, where $1 \le i \le z^*$, starts when the Hamming distance to a has become i and ends before the step where the distance becomes strictly less than i; the phase may be empty. The aim is to show that the expected length of phase i is $O(n/i)$ independently of α. To this end, we concretely consider a phase of length cn/i for a sufficiently large constant c and divide it into subphases of length c_1/α^2 for a sufficiently large constant c_1. The crucial observation we will prove is that such a subphase with probability $\Omega(1)$ contains at least c_2/α^2 steps such that the r_0-strength is in the interval $[1/4, 3/4]$, with c_2 being another sufficiently large constant. During these steps the probability of ending the phase due to an improvement is at least $c_3 i/n$ for some constant $c_3 > 0$. Hence, the probability of an improvement within $O(n\alpha^2/i)$ such subphases is $\Omega(1)$, proving that the expected number of phases of length cn/i is $O(1)$. Altogether, the expected length of phase i is $O(n/i)$. Note that $i \le z^*$ and $1/\alpha = O(\log n/\log\log n)$, so that $n\alpha^2/i$ is asymptotically larger than 1. Summing over $i \in \{1, \dots, z^*\}$, the expected length of the third epoch is $O(n \log z)$.

We are left with the claim that at least $c_2/(2\alpha^2)$ steps in a subphase of length c_1/α^2 have a 0-strength within $[1/4, 3/4]$. To this end, we study the random process of the 0-strength and relate it to an unbiased Markovian random walk on the state space $\alpha, 2\alpha, \dots, 1-2\alpha, 1-\alpha$, which we rescale to $1, \dots, \alpha - 1$ by multiplying with $1/\alpha$. If the overarching phase of length cn/i leads to an improvement, there is nothing to show. Hence, we pessimistically assume that no improvement has been found yet in the considered sequence of subphases of length c_1/α^2 each. Therefore, since the algorithm makes a uniform choice in the absence of improvements, both the probability of increasing and decreasing the 0-strength equal 1/2. The borders 1 and $\alpha - 1$ of the chain are reflecting with probability 1/2 and looping with the remaining probability.

From any state of the Markov chain, it is sufficient to bridge a distance of $m = 1/(2\alpha)$ to reach 1/2. Pessimistically, we assume strength α (i. e., state 1) at the beginning of the subphase. Using well-known results from the fair Gambler's Ruin Theorem [12], the expected time to reach state m from X_0 conditional on not reaching state 0 (which we here interpret as looping on state 1) is $X_0(m-X_0)$. Also, the probability of reaching m from X_0 before reaching 0 equals X_0/m and the expected number of repetitions until state m is reached is the reciprocal of this. Combining both, the expected time until m is reached from $X_0 = 1$ is at most $m^2 - m$, and by Markov's inequality, the time is at most $2m^2 = \alpha^{-2}$ with probability at least 1/2.

We next show that the time to leave the interval $[m/2, 3m/2]$ (corresponding to strength $[1/4, 3/4]$) is at least c/α^2 with probability at least $\Omega(1)$. To this end, we apply Chernoff bounds to estimate the number of decreasing steps within c/α^2 steps and note that it is less than $1/\alpha$ with probability $\Omega(1)$. Altogether, with probability $\Omega(1)$ a subphase of length c_1/α^2 contains at least c_2/α^2 steps with strength within $[1/4, 3/4]$ as suggested.

The theorem follows by summing up the expected lengths of the epochs. □

5 Experiments

The results in the previous sections are mostly asymptotic. In addition, although the obtained sufficient value for N derived in Lemma 1 is constant for $\alpha = \Omega(1)$, it can be large for a relatively small bit string. Hence, in this section, we present the results of the experiments conducted in order to see how the presented algorithm performs in practice. More information about the experiments is available in [22].

We ran an implementation of Algorithm 2 (Asym-SA-(1+1) EA) on OneMax and OneMax$_a$ with $a = 0^{n/2}1^{n/2}$ and with n varying from 8000 to 20000. The selected parameters of the Algorithm Asym-SA-(1+1) EA are $\alpha = 0.1$ and $N = 50$. We compared our algorithm against the (1+1) EA with standard mutation rate $1/n$ and asymmetric (1+1) EA proposed in [14].

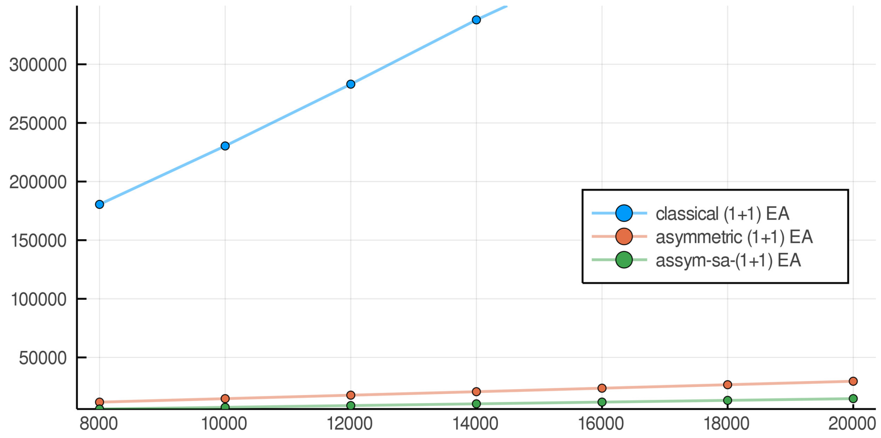

Fig. 1. Average number of fitness calls (over 1000 runs) the mentioned algorithms took to optimize OneMax.

The average optimization time of the experiment carried out on OneMax can be seen in Fig. 1. There is a clear difference between the performance of classical (1+1) EA and the algorithms using asymmetric mutations. It shows that these biased mutations can speed up the optimization time considerably. Also, the Asym-SA-(1+1) EA outperforms asymmetric (1+1) EA although we used relatively small N and large α. In detail, the average number of iterations that the Asym-SA-(1+1) EA and asymmetric (1+1) EA took to optimize for $n = 20000$ are 14864 and 29683 respectively. By considering the standard deviation of 238 and 385 as well, we can argue that our algorithm is faster by a factor of empirically around 2.

In Table 1, the average numbers of iterations which are taken for the algorithms to find the optimum on OneMax$_a$ with $a = 0^{n/2}1^{n/2}$ are available. The

Table 1. Average number of fitness calls (over 1000 runs) the mentioned algorithms took to optimize OneMax_a with $a = 0^{n/2}1^{n/2}$.

Algorithm	n						
	8000	10000	12000	14000	16000	18000	20000
Classical (1+1) EA	180019	231404	283965	335668	389859	444544	498244
Asymmetric (1+1) EA	180325	231999	283025	338264	391415	443910	502393
Asym-SA-(1+1) EA	180811	232412	284251	337519	390710	446396	503969

similarity between data for each bit string size suggests that the asymmetric algorithms perform neither worse nor better compared to classical (1+1) EA. More precisely, all p-values obtained from a Mann-Whitney U test between algorithms, with respect to the null hypothesis of identical behavior, are greater than 0.1.

6 Conclusions

We have designed and analyzed a (1+1) EA with self-adjusting asymmetric mutation. The underlying mutation operator chooses 0- and 1-bits of the current search point with different probabilities that can be adjusted based on the number of successes with a given probability profile.

As a proof of concept, we analyzed this algorithm on instances from the function class OneMax_a describing the number of matching bits with a target $a \in \{0,1\}^n$. A rigorous runtime analysis shows that on the usual OneMax function with target 1^n (and analogously for target 0^n), the asymmetry of the operator is adjusted in a beneficial way, leading to a constant-factor speedup compared to the asymmetric (1+1) EA without self-adjustment from [14]. For different targets a, the asymmetry of our operator does not become considerably pronounced so that the self-adjusting scheme asymptotically does not slow down the algorithm. Experiments confirm that our algorithm is faster than the static asymmetric variant on OneMax with target 1^n and not considerably slower for a target with equal number of 0- and 1-bits.

An obvious topic for future research is to consider other ranges for the parameter r that determines the maximum degree of asymmetry chosen by the algorithm. In the present framework, this parameter is linked to the expected number of flipping bits (regardless of their value), so that it is usually not advisable to increase it beyond constant values. Instead, we plan to investigate settings with two parameters where 0- and 1-bits each have their self-adjusted mutation strengths.

References

1. Doerr, B.: Probabilistic tools for the analysis of randomized optimization heuristics. In: Doerr and Neumann [6], pp. 1–87

2. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the (1+(λ, λ)) genetic algorithm. Algorithmica **80**(5), 1658–1709 (2018)
3. Doerr, B., Doerr, C.: Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices. In: Doerr and Neumann [6], pp. 271–321
4. Doerr, B., Doerr, C., Kötzing, T.: Static and self-adjusting mutation strengths for multi-valued decision variables. Algorithmica **80**(5), 1732–1768 (2018)
5. Doerr, B., Gießen, C., Witt, C., Yang, J.: The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate. Algorithmica **81**(2), 593–631 (2019)
6. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation - Recent Developments in Discrete Optimization. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-29414-4
7. Doerr, B., Witt, C., Yang, J.: Runtime analysis for self-adaptive mutation rates. In: Proceedings of GECCO 2018, pp. 1475–1482. ACM Press (2018)
8. Doerr, C.: Complexity theory for discrete black-box optimization heuristics. In: Doerr and Neumann [6], pp. 133–212
9. Doerr, C., Wagner, M.: Sensitivity of parameter control mechanisms with respect to their initialization. In: Proceedings of PPSN 2018, pp. 360–372 (2018)
10. Doerr, C., Ye, F., van Rijn, S., Wang, H., Bäck, T.: Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling (1+λ) EA variants on OneMax and LeadingOnes. In: Proceedings of GECCO 2018, pp. 951–958. ACM Press (2018)
11. Fajardo, M.A.H.: An empirical evaluation of success-based parameter control mechanisms for evolutionary algorithms. In: Proceedings of GECCO 2019, pp. 787–795. ACM Press (2019)
12. Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 1, 3rd edn. Wiley, Hoboken (1968)
13. Jansen, T.: Analyzing Evolutionary Algorithms - The Computer Science Perspective. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-17339-4
14. Jansen, T., Sudholt, D.: Analysis of an asymmetric mutation operator. Evol. Comput. **18**(1), 1–26 (2010)
15. Lässig, J., Sudholt, D.: Adaptive population models for offspring populations and parallel evolutionary algorithms. In: 2011 Proceedings of FOGA 2011, Schwarzenberg, Austria, 5–8 January 2011, Proceedings, pp. 181–192. ACM Press (2011)
16. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica**64**(4), 623–642 (2012)
17. Lengler, J.: Drift analysis. In: Doerr and Neumann [6], pp. 89–131
18. Neumann, A., Alexander, B., Neumann, F.: Evolutionary image transition using random walks. In: Correia, J., Ciesielski, V., Liapis, A. (eds.) EvoMUSART 2017. LNCS, vol. 10198, pp. 230–245. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55750-2_16
19. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theor. Comput. Sci. **378**(1), 32–40 (2007)
20. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16544-3
21. Raidl, G.R., Koller, G., Julstrom, B.A.: Biased mutation operators for subgraph-selection problems. IEEE Trans. Evol. Comput. **10**(2), 145–156 (2006)
22. Rajabi, A., Witt, C.: Evolutionary algorithms with self-adjusting asymmetric mutation. CoRR (2020). http://arxiv.org/abs/2006.09126

23. Rajabi, A., Witt, C.: Self-adjusting evolutionary algorithms for multimodal optimization. In: Proceedings of GECCO 2020 (2020, to appear)
24. Rodionova, A., Antonov, K., Buzdalova, A., Doerr, C.: Offspring population size matters when comparing evolutionary algorithms with self-adjusting mutation rates. In: Proceedings of GECCO 2019, pp. 855–863. ACM Press (2019)
25. Sutton, A.M.: Superpolynomial lower bounds for the (1+1) EA on some easy combinatorial problems. Algorithmica **75**(3), 507–528 (2016)