On the Log-Normal Self-Adaptation of the Mutation Rate in Binary Search Spaces

[Genetic Algorithms]

Johannes Kruisselbrink
Natural Computing Group
LIACS, Leiden University
Niels Bohrweg 1, Leiden
The Netherlands
jkruisse@liacs.nl

Rui Li
Natural Computing Group
LIACS, Leiden University
Niels Bohrweg 1, Leiden
The Netherlands
ruili@liacs.nl

Edgar Reehuis*
Natural Computing Group
LIACS, Leiden University
Niels Bohrweg 1, Leiden
The Netherlands
ereehuis@liacs.nl

Jeroen Eggermont
Division of Image Processing
Department of Radiology
C2S, LUMC
Albinusdreef 2, Leiden
The Netherlands
j.eggermont@lumc.nl

Thomas Bäck
Natural Computing Group
LIACS, Leiden University
Niels Bohrweg 1, Leiden
The Netherlands
baeck@liacs.nl

ABSTRACT

This paper discusses the adoption of self-adaptation for Evolutionary Algorithms operating in binary spaces using a direct encoding of the mutation rate. In particular, it focuses on the log-normal update rule for adapting the mutation rate, incorporated in a (μ, λ)-strategy. Although it is well known that this update rule requires a lower boundary of the mutation rate to prevent it from collapsing to zero, the naive approach of enforcing a fixed lower boundary has undesirable side-effects. This paper studies the dynamics of the fixed lower boundary approach in depth and proposes a simple alternative for dealing with the lower boundary issue.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Experimentation

Keywords

Genetic algorithms, adaptation/self-adaptation, empirical study

*Honda Research Institute Europe GmbH, OF/M, Germany

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Evolutionary Algorithms (EAs) working on binary search spaces commonly feature a probabilistic bit-flip scheme as mutation operator, where each bit is mutated with a certain probability p_{m}. Given an individual of the form $\vec{a} = (x_1, \ldots, x_n) \in \{0,1\}^n$, the mutation operator is

$$x_i' = \begin{cases} 1 - x_i & \text{with probability } p_{\mathrm{m}} \\ x_i & \text{otherwise} \end{cases}. \quad (1)$$

Choosing an appropriate mutation rate is known to have an important impact on the performance of an EA, but can be a tedious matter. This is because the optimal mutation rate is not constant over the course of the optimization and also varies between different problems being optimized. To avoid inappropriate settings of the mutation rate, which can lead to poor performance of an EA, the tuning of such control strategy parameters could also be done online and in an automated fashion. One way of doing so is to leave the tuning of such parameters to the process of evolution itself by embedding them into an individual's genome. This approach is known as self-adaptation and has shown to be beneficial for various EA variants [13].

Several approaches have been suggested to apply the concept of self-adaptation in binary spaces (amongst others, [2, 14, 5, 6, 17, 16, 15]). Although these studies report on self-adapting algorithms outperforming fixed mutation rate schemes, as of today the concept does not have a stable foothold in the Genetic Algorithms (GAs) community. Possible reasons for this are that including self-adaptation is more complicated compared to fixed mutation rate approaches, the gain is often marginal, and the resulting dynamics is not fully understood. This paper will focus on the latter, aiming to contribute to the understanding of the dynamics resulting from the incorporation of self-adaptation.

This paper studies the self-adaptation scheme as proposed in [14, 5, 6]. In this scheme, individuals are encoded as $\vec{a} =$

$(x_1, \ldots, x_n, p_\mathrm{m}) \in \{0,1\}^n \times]0,1[$ tuples, where $x_1, \ldots, x_n$ are the object variables of the individual and p_m is the mutation rate, included as an endogenous, directly encoded strategy parameter (cf. in indirect encoding, as proposed in [1] and used in [16], p_m is represented by a bitstring). The operator used for mutating the individuals is

$$p'_\mathrm{m} = \frac{1}{1 + \frac{1-p_\mathrm{m}}{p_\mathrm{m}} \cdot \exp\left(\gamma \cdot \mathcal{N}(0,1)\right)}, \tag{2}$$

$$x'_i = \begin{cases} 1 - x_i & \text{with probability } p'_\mathrm{m} \\ x_i & \text{otherwise} \end{cases}, \tag{3}$$

that is, bit-flip mutation is applied to each bit with the mutated probability p'_m. This scheme resembles the mutation scheme used in Evolution Strategies (ESs) (i.e., the single stepsize case [4]), but applies a *log-normal update rule* to p_m, see (2), keeping p_m in the interval $]0,1[$. For γ an empirically determined value of 0.22 is used [14].

Typical behavior of this self-adaptation scheme, as observed by, amongst others, Smith [16], is that when applying it in this pure form, the population is likely to converge to suboptimal solutions, caused by the mutation rates that prematurely collapse to very small values and thereby effectively yielding no mutation. An important fix included in [5, 6] is to restrict the value of p_m to the interval $[\frac{1}{n}, \frac{1}{2}]$, i.e.,

$$p'_\mathrm{m} = \min\left\{\frac{1}{2}, \max\left\{\frac{1}{n}, \frac{1}{1 + \frac{1-p_\mathrm{m}}{p_\mathrm{m}} \cdot \exp\left(\gamma \cdot \mathcal{N}(0,1)\right)}\right\}\right\}. \tag{4}$$

We investigate the effect of using this fix, which, as will be shown, can lead to unintended algorithmic behavior. Moreover, an alternative fix for dealing with the collapsing mutation rates is presented and compared with the other mutation schemes on multiple benchmark functions.

The remainder of this work is structured as follows: Section 2 presents the general algorithmic setup that is considered in this paper. Section 3 illustrates typical behavior induced by applying the self-adaptation schemes of (2) and (4). Section 4 studies the log-normal update rule and the effect of the fix of (4) in more depth. Section 5 presents an alternative to the fix of (4). Section 6 compares this new fix to the other mutation schemes on the *Counting Ones* and *Leading Ones* problems. Section 7 discusses *implicit elitism* that occurs using a mutation operator based on (1). Section 8 presents additional experiments on *NK landscape* problem instances. Section 9 closes with a discussion and outlook.

2. ALGORITHMIC SETUP

In this study we consider a (μ, λ)-strategy that uses only mutation. Hence, there is no recombination and there is no elitism. This is because we are purely interested in the isolated behavior of self-adaptation of the mutation rate. Authors acknowledge that within GAs, generally the crossover operator takes in a prominent place.

Algorithm 1 shows the general evolution loop that is considered in this study. It comprises a simple evolution cycle where in each generation, λ offspring are generated from the parent, and of those offspring, the best is selected as the parent for the next generation. As default mutation rate update methods, we use the following three schemes from literature:

Algorithm 1 General (μ, λ)-Strategy

```
t ← 0
P^(0) ← generate μ individuals a⃗_1, ..., a⃗_μ, randomly
while not terminate do
  for i = 1 to λ do
    a⃗'_i ← copy a randomly selected parent from P^(t)
    a⃗'_i ← mutate(a⃗'_i)
    f_i ← evaluate(a⃗'_i)
  end for
  P^(t+1) ← {a⃗'_{1:λ}, ..., a⃗'_{μ:λ}}, select μ best from λ total
  t ← t + 1
end while
```

- **Fixed**: Mutation with a fixed rate of $p_\mathrm{m} = \frac{1}{n}$;
- **SA1**: Self-adaptive mutation, were p'_m is generated by applying (2);
- **SA2**: Self-adaptive mutation with restricted bounds on p'_m, according to (4).

3. TYPICAL BEHAVIOR

We start the empirical study with a brief review of the behavior that is observed in using the three mutation schemes from literature with a $(1, 10)$-strategy on the classical *Counting Ones* problem [3] (also known as *ONEMAX* problem), which is defined as maximizing

$$f_\mathrm{CO}(\vec{x}) = \sum_{i=1}^{n} x_i. \tag{5}$$

For this problem, comprehensive information on the optimal mutation rate setting throughout the course of evolution is available [3].

Running each scheme only once, using the parameter settings listed in Table 1, yields the typical behavior plotted in Figure 1. With a $(1, 10)$-strategy on the Counting Ones problem, it can be seen that there is not much gained by applying self-adaptation. Although the self-adaptive schemes are marginally faster in the early stages of the evolutionary process, both stagnate prematurely and cannot find the optimum precisely. Using SA1 indeed shows the mutation rate collapsing to very small values, in line with earlier findings of, e.g., [16]. However, when using SA2, the algorithm shows drops in fitness when getting close to the optimum. Analyzing the mutation rate plot, p_m can be seen to start hovering at a certain level above $\frac{1}{n}$. This paper specifically focuses on the hovering behavior. Given the observations, one can surmise the hovering effect to be an undesirable artifact of the enforced lower bound of $p_\mathrm{m,min} = \frac{1}{n}$.

	Fixed	**SA1**	**SA2**
n	1,000	1,000	1,000
evaluation budget	50,000	50,000	50,000
$p_\mathrm{m,init}$	$\frac{1}{n}$	0.2	0.2
γ	N/A	0.22	0.22
$p_\mathrm{m,min}$	N/A	N/A	$1/n$
$p_\mathrm{m,max}$	N/A	N/A	1/2

Table 1: Experimental setup.

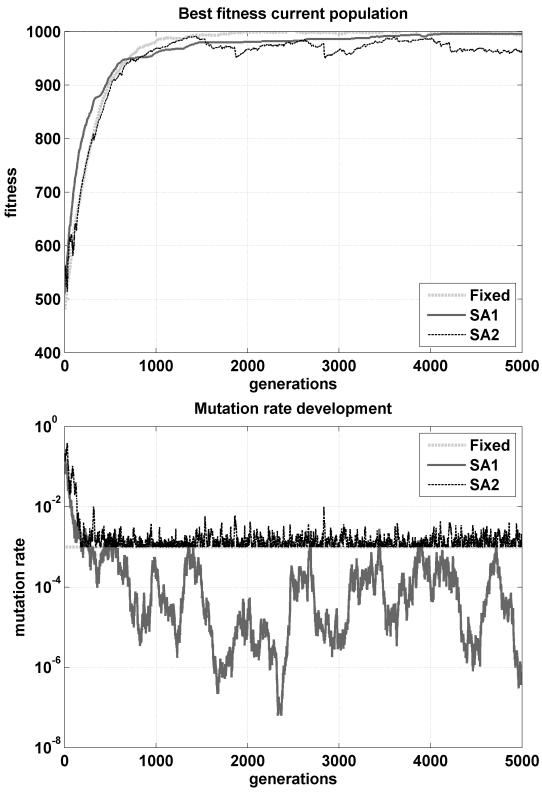

Figure 1: Fitness development (top) and the mutation rate development (bottom) for a single run per scheme, using a $(1, 10)$-strategy on the Counting Ones problem.

4. THE MUTATION RATE UPDATE RULE

In this section we take a closer look at the log-normal update rule for the mutation rate p_{m}, given in (2). The main question is: What is the magnitude of the difference in the mutation rate before and after adaptation (i.e., the difference between p_{m} and p'_{m}) as it approaches the lower boundary of $\frac{1}{n}$? This variance may provide insight in the magnitude of the hovering effect.

According to [14, 6], the update rule induces p'_{m} to be drawn from a logistic normal distribution with pdf

$$\mathrm{pdf}_{p'_{\mathrm{m}}}(x) = \frac{1}{\sqrt{2\pi}\gamma x(1-x)} \exp\left(\frac{-\left(\ln\frac{x}{1-x} - \ln\frac{p_{\mathrm{m}}}{1-p_{\mathrm{m}}}\right)^2}{2\gamma^2}\right). \tag{6}$$

One of the properties of the update rule is that the effective change of p_{m} after repeatedly updating it should be zero, which is enforced by construction, making the median of this distribution equal to p_{m}.

In order to gain insight into the behavior of $\mathbf{E}[p_{\mathrm{m}}]$, we set up the following experiment:

- Taking $n = 1000$ and $\gamma = 0.22$, we generate 10^6 mutated values p'_{m} for each $p_{\mathrm{m}} \in \{\frac{1}{n}, \ldots, \frac{n-1}{n}\}$ using (2), i.e., each p'_{m} value is the outcome of directly mutating one out of the $n-1$ p_{m} values;
- The median and mean of p'_{m} minus p_{m}, and variance of p'_{m} are computed for each p_{m} value; these are displayed in Figure 2.

What can be observed is that the mutation operator working on p_{m} indeed has a median equal to zero. The mean, however, has a clear bias, which varies with the value of p_{m}. Interestingly, the bias has the shape of a sine function. The variance of the distribution has a typical shape as well, being relatively high at $p_{\mathrm{m}} = 0.5$ and approaching zero for $p_{\mathrm{m}} \to 0$ and $p_{\mathrm{m}} \to 1$. When zooming in on the variance, we find a value of $\mathrm{Var}[p'_{\mathrm{m}}|p_{\mathrm{m}} = \frac{1}{1000}] \approx 5.2055 \cdot 10^{-8}$, corresponding to a standard deviation of $2.2765 \cdot 10^{-4}$.

Repeating the same experiment, but now incorporating the strict lower bound of $\frac{1}{n}$ on p_{m} using (4) and zooming in on the values $p_{\mathrm{m}} \in \{\frac{10}{10^4}, \ldots, \frac{50}{10^4}\}$ yields the same picture for the median, that is, it is not affected by the lower bound (see Figure 3). The mean on the other hand is affected, meaning that the expected value of p'_{m} will be greater than p_{m}, and that this effect will be stronger near $p_{\mathrm{m}} = \frac{1}{n}$.

5. A SIMPLE ALTERNATIVE

Although the SA2 method seems to work well, the inability of the mutation rate to stabilize at $\frac{1}{n}$ can be regarded as an undesirable artifact. In fact, this is solely due to the boundaries being set to $[\frac{1}{n}, \frac{1}{2}]$. In an attempt to remove the hovering behavior, we propose an alternative for enforcing the lower boundary and change the mutation scheme into

$$p'_{\mathrm{m}} = \frac{1}{1 + \frac{1-p_{\mathrm{m}}}{p_{\mathrm{m}}} \cdot \exp(\gamma \cdot \mathcal{N}(0,1))}, \tag{7}$$

$$x'_i = \begin{cases} 1-x & \text{with probability } p'_{\mathrm{m}} + p_{\mathrm{m,min}} \\ x & \text{otherwise} \end{cases}, \tag{8}$$

with $p_{\mathrm{m,min}} = \frac{1}{n}$. Hence, we preserve the original mutation scheme as used in SA1, but the actual mutation now takes place with an offset of $p_{\mathrm{m,min}}$. The envisioned effect of this change is that allowing p_{m} to become infinitely small will enable the actual mutation rate to get infinitely close to $p_{\mathrm{m}} = \frac{1}{n}$, and stabilize in the final stages of the evolution. We will refer to this alternative scheme as SA3.

6. EXPERIMENTS ON THE COUNTING ONES AND LEADING ONES PROBLEM

We perform an experimental study comparing the alternative self-adaptation mechanism SA3 to Fixed, SA1, and SA2. For SA3 we use the same settings as for SA2 (see Table 1). Next to the Counting Ones problem, the *Leading Ones* problem is considered, recently studied in a similar context in [7], and given by the maximization of

$$f_{\mathrm{LO}}(\vec{x}) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j. \tag{9}$$

The four mutation schemes are applied in two variants of the general (μ, λ)-strategy (see Algorithm 1): $(1, 10)$ and $(5, 35)$.

6.1 Results for the (1,10)-Strategy

Figure 4 shows the results of 30 runs of each method incorporated in a $(1, 10)$-strategy. Note that the plots are computed as the median of 30 runs, hence, single run dynamics are smoothed out. It can be observed that applying SA3

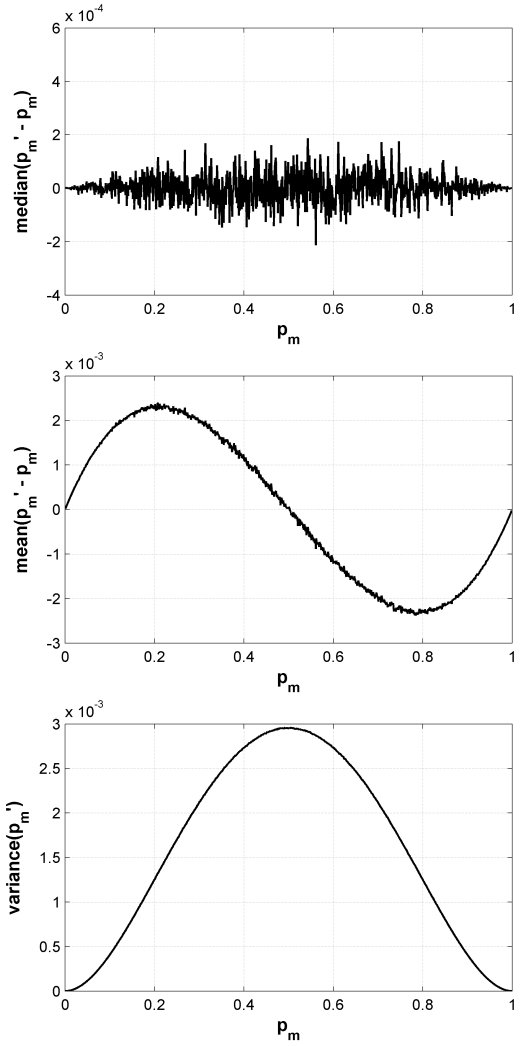

Figure 2: Analysis of the distribution of the mutation rate update rule. The median and mean of $p'_{\mathrm{m}} - p_{\mathrm{m}}$ and variance of p'_{m} are given for $p_{\mathrm{m}} \in \{\frac{1}{1000}, \ldots, \frac{999}{1000}\}$, using 10^6 samples per p_{m} value.

leads to self-adaptive mutation rate behavior that is much more stable than when using its counterparts SA1 and SA2. On the Counting Ones problem it reaches the optimum and remains there, obtaining a p_{m} value very close to zero that yields mutations with a rate close to $1/n$. On the Leading Ones, one can observe that SA3 also has a positive effect on the stability of the fitness development in the EA, as the higher mutation rate of the SA2 scheme gives rise to sudden drastic drops in fitness.

Another interesting observation is that the SA1 approach behaves differently on the Leading Ones problem than on the Counting Ones problem. It seems that the mutation rate remains at a higher rate throughout the course of evolution. From this, one may conclude that the principle of self-adaptation does work here, but is greatly biased towards selecting smaller values of p_{m}, because this decreases the chance of deteriorating an individual's fitness. That is, not mutating leads to a higher survival probability than mutating. SA2 tends to go to zero as well, but the artificial bound

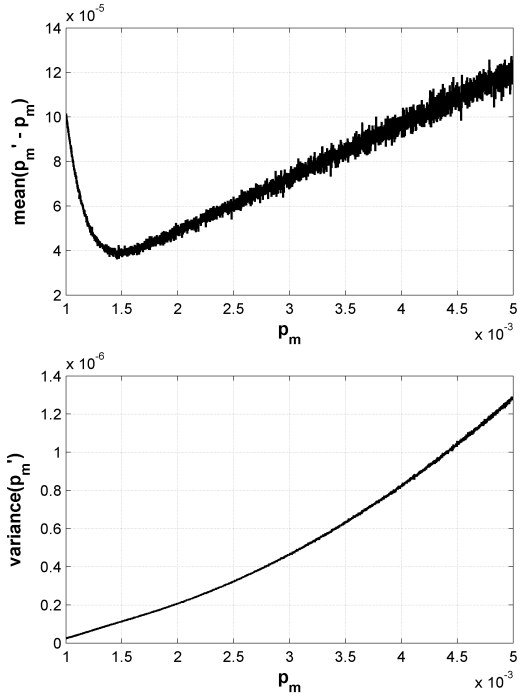

Figure 3: Analysis on the distribution of the mutation rate update rule with a strict lower bound of $\frac{1}{1000}$ on p'_m . The mean of $p'_{\mathrm{m}} - p_{\mathrm{m}}$ and variance of p'_{m} are given for $p_{\mathrm{m}} \in \{\frac{10}{10^4}, \ldots, \frac{50}{10^4}\}$, using 10^6 samples per p_{m} value.

induces a hovering effect. In SA3, the attraction to zero has the effective mutation rate converging to the lower boundary of $p_{\mathrm{m,min}} = 1/n$.

6.2 Results for the (5,35)-Strategy

Repeating the same experiments, but now using a $(5, 35)$-strategy yields interesting differences to the $(1, 10)$ results, see Figure 5. Increasing the population is beneficial for the SA2 and SA3 methods; especially SA2 shows drastic improvement. On the Counting Ones problem, the hovering effect still occurs but at a reduced rate, and the algorithm actually manages to converge to the global optimum. On the Leading Ones problem, SA2 gains in stability and now even outperforms SA3. With Leading Ones being a harder problem for which pure hill-climbing is not feasible, SA2 appears to benefit from the hovering effect through higher effective mutation rates, leading to more explorative behavior.

7. IMPLICIT ELITISM

Intuitively reasoning, we can state that in using larger population sizes there is an increased level of *implicit elitism*. Given a mutation rate p_{m}, the expected number of offspring that are not changed by mutation (i.e., the offspring for which no bit is flipped) can be calculated by

$$\begin{aligned} & \mathbf{E}[\text{number of offspring not changed by mutation}] \\ = \; & \sum_{i=1}^{\lambda} i \cdot \binom{\lambda}{i} \cdot ((1-p_{\mathrm{m}})^n)^i \cdot (1-(1-p_{\mathrm{m}})^n)^{\lambda-i}, \quad (10) \\ = \; & \lambda(1-p_{\mathrm{m}})^n. \quad (11) \end{aligned}$$

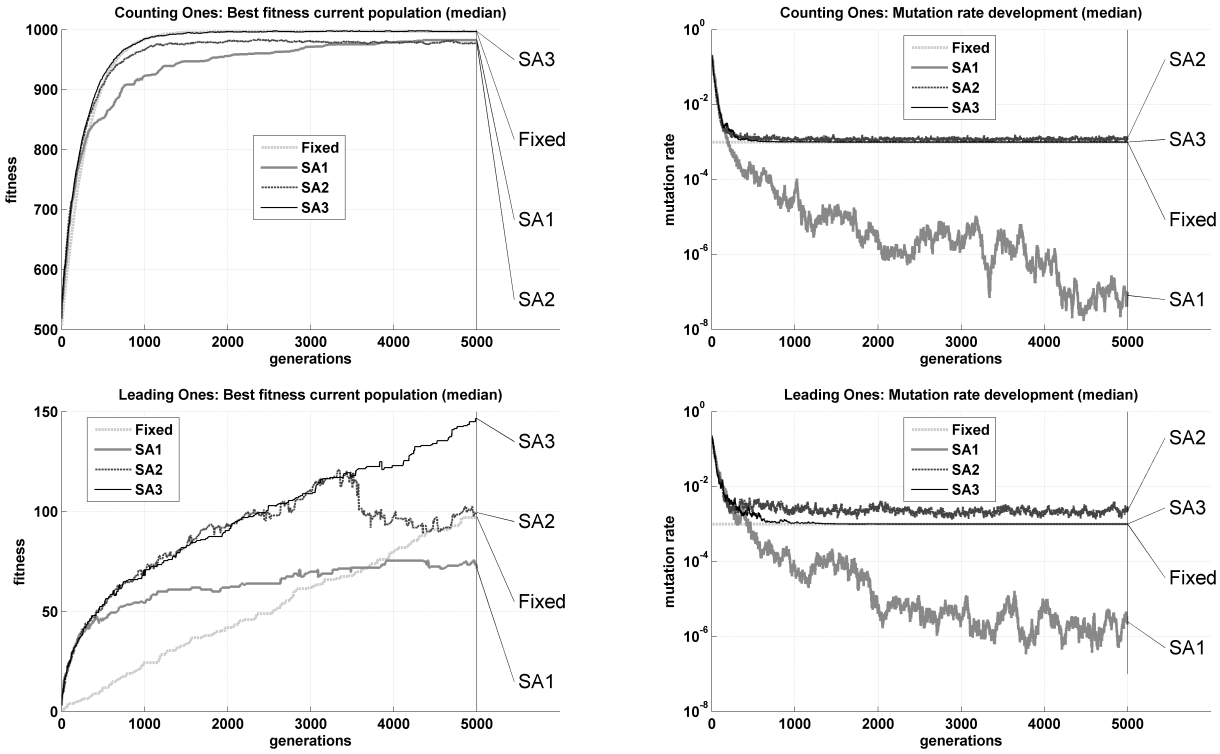

Figure 4: Results of 30 runs with a $(1, 10)$-strategy using the different mutation schemes. Top row: Counting Ones, bottom row: Leading Ones. Left column: median of the fitness of the best individual in the current population, right column: median of the mutation rate of the best individual in the current population.

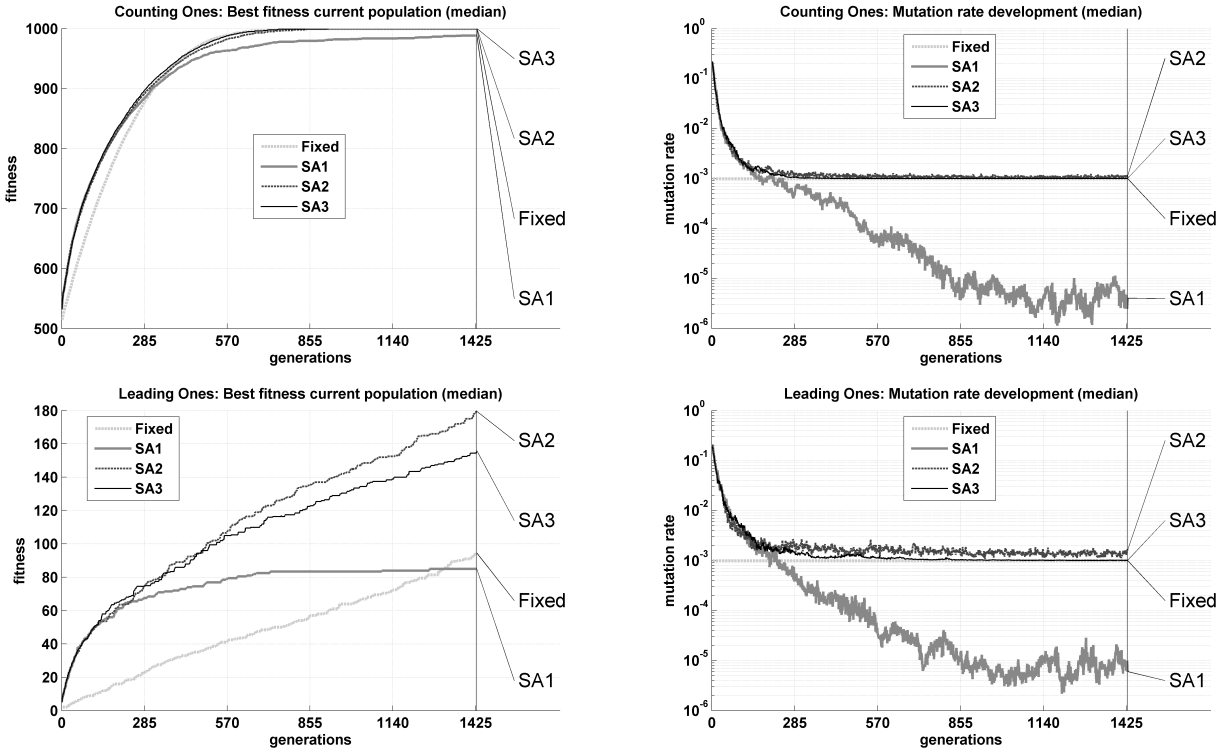

Figure 5: Results of 30 runs with a $(5, 35)$-strategy using the different mutation schemes. Top row: Counting Ones, bottom row: Leading Ones. Left column: median of the fitness of the best individual in the current population, right column: median of the mutation rate of the best individual in the current population.

From this, it can be seen that increasing the number of offspring will lead to a higher probability of generating an offspring individual equal to one of the parents after mutation, simply because of the fact that the actual mutation has not taken place. Moreover, this number scales proportionally to the population size and, interestingly, if we take $p_{\mathrm{m}} = 1/n$ we observe that

$$\lim_{n\to\infty} \lambda \left(1 - \frac{1}{n}\right)^n = \frac{\lambda}{e}. \quad (12)$$

That is, the fraction of individuals in the population that will not be affected by mutation is $1/e$ for a mutation rate of $p_{\mathrm{m}} = 1/n$.

Note that a similar analysis regarding this implicit elitism effect was made in [8], referring to it as a "hidden plus strategy".

8. EXPERIMENTS ON NK LANDSCAPES

Besides the experiments on the Counting Ones and Leading Ones problems, we consider *NK landscape* (NKL) problem instances [10, 9]. Next to verifying the results obtained on the two classical benchmarks, authors aim to extend current research to the more general class of *discrete nominal* problems in future work. NKLs form a good starting point, as these can be extended from the traditional binary case to (optionally mixed variable) instances involving nominal discrete, continuous, and/or integer variables [11].

NK landscapes were introduced by Kauffman [10] and devised to explore the way that *epistasis* (i.e., the interplay between genes) controls the 'ruggedness' of an adaptive landscape. NKLs are particularly useful for analyzing the dynamics of evolutionary search and thus often used as test problem generators for GAs. By design, NKLs have two favorable traits: First, the ruggedness and the degree of interaction between variables of NKL is easily controlled by two tunable parameters, the number of genes N and the number of epistatic links of each gene to other genes K. Second, for given values of N and K, a large number of NK landscapes can be created at random.

Traditional NKLs define a family of pseudo-boolean fitness functions $F : \{0,1\}^N \to \mathbb{R}^+$ that are generated by a stochastic algorithm. An NKL has two basic components: a structure for gene interaction (using an epistasis matrix E), and a way this structure is used to generate a fitness function for the possible genotypes. The gene interaction structure is created as follows: the genotype's fitness is the average of N fitness components F_i, with $i \in \{1,\ldots,N\}$. Each gene's fitness component F_i is determined by its own allele x_i, and also by K alleles at K ($0 \le K \le N-1$) epistatic genes distinct from i. The fitness function (which is to be minimized) reads

$$f_{\mathrm{NKL}}(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} F_i(x_i; x_{i_1}, \ldots, x_{i_K}), \vec{x} \in \{0,1\}^N, \quad (13)$$

where $\{i_1,\ldots,i_K\} \subset \{1,\ldots,N\} - \{i\}$. There are two ways for choosing K other genes: *adjacent neighborhoods*, where the K genes nearest to position i on the vector are chosen; and *random neighborhoods*, where these positions are chosen randomly on the vector. In this work we use random neighborhoods.

The computation of $F_i : \{0,1\}^K \to [0,1)$, $i \in \{1,\ldots,N\}$ is based on a fitness matrix F. For any i and for each of the 2^{K+1} bit combinations, a random number is drawn independently from a uniform distribution over $[0,1)$. Accordingly, for the generation of one (binary) NKL the setup algorithm has to generate $2^{K+1}N$ independent random numbers. The setup algorithm also creates an epistasis matrix E which for each gene i contains references to its K epistatic genes. Table 2 illustrates the fitness matrix and epistasis matrix of a NKL. A more detailed description of its implementation can be found in [9].

After having generated the epistasis and fitness matrices, for any input vector $\vec{x} \in \{0,1\}^N$ we can compute the fitness in $\mathcal{O}(KN)$ computational complexity via

$$f_{\mathrm{NKL}}(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} F_i[2^0 x_i + 2^1 x_{E_i[1]} + \cdots + 2^K x_{E_i[K]}]. \quad (14)$$

Note that the generation of F has an exponential computational complexity and space complexity in K, while being linear in N. The computational complexity for computing function values is linear in K and N for this implementation.

For the experiments, we generate standard (i.e., binary) NKLs with $K = 2$ and $K = 5$, converting from minimization to maximization by using the straightforward transformation $\tilde{f}_{\mathrm{NKL}}(\vec{x}) = -f_{\mathrm{NKL}}(\vec{x})$. Each generated problem instance consists of $N = 100$ boolean variables.

$E_1[1]$	$\cdots$	$\cdots$	$\cdots$	$E_1[K]$
$\cdots$	$\cdots$	$E_i[j]$	$\cdots$	$\cdots$
$E_N[1]$	$\cdots$	$\cdots$	$\cdots$	$E_N[K]$

$F_1[0]$	$\cdots$	$\cdots$	$\cdots$	$F_1[2^{K+1}-1]$
$\cdots$	$\cdots$	$F_i[j]$	$\cdots$	$\cdots$
$F_N[0]$	$\cdots$	$\cdots$	$\cdots$	$F_N[2^{K+1}-1]$

Table 2: Epistasis matrix E (left) and fitness matrix F (right).

8.1 Results for the (1,10)-Strategy

Figure 6 shows the results of 30 runs of each method using a $(1,10)$-strategy, on different NK landscape problem instances. The most striking result is that here, all SA schemes are, in the first 100 generations, outperformed by the fixed mutation rate scheme. Apparently, for NK landscapes problems, using SA in combination with a relatively high initial mutation rate does not work too well. On the other hand, all SA schemes catch up with the fixed mutation rate scheme after approximately 100 generations. Furthermore, it can be observed that the SA3 scheme for self-adaptation shows much more stable mutation rate behavior than SA1 and SA2, and manages to attain better fitness on both instances.

8.2 Results for the (5,35)-Strategy

Figure 7 shows the results of 30 runs of each method using a $(5,35)$-strategy. Like with the $(1,10)$-strategy, the SA3 scheme shows stable behavior on both problem instances. Here again we can observe a very good performance of the fixed mutation rate scheme in the early stages of the optimization. However, besides that, regarding performance all schemes perform quite similar. Like in the Counting Ones and Leading Ones experiments, presumably due to implicit elitistm, the larger population size improves the fitness development of the SA2 scheme.

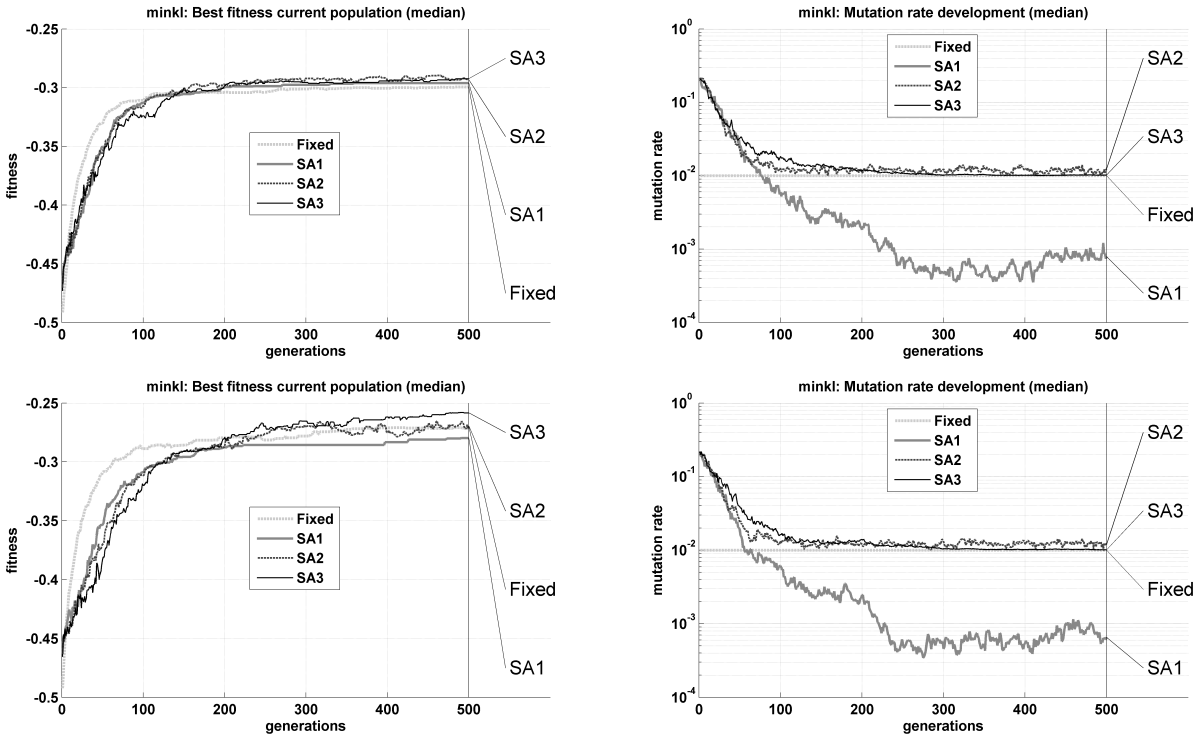

Figure 6: Results of the experiments on different NK landscapes with different mutation schemes, including SA3, using a $(1, 10)$-strategy. All results are calculated over 30 runs. Top row: $K = 2$, bottom row: $K = 5$. Left column: median of the fitness of the best individual in the current population, right column: median of the mutation rate of the best individual in the current population.

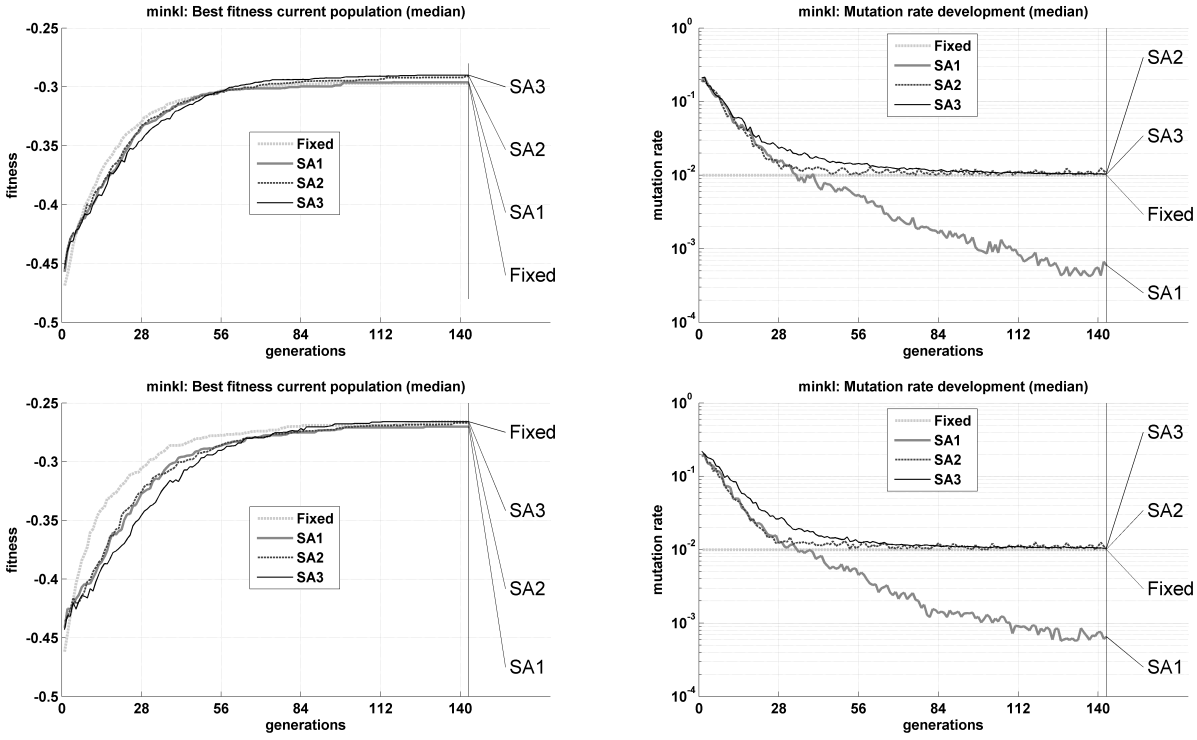

Figure 7: Results of the experiments on different NK landscapes with different mutation schemes, including SA3, using a $(5, 35)$-strategy. All results are calculated over 30 runs. Top row: $K = 2$, bottom row: $K = 5$. Left column: median of the fitness of the best individual in the current population, right column: median of the mutation rate of the best individual in the current population.

9. DISCUSSION AND CONCLUSION

We have shown that the boundary fix commonly applied to self-adaptation in EAs operating on binary search spaces, given in (4) and in this paper denoted SA2, has some unforeseen side effects that especially emerge in non-elitist schemes with relatively small population sizes. A simple alternative to the boundary fix was proposed, presented in Section 5 and denoted by SA3, that yields more stable behavior with respect to fitness development, and also a mutation rate development that is more in line with what would be expected from self-adaptation. It is shown that for smaller population sizes, this alternative boundary handling method can outperform the boundary fix. However, when using larger population sizes, the boundary fix wins from the simple alternative. This is probably due to an increased implicit elitism effect when using larger population sizes, which allows the EA to benefit from the higher mutation rate that is characteristic to SA2, leading to better explorative behavior.

For future research two questions are of particular interest: First, it is interesting to see what effect the incorporation of a crossover mechanism has on the behavior of the self-adaptation mechanisms discussed in this paper. Secondly, authors envision to apply the investigated self-adaptation mechanisms in discrete nominal search spaces. Next to studying the behavior on isolated discrete nominal problems, future work involving *Mixed-Integer Evolution Strategies* [12], which apply a discrete nominal self-adaptation scheme based on SA2, seems an interesting course of action. In [11], traditional binary NK landscapes were extended to a more general case in which the fitness value can be computed over different parameter types, namely continuous, integer, and discrete nominal.

Acknowledgments

J. Kruisselbrink acknowledges financial support of the Netherlands Organization for Scientific Research (NWO); dossier no. 612.066.618.

10. REFERENCES

[1] T. Bäck. Self-adaptation in genetic algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press, 1992.

[2] T. Bäck. The Interaction of Mutation Rate, Selection, and Self-Adaptation Within a Genetic Algorithm. In R. Männer and B. Manderick, editors, *2nd International Conference on Parallel Problem Solving from Nature - PPSN II*, pages 87–96. Elsevier, 1992.

[3] T. Bäck. Optimal Mutation Rates in Genetic Search. In S. Forrest, editor, *Fifth International Conference on Genetic Algorithms - ICGA 5*, pages 2–8. Morgan Kaufmann, 1993.

[4] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK, 1996.

[5] T. Bäck and M. Schütz. Evolution Strategies for Mixed-Integer Optimization of Optical Multilayer Systems. In J. McDonnell et al., editor, *Evolutionary Programming IV - Fourth Annual Conference on Evolutionary Programming - EP'95*, pages 33–51. MIT Press, 1995.

[6] T. Bäck and M. Schütz. Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In Z. Ras and M. Michalewicz, editors, *9th International Symposium on Foundations of Intelligent Systems - ISMIS'96*, volume 1079 of *LNCS*, pages 158–167. Springer, 1996.

[7] S. Böttcher, B. Doerr, and F. Neumann. Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem. In R. Schaefer et. al., editor, *11th International Conference on Parallel Problem Solving from Nature - PPSN XI*, volume 6238 of *LNCS*, pages 1–10. Springer, 2010.

[8] R. Breukelaar and T. Bäck. Self-adaptive Mutation Rates in Genetic Algorithm for Inverse Design of Cellular Automata. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, GECCO '08, pages 1101–1102, New York, NY, USA, 2008. ACM.

[9] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[10] S. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, USA, 1993.

[11] R. Li, M. Emmerich, J. Eggermont, E. Bovenkamp, T. Bäck, J. Dijkstra, and J. Reiber. Mixed-Integer NK Landscapes. In T. Runarsson et. al., editor, *9th International Conference on Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *LNCS*, pages 42–51. Springer, 2006.

[12] R. Li, M.T.M. Emmerich, J. Eggermont, E.G.P. Bovenkamp, T. Bäck, J. Dijkstra, and J. Reiber. Optimizing a Medical Image Analysis System Using Mixed-Integer Evolution Strategy. In Stefano Cagnoni, editor, *Evolutionary Image Analysis and Signal Processing*, volume 213 of *SCI*, pages 91–112. Springer, 2009.

[13] S. Meyer-Nieberg and H.-G. Beyer. Self-adaptation in evolutionary algorithms. In *Parameter Setting in Evolutionary Algorithms*, pages 47–75. 2007.

[14] J. Obalek. Rekombinationsoperatoren für Evolutionsstrategien. Diploma thesis, University of Dortmund, Department of Computer Science, 1994.

[15] M. Serpell and J. Smith. Self-Adaptation of Mutation Operator and Probability for Permutation Representations in Genetic Algorithms. *Evolutionary Computation*, 18(3):491–514, 2010.

[16] J. Smith. Parameter Perturbation Mechanisms in Binary Coded GAs with Self-Adaptive Mutation. In K. De Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms 7*, pages 329–346. Morgan Kaufmann, 2003.

[17] D. Thierens. Adaptive Mutation Rate Control Schemes in Genetic Algorithms. In *2002 IEEE Congress on Evolutionary Computation, CEC 2002*, pages 980–985. IEEE Press, 2002.