Optimal Mutation Rates in Genetic Search

Thomas Bäck*
Department of Computer Science
University of Dortmund
D–44221 Dortmund
Germany

Abstract

The optimization of a single bit string by means of iterated mutation and selection of the best (a (1+1)-Genetic Algorithm) is discussed with respect to three simple fitness functions: The counting ones problem, a standard binary encoded integer, and a Gray coded integer optimization problem. A mutation rate schedule that is optimal with respect to the success probability of mutation is presented for each of the objective functions, and it turns out that the standard binary code can hamper the search process even in case of unimodal objective functions. While normally a mutation rate of $1/l$ (where l denotes the bit string length) is recommendable, our results indicate that a variation of the mutation rate is useful in cases where the fitness function is a multimodal pseudo-boolean function, where multimodality may be caused by the objective function as well as the encoding mechanism.

1 INTRODUCTION

Genetic Algorithms (GAs) (Holland 1975) are the most prominent, widely used representatives of *Evolutionary Algorithms*, a class of probabilistic search algorithms based on the model of organic evolution (see (Bäck and Schwefel 1993) for an overview of Evolutionary Algorithms). Basic components of all Evolutionary Algorithms are a *population* of *individuals*, each of which represents a search point in the space of potential solutions to a given optimization problem, and random operators that are intended to model *mutation* and *selection*. The optimization problem gives a quality information (*fitness*) for the individuals, and the selection process favours individuals of higher fitness to transfer their information to the next generation.

*baeck@ls11.informatik.uni-dortmund.de

As a particular instance of Evolutionary Algorithms, Holland's Genetic Algorithm is characterized by special realizations of genetic operators and the representation of individuals (see e.g. (Holland 1975; Goldberg 1989)):

(1) Individuals a are represented as bit strings of fixed length l, i.e. $a \in \mathbb{B}^l$ where $\mathbb{B} = \{0, 1\}$.
(2) Mutation is a bit reversal event that occurs with small probability p_m per bit (common settings are $p_m \approx 0.001$ (De Jong 1975), $p_m \in [0.005, 0.01]$ (Schaffer et al. 1989), $p_m \approx 0.01$ (Grefenstette 1986)).
(3) The algorithm uses a *recombination* (*crossover*) operator that exchanges arbitrary substrings between two individuals with probability p_c (e.g., $p_c \approx 0.6$ (De Jong 1975), $p_c \in [0.75, 0.95]$ (Schaffer et al. 1989), $p_c \approx 0.95$ (Grefenstette 1986)). Length and position of these substrings are chosen at random, but are identical for both individuals.
(4) The probabilistic selection operator forms the next generation by copying individuals on the basis of fitness-proportionate probabilities $p_i = \Phi(a_i)/\sum_{j=1}^{\mu} \Phi(a_j)$, where μ denotes the population size and $\Phi : \mathbb{B}^l \to \mathbb{R}$ is the fitness function (we assume a maximization task and positive fitness in the following in connection with proportional selection, such that no *scaling* mechanism is necessary).

Theoretically, GAs are usually analyzed by viewing them as algorithms that process *schemata*, i.e. descriptions of hyperplanes in l-dimensional bit space. The fundamental *Schema Theorem* explains the power of GAs by an exponential growth of relatively short, useful substrings — so-called *building blocks* — and their accumulation and concatenation to useful substrings of increasing length (Holland 1975). The choice of a binary alphabet is supported by the fact that this maximizes the total number of schemata available for processing, i.e. consideration of the schema theorem, the building block hypothesis, and a minimal alphabet are seen as the main design criteria for applying a GA to an optimization problem (see (Goldberg 1989), pp. 28–42, pp. 80–82).

Using a binary alphabet, only *pseudoboolean optimization problems* $f : \mathbb{B}^l \rightarrow \mathbb{R}$ can be handled directly by a GA (i.e. $\Phi = f$), while other problems of the general form $f : M \rightarrow \mathbb{R}$ require the development of a code that in principle allows the decoding of a bit string to a representation of an element of the problem space M. For this reason, it is useful to distinguish between the decoding function $\Gamma : \mathbb{B}^l \rightarrow M$ and the objective function $f : M \rightarrow \mathbb{R}$, such that $\Phi = f \circ \Gamma$. Both functions play an important role in the application of GAs to optimization problems. A formal mathematical description as presented in the remainder of this introduction seems useful for two reasons: First, the often informal, intuitive, and even distributed descriptions presented in GA literature offer ambiguities. Second, the formal description helps to identify and characterize special mappings which are needed in the remainder of this paper.

While Holland pointed out how to apply the GA to parameter optimization problems of the form $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (see (Holland 1975), pp. 54–58), K. De Jong was the first who realized this important application as an optimization algorithm (De Jong 1975). The binary decoding function Γ requires a constrained codomain $M = \times_{i=1}^{n}[u_i, v_i]$ ($u_i < v_i$) and logically divides the bit string into n segments of (in most cases) equal length l_x, thus implying $l = n \cdot l_x$. Each segment is interpreted as the binary code of the corresponding object variable $x_i \in [u_i, v_i]$ by applying a segment decoding function $\Gamma_i : \mathbb{B}^{l_x} \rightarrow [u_i, v_i]$:

$$\Gamma_i(a_{i1} \ldots a_{il_x}) = u_i + \frac{v_i - u_i}{2^{l_x} - 1} \cdot \left(\sum_{j=0}^{l_x - 1} a_{i(l_x - j)} \cdot 2^j \right) \quad , \qquad (1)$$

where $(a_{i1} \ldots a_{il_x})$ denotes the i-th segment of an individual $a \in \mathbb{B}^l$. Then, $\Gamma = \Gamma_1 \times \ldots \times \Gamma_n$ yields a vector of real values by interpreting the bit string as a concatenation of binary encoded integers and by linearly mapping each integer to the desired range $[u_i, v_i]$.

Nowadays, besides the simple code presented in equation (1) a *Gray code* interpretation of the bit string is often used for decoding purposes. This works by first converting a string segment $(b_1 \ldots b_{l_x})$ (for reasons of simplicity we omit the index which denotes the segment number) to the simple code $(a_1 \ldots a_{l_x})$ by means of a mapping $\gamma^{-1} : \mathbb{B}^{l_x} \rightarrow \mathbb{B}^{l_x}$ such that $\forall i \in \{1, \ldots, l_x\}$ (see (Wright 1991)):

$$a_i = \bigoplus_{j=1}^{i} b_j \quad , \qquad (2)$$

where $\oplus$ denotes addition modulo 2. Conversely, the standard binary code can be converted to Gray code by the mapping γ, where $\forall i \in \{1, \ldots, l_x\}$ (Wright 1991):

$$b_i = \begin{cases} a_i & , \text{ if } i = 1 \\ a_{i-1} \oplus a_i & , \text{ if } i > 1 \end{cases} \quad . \qquad (3)$$

Indicated by several researchers, the main advantage of a Gray code is seen in the fact that it maps Euclidean neighborhoods into Hamming neighborhoods due to the representation of adjacent integers by bit strings of Hamming distance one (see (Bethke 1981), pp. 100–104). This was supported by empirical comparisons between standard code and Gray code, indicating statistically significant advantages for the latter (see e.g. (Caruna and Schaffer 1988)). Gray code is the default mechanism in the important public domain implementation of (Grefenstette 1987).

The impact of the highly nonlinear binary representation on the search process, however, was not investigated in a systematical way after the general acceptance of a Gray code. To shed some light on this topic, we will investigate the mutation operator with respect to its role in case of a simple pseudoboolean optimization problem (using no coding at all, i.e. $\Phi = f$), a parameter optimization problem using the simple binary code (i.e. $\Phi = \Gamma$), and a parameter optimization problem using Gray code (i.e. $\Phi = \Gamma_{\text{Gray}}$).

2 MUTATION

Rather than looking at schemata, the mutation operator $m : \mathbb{B}^l \rightarrow \mathbb{B}^l$ is analyzed here with respect to its *success probability* when optimizing a particular fitness function $\Phi : \mathbb{B}^l \rightarrow \mathbb{R}$. The success probability $p^+_{\Phi(a)}(p)$, where $a \in \mathbb{B}^l$ denotes an individual and p is the mutation probability per bit, is defined here according to

$$p^+_{\Phi(a)}(p) = \mathcal{P}\{\Phi(m(a)) > \Phi(a)\} \quad , \qquad (4)$$

i.e. the probability of improvement of a bit string by mutation. Of course, the success probability depends on both the mutation rate and the actual fitness function value. In the following, the basic question to be answered is: *How can we maximize the success probability during the course of evolution ?*

This question seems reasonable both for unimodal and for multimodal problems. In the former case, maximizing the success probability also maximizes the convergence velocity, i.e. the expectation value of the improvement per generation (see (Bäck 1992a) for an in-depth analysis of convergence velocity). In the latter case, however, maximizing the success probability optimizes the chance to leave a local optimum for any local optimum, though this chance may become arbitrarily small (e.g., for a one-step transition from a local optimum to the isolated global optimum). These arguments justify the advantage of asking for the actually maximal success probability.

The subsequent sections address the maximization of success probabilities with respect to the three different objective functions as indicated above.

2.1 COUNTING ONES

The simple counting problem $f : \mathbb{B}^l \to \{0, \ldots, l\}$:

$$f(a_1 \ldots a_l) \quad = \quad \sum_{i=1}^{l} a_i \tag{5}$$

was analyzed independently by Mühlenbein and Bäck, resulting in the approximation

$$p^+_{f(a)}(p) \quad \approx \quad (1-p)^{f(a)} \left(1 - (1-p)^{l-f(a)}\right) \tag{6}$$

presented by Mühlenbein (see (Mühlenbein 1992)) and the exact expression

$$\begin{aligned} p^+_{f(a)}(p) \quad = \quad & \sum_{i=0}^{f(a)} \binom{f(a)}{i} p^i (1-p)^{f(a)-i} \\ & \cdot \sum_{j=i+1}^{l-f(a)} \binom{l-f(a)}{j} p^j (1-p)^{l-f(a)-j} \end{aligned} \tag{7}$$

presented by Bäck (see (Bäck 1992a)).

The optimal mutation rate p^*, such that $p^+_{f(a)}(p)$ is maximized for $p = p^*$, is approximated by Mühlenbein to a value of

$$p^*(f(a)) \quad \approx \quad 1 - (f(a)/l)^{1/(l-f(a))} \approx 1/l \tag{8}$$

(denoted Appr. (A) in figure 1). The exact expression (7), however, can not be solved analytically but was optimized numerically by Bäck, resulting in a curve that can well be fitted by the expression

$$p^*(f(a)) \approx \frac{1}{2(f(a)+1) - l} \tag{9}$$

(denoted Appr. (B) in figure 1; $f(a) \geq l/2$ is assumed in expression (9)). Both curves for $p^*(f(a))$ are shown in the left part of figure 1.

Though the resulting mutation rate schedules differ remarkably, it can be shown that the impact on the expected time (i.e. number of applications of the mutation operator) to locate the optimal string by a (1+1)-GA (an algorithm that modifies one parent bit string by means of mutation and retains the better of parent and offspring) is relatively small.

Assuming an initial string with $f(a) = \lfloor l/2 \rfloor$, Mühlenbein derives an expected time $\langle T(p^*) \rangle \approx \mathrm{e} \cdot l \cdot \ln(l/2)$ for $p^* = 1/l$, and he indicates that the $\mathcal{O}(l \cdot \ln l)$ estimate is valid for any unimodal pseudoboolean function (Mühlenbein 1992). The expected time resulting from the alternative approximation by Bäck can only be calculated numerically by means of Markov chain methods (see (Goodman 1988), pp. 157–162). For $l \leq 100$ the resulting time to absorption (i.e. until the algorithm has found the optimum) is shown for both approximations in the right part of figure 1.

The most time-consuming part of optimization consists in fine-tuning of the last few bits, such that these remarkably different schedules spend most time in almost optimal states of the Markov chain. This fact is reflected by the small differences concerning the expected time to absorption. Furthermore, the optimal schedule of the mutation rate is characterized by a continuous, smooth curve.

2.2 STANDARD CODE

The second example of an objective function is obtained as a simplification of the standard code presented in equation (1), i.e. $g : \mathbb{B}^l \to \{0, \ldots, 2^l - 1\}$:

$$g(a_1 \ldots a_l) \quad = \quad \sum_{i=0}^{l-1} a_{l-i} \cdot 2^i \quad , \tag{10}$$

such that $g = \Gamma$ for $n = 1$, $u = 0$, $v = 2^l - 1$. Since the assumption of a global optimum located at the string consisting completely of one bits is too restrictive, g is generalized here to a function g', where

$$g'(a) \quad = \quad \begin{cases} g(a) & , \text{ if } g(a) \leq w \\ -1 & , \text{ if } g(a) > w \end{cases} \tag{11}$$

for arbitrary $w \in \{1, \ldots, 2^l - 1\}$. g' attains its optimal value w for the bit string a^* that represents w.

The success probability $p^+_{g'(a)}(p) = \mathcal{P}\{g'(m(a)) > g'(a)\}$ can be obtained by summing over all possible improvements:

$$\begin{aligned} p^+_{g'(a)}(p) \quad &= \quad \sum_{i=g'(a)+1}^{w} \mathcal{P}\{g'(m(a)) = i\} \\ &= \quad \sum_{i=g'(a)+1}^{w} \mathcal{P}\{g(m(a)) = i\} \\ &= \quad \sum_{i=g'(a)+1}^{w} \mathcal{P}\{m(a) = g^{-1}(i)\} \quad . \end{aligned} \tag{12}$$

By using the fact that the restriction of g' to $\{0, \ldots, w\}$ is invertible, the *transition probabilities* $\mathcal{P}\{g(m(a)) = i\}$ are reduced to transition probabilities $\mathcal{P}\{m(a) = g^{-1}(i)\}$ on bit strings, which can in general be calculated according to

$$\mathcal{P}\{m(a) = b\} \quad = \quad (1-p)^{l-H(a,b)} \cdot p^{H(a,b)} \quad , \tag{13}$$

where $H(a, b)$ denotes the Hamming distance of a and b. The general transition probability (13) is maximized for $p^* = H(a, b)/l$.

For small l the optimal mutation rate $p^*(g'(a))$ can be calculated numerically to obtain a graphic comparable

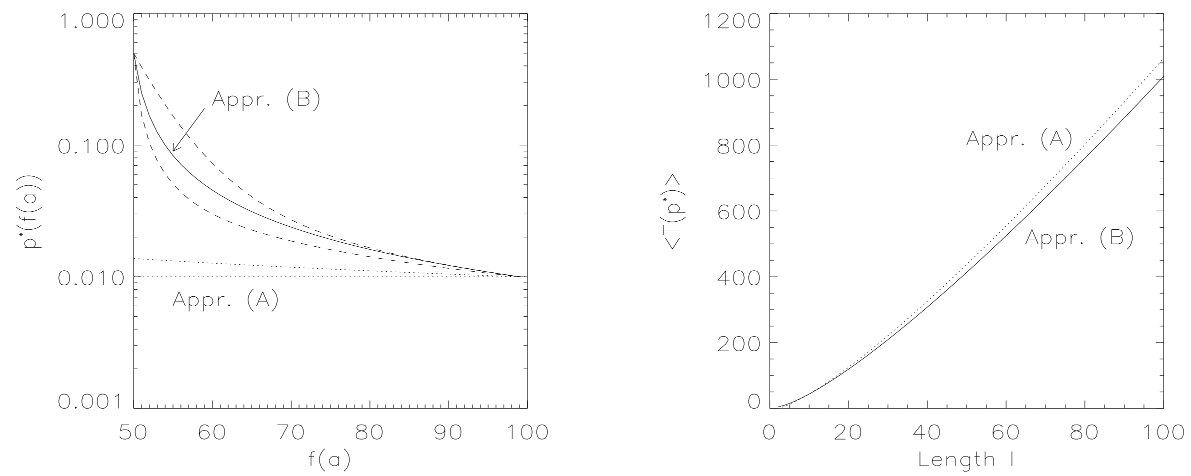

Figure 1: Left: Approximations of the Optimal Mutation Rate Schedules Obtained by Mühlenbein (Appr. (A)) and Bäck (Appr. (B)) for $l = 100$. Right: The Corresponding Expected Time to Absorption for Different String Lengths $2 \leq l \leq 100$.

to the left part of figure 1. With $l = 10$, $w = 2^l - 1$ (i.e. $g' = g$) and $g'(a)$ varying from 512 to 1022, the mutation rate schedule shown in the left part of figure 2 is obtained.

The optimal schedule is no longer continuous, but clearly reflects the Hamming cliffs occurring periodically. This is illustrated most impressively for the cliff observed between $p^*(767) = 0.5$ and $p^*(768) = 0.1822$. The corresponding bit strings `1011111111` and `1000000000` reflect the necessity to change many bits simultaneously to overcome the Hamming cliff. However, the situation is not as bad as it would be if transition probabilities were considered instead of success probabilities: To maximize the transition probability $\mathcal{P}\{m(g^{-1}(767)) = g^{-1}(768)\}$, a mutation rate $p = 0.9$ would be optimal — but still yields a very small transition probability of $3.87 \cdot 10^{-2}$. In contrast to this, the success probability $p^+_{g(a)=767}(0.5) = 0.249$ is remarkably large.

2.3 GRAY CODE

It is even more interesting to investigate an objective function $h : \mathbb{B}^l \longrightarrow \{0, \ldots, 2^l - 1\}$:

$$h(a) \quad = \quad g(\gamma^{-1}(a)) \tag{14}$$

that interpretes the bit string as Gray-coded integer, converts it to the corresponding binary code, and applies g. The interdependencies of g, h, and γ are summarized for clarity in the commutative diagram shown in the right part of figure 2.

As before, we modify h to a function h' where

$$h'(a) \quad = \quad \begin{cases} h(a) & \text{, if } h(a) \leq w \\ -1 & \text{, if } h(a) > w \end{cases} \tag{15}$$

for arbitrary $w \in \{1, \ldots, 2^l - 1\}$. The success probabilities can be calculated in analogy to equation (12) according to

$$p^+_{h'(a)}(p) \quad = \quad \sum_{i=h'(a)+1}^{w} \mathcal{P}\{m(a) = \gamma(g^{-1}(i))\} \quad . \tag{16}$$

For the special case $w = 2^l - 1$ (i.e. $h = h'$) an inductive argument based on the observation that γ and γ^{-1} do not change the most significant bit yields the result $p^*(g(a)) = p^*(h(a))$ for all a (see figure 2, left). In this case, the choice of a Gray code has no advantage compared to the standard code.

Chosing $w < 2^l - 1$, however, reveals some quantitative differences between both codes with respect to maximization of success probabilities. These differences are illustrated in figure 3 for $l = 10$, an arbitrarily chosen value $w = 822$, and $g'(a), h'(a) \in \{512, \ldots, 821\}$. The figure shows plots of the optimal mutation rate depending on the objective function value for the standard binary code (left) and the Gray code (right).

Sharp Hamming cliffs are identified for both objective functions, but the standard code shows a remarkable behaviour in the range of objective function values around 767, where the most noticeable Hamming cliff was already identified in figure 2. In this range of function values the success probability $p^+_{g(a)}$ becomes a *multimodal* function of p, having a local maximum for

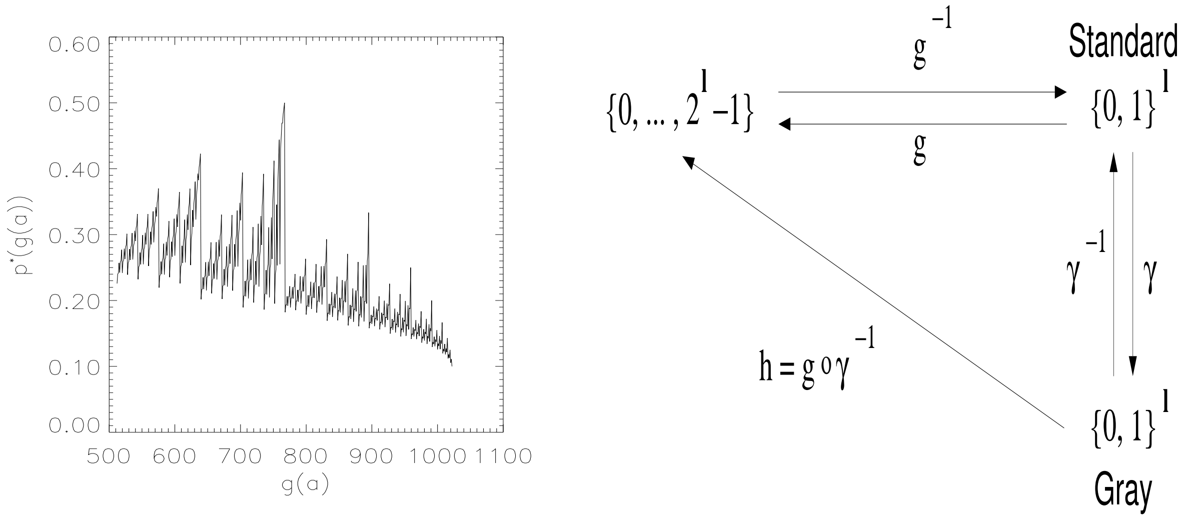

Figure 2: Left: Optimal Mutation Rate Schedule $p^*(g(a))$ for $l = 10$. Right: Diagram Demonstrating the Interdependencies of the Mappings g, h, and γ.

relatively small p and attaining the global maximum for a large value of p. To understand the situation, we assume an actual position $g^{-1}(766) = \texttt{1011111110}$ and $w = 769$, resulting in the transition probabilities summarized in table 1.

Table 1: Example of Transition Probabilities that Cause Multimodality of the Success Probability Function.

i	$g^{-1}(i)$	$\mathcal{P}\{m(g^{-1}(766)) = g^{-1}(i)\}$
767	1011111111	$p \cdot (1-p)^9$
768	1100000000	$p^8 \cdot (1-p)^2$
769	1100000001	$p^9 \cdot (1-p)$

The first derivative of the resulting polynomial $p^+_{g'(766)} = p(1-p)^9 + p^8(1-p)^2 + p^9(1-p)$ equals zero for

- $p_1 = 0.10000$ ($p^+_{g'(766)}(p_1) = 0.0387$),
- $p_2 = 0.47753$ ($p^+_{g'(766)}(p_2) = 0.0028$), and
- $p_3 = 0.88890$ ($p^+_{g'(766)}(p_3) = 0.0453$).

This anomaly can be explained by looking at the bit strings, where a one-bit mutation would allow an improvement to 767, while a many-bit mutation is necessary to cross the Hamming cliff. Such multimodalities of the success probability function seemingly do not exist when a Gray code is used. A second difference of both graphics concerns the proximity of w, where the Gray code shows smaller fluctuations of the optimal mutation rate than the standard code, thus reflecting the observation that fine-tuning of almost optimal solutions is simplified by a Gray code.

Indeed, the fundamental difference between the transition from g to g' and those from h to h' consists in the fact that for $w < 2^l - 1$ the function g' may become a multimodal pseudoboolean function while h' remains unimodal. A local maximum $\hat{f} = f(\hat{a})$ of a pseudoboolean function $f : \mathbb{B}^l \to \mathbb{R}$ is defined by means of the $\mathcal{N}_1$-neighborhood

$$\mathcal{N}_1(a) \quad = \quad \{b \in \mathbb{B}^l \mid H(a,b) = 1\} \tag{17}$$

by requiring $\forall b \in \mathcal{N}_1(\hat{a})$: $\hat{f} > f(b)$ (see e.g. (Antamoshkin et al. 1990)). It is easy to see that h and h' are unimodal by definition of the Gray code, since for any $a \in \mathbb{B}^l$ ($a \neq a^*$, where a^* denotes the global optimum) the Gray code assures existence of a $b \in \mathcal{N}_1(a)$ such that $g(\gamma^{-1}(b)) = g(\gamma^{-1}(a)) + 1$. The function g is unimodal due to the fact that changing a single zero to one assures an improvement. To see that g' is multimodal, we notice that $g'^{-1}(767) = \texttt{1011111111}$ is a local maximum since no change of a single bit yields an improvement (remember that $g'(\texttt{1111111111}) = -1$).

These observations are confirmed by some experimental investigations concerning the time to absorption needed for g' and h' with optimal mutation rate schedule and a constant mutation rate of $1/l$, respectively. Experimental investigations are necessary in this case due to the fact that the Markov chain has 2^l states for g' and h' rather than l as for the function f discussed in section 2.1.

The results are collected in table 2 for values of $l \in \{8, \dots, 12\}$ and initial objective function values 2^{l-1}

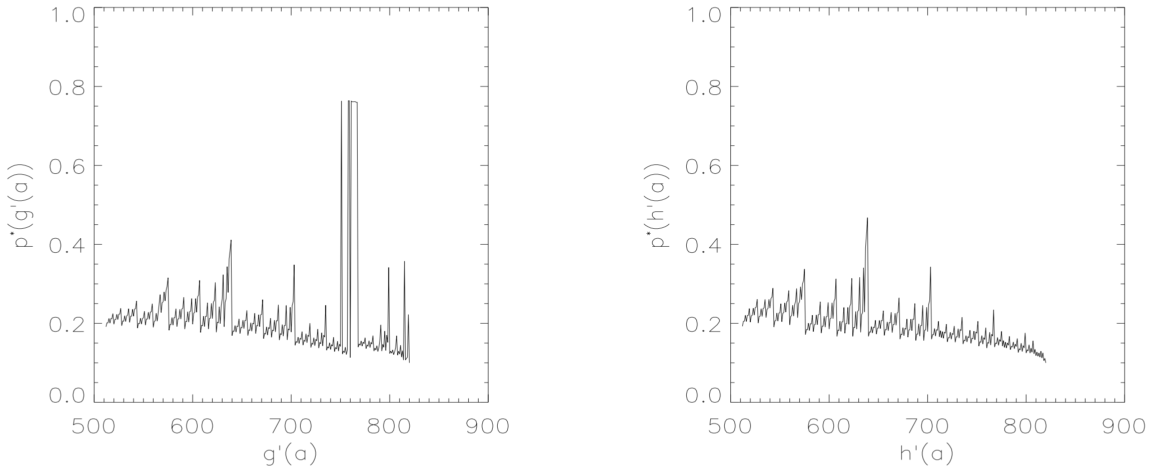

Figure 3: Left: Optimal Mutation Rate Schedule $p^*(g'(a))$ for the Standard Code. Right: Optimal Mutation Rate Schedule $p^*(h'(a))$ for the Gray Code. $l = 10$ and $w = 822$ are Chosen in Both Cases.

in each case. The time values are averaged over 100 experiments, each.

Table 2: Experimental Results Concerning the Time to Absorption for g' and h' with Optimal Mutation Rate Schedules (p^*) and Constant Mutation Rate ($p = 1/l$) for Different String Lengths l.

l	w	g'		h'	
		$\bar{T}(p^*)$	$\bar{T}(1/l)$	$\bar{T}(p^*)$	$\bar{T}(1/l)$
8	206	79.1	812.3	56.5	76.4
9	411	87.6	1745.4	71.2	86.4
10	822	221.3	4227.1	89.8	132.4
11	1644	576.4	6403.5	132.6	156.0
12	3288	2160.5	34225.1	157.5	187.1

The results for g' clearly reflect the problem to overcome the local optimum in case of the fixed mutation rate, while the optimal schedule simplifies this problem remarkably. The Gray code, however, still enables the algorithm to find the optimal solution within time complexity $\mathcal{O}(l \cdot \ln(l))$ and benefits just slightly from the optimal mutation rate schedule.

3 CONCLUSIONS

The results presented here help to explain the observations reported in literature that a time-dependent variation of the mutation rate may accelerate optimization by means of a Genetic Algorithm (Fogarty 1989; Hesser and Männer 1991; Bäck 1992b). Furthermore, the results enable us to distinguish two classes of fitness functions $\Phi : \mathbb{B}^l \rightarrow \mathbb{R}$: Unimodal and multimodal *pseudoboolean* functions, causing a shift of emphasis from unimodality or multimodality of the objective function to the properties of the fitness function.

As long as Φ is a unimodal pseudoboolean function, the results indicate that a mutation rate $p = 1/l$ is the best choice and a mutation rate schedule decreasing towards $1/l$ during optimization would only slightly accelerate the search. Whenever the objective function is unimodal, a Gray code assures unimodality of the fitness function and should therefore be the best choice.

However, when the fitness function becomes multimodal — this may happen by chosing a standard binary code or in case of a multimodal objective function — our observations indicate that the search for a mutation rate control different from a constant value $1/l$ may be worthwhile in order to overcome local optima. This is likely to be still valid when crossover is introduced, since one-point crossover in case of an encoded parameter optimization problem corresponds to a crossover at a segment boundary plus a mutation-like perturbation of the parameter within whose segment the crossover position is located (Wright 1991). If the number of crossover points is much smaller than the dimension of the objective function (which is quite usual), the effect of crossover is expected to be small. Preliminary experimental investigations seem to confirm this suspicion (see (Hoffmeister and Bäck 1992), pp. 64–66).

For the hard case of a multimodal fitness function Φ (where multimodality may be caused by either the decoding function, the objective function, or both) the results reported here may be interpreted as an explanation of the usefulness of a self-adaptation mechanism for mutation rates as described in (Bäck 1992b), where a remarkable diversity of mutation rates exists in a population of individuals. Further investigations on this mechanism are necessary in order to evaluate its general usefulness.

Acknowledgements

The author gratefully acknowledges support by the DFG (Deutsche Forschungsgemeinschaft), grant Schw 361/5-1.

References

Antamoshkin, A. N., V. N. Saraev, and E. S. Semenkin (1990). Optimization of unimodal monotone pseudoboolean functions. *Kybernetika 26*(5), 432–442.

Bäck, T. (1992a). The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Männer and Manderick (1992), pp. 85–94.

Bäck, T. (1992b). Self–Adaptation in Genetic Algorithms. In F. J. Varela and P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life*, pp. 263–271. The MIT Press, Cambridge, MA.

Bäck, T. and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation 1*(1), 1–23.

Bethke, A. D. (1981). *Genetic algorithms as function optimizers.* Ph. D. thesis, University of Michigan. Diss. Abstr. Int. 41(9), 3503B, University Microfilms No. 8106101.

Caruna, R. A. and J. D. Schaffer (1988). Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In J. Laird (Ed.), *Proceedings of the 5th International Conference on Machine Learning*, pp. 153–161. Morgan Kaufmann Publishers, San Mateo, CA.

Fogarty, T. C. (1989). Varying the probability of mutation in the genetic algorithm. In Schaffer (1989), pp. 104–109.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, Reading, MA.

Goodman, R. (1988). *Introduction to Stochastic Models.* The Benjamin/Cummings Publishing Company, Menlo Park, CA.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics SMC–16*(1), 122–128.

Grefenstette, J. J. (1987). *A User's Guide to GENESIS.* Washington, D. C.: Navy Center for Applied Research in Artificial Intelligence.

Hesser, J. and R. Männer (1991). Towards an optimal mutation probability in genetic algorithms. In H.-P. Schwefel and R. Männer (Eds.), *Parallel Problem Solving from Nature — Proceedings 1st Workshop PPSN I*, Volume 496 of *Lecture Notes in Computer Science*, pp. 23–32. Springer, Berlin.

Hoffmeister, F. and T. Bäck (1992). Genetic algorithms and evolution strategies: Similarities and differences. Report of the Systems Analysis Research Group SYS–1/92, University of Dortmund, Department of Computer Science.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: The University of Michigan Press.

De Jong, K. A. (1975). *An analysis of the behaviour of a class of genetic adaptive systems.* Ph. D. thesis, University of Michigan. Diss. Abstr. Int. 36(10), 5140B, University Microfilms No. 76–9381.

Männer, R. and B. Manderick (Eds.) (1992). *Parallel Problem Solving from Nature 2.* Elsevier, Amsterdam.

Mühlenbein, H. (1992). How genetic algorithms really work: I. mutation and hillclimbing. In Männer and Manderick (1992), pp. 15–25.

Schaffer, J. D. (Ed.) (1989). *Proceedings of the 3rd International Conference on Genetic Algorithms and Their Applications.* Morgan Kaufmann Publishers, San Mateo, CA.

Schaffer, J. D., R. A. Caruana, L. J. Eshelman, and R. Das (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer (1989), pp. 51–60.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms*, pp. 205–218. Morgan Kaufmann Publishers, San Mateo, CA.