UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

On the Choice of the Mutation Probability for the (1+1) EA

Thomas Jansen   Ingo Wegener

No. CI-92/00

Technical Report   ISSN 1433-3325   August 2000

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

On the Choice of the Mutation Probability for the (1+1) EA*

Thomas Jansen and Ingo Wegener

FB 4, LS 2, Univ. Dortmund, 44221 Dortmund, Germany
{jansen, wegener}@ls2.cs.uni-dortmund.de

Abstract. When evolutionary algorithms are used for function optimization, they perform a heuristic search that is influenced by many parameters. Here, the choice of the mutation probability is investigated. It is shown for a non-trivial example function that the most recommended choice for the mutation probability $1/n$ is by far not optimal, i. e., it leads to a superpolynomial running time while another choice of the mutation probability leads to a search algorithm with expected polynomial running time. Furthermore, a simple evolutionary algorithm with an extremely simple dynamic mutation probability scheme is suggested to overcome the difficulty of finding a proper setting for the mutation probability.

1 Introduction

Evolutionary algorithms (EAs) are randomized search heuristics that are often applied to the task of function optimization. They are influenced by many parameters. Though EAs are in general assumed to be robust and more or less insensitive to the setting of the parameters, it is a well-known fact that the choice of the parameter settings has great impact on the success and efficiency of the search. We consider optimization by means of EAs and look for parameter settings that allow an efficient exact optimization of a given objective function that we consider to be unknown to the algorithm.

We concentrate on maximization of discrete functions and assume that the objective function is some function $f\colon \{0,1\}^n \rightarrow \mathbb{R}$. We consider the $(1+1)$ evolutionary algorithm ($(1+1)$ EA). It is a very simple EA using only mutation and selection. In the next section, we give a formal definition of the (1+1) EA and discuss known examples that demonstrate that the most recommended choice for the mutation probability is by far not optimal. These examples are not of practical relevance, since also the best mutation probability leads to an algorithm with an expected exponential running time. In Section 3, we present a function with the following properties. The (1+1) EA with a mutation probability which is substantially larger or smaller than $(\log n)/n$ needs superpolynomial time with overwhelming probability while mutation probabilities growing as $(\log n)/n$ lead to a (1+1) EA which finds the optimum in an expected polynomial number of

* This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (531).

steps and even in a polynomial number of steps with overwhelming probability. In Section 4, we introduce a variant of the $(1+1)$ EA that employs an extremely simple dynamic variation scheme for the mutation probability. We prove that this dynamic EA optimizes the example function efficiently. We finish with some concluding remarks.

2 The (1+1) EA

As already mentioned we concentrate on the maximization of functions $f: \{0,1\}^n \to \mathbb{R}$ by the (1+1) EA. This algorithm is often subject to theoretical studies [3, 5, 9, 11]. We present results using the common notions O, Ω, and Θ, where for two function $s, t: \mathbb{N} \to \mathbb{R}$ we say that $s(n) = O(t(n))$, if there exist constants $n_0 \in \mathbb{N}$ and $c \in \mathbb{R}^+$, such that for all $n \geq n_0$ we have that $s(n) \leq c \cdot t(n)$ holds. We say that $s(n) = \Omega(t(n))$, if $t(n) = O(s(n))$. Finally, we say $s(n) = \Theta(t(n))$ if both, $s(n) = O(t(n))$ and $s(n) = \Omega(t(n))$ hold.

Algorithm 1 (The (1+1) EA).

1. *Choose $x \in \{0,1\}^n$ uniformly at random.*
2. *Fix $p(n) \in (0, 1/2]$.*
3. *$y := x$*
4. *Flip each bit in y independently with probability $p(n)$.*
5. *If $f(y) \geq f(x)$, set $x := y$.*
6. *Continue at 3.*

It is a common experience that simple hill-climbers are often able to find solutions that are at least comparable to those of more sophisticated evolutionary algorithms [8]. Setting $p(n) = 1/n$ implies that during one mutation step on average one bit flips. Thus, the (1+1) EA may be regarded as a kind of randomized hill-climber. In fact, the most recommended fixed choice for the mutation probability $p(n)$ is $1/n$ [1, 9]. For linear functions choosing $p(n) = \Theta(1/n)$ can be proven to be optimal [3]. Sometimes it is conjectured that $p(n) = \Theta(1/n)$ may be optimal for all functions.

First of all, we present known examples where the choice $p(n) = 1/n$ for the mutation probability is much worse than the choice $p(n) = 1/2$. The choice $p(n) = 1/2$ changes the (1+1) EA into a random search. We only consider functions with a unique global maximum. Then the probability of choosing x_{opt} equals 2^{-n} for each step independently of the history. Hence, the expected running time equals 2^n.

The first example is the well-known "needle in a haystack" function, that yields the function value 0 for all $x \in \{0,1\}^n \setminus \{(1, \ldots, 1)\}$ and 1 else. A search algorithm gets no information until it finds the global optimum by chance. The mutation probability $p(n) = 1/n$ is worse than $p(n) = 1/2$, since it increases the probability to investigate the same strings again and again. Indeed, Garnier, Kallel, and Schoenauer [5] have shown that the expected running time of the (1+1) EA with $p(n) = 1/n$ equals $2^n/(1 - e^{-1}) \approx 1.582 \cdot 2^n$ and is by a constant factor larger than the expected running time of random search.

The second example is the "trap" function with $\text{TRAP}(x) = \text{ONEMAX}(x) = x_1 + \cdots + x_n$ for all $x \in \{0,1\}^n \setminus \{(0,\ldots,0)\}$ and $\text{TRAP}((0,\ldots,0)) = n+1$. Hence, TRAP differs from the linear function ONEMAX only for the all zero string which is optimal. A search strategy for ONEMAX should find the all one string quickly. As long as the all zero string is not found by chance, the (1+1) EA works on TRAP as on ONEMAX and, therefore, gets hints in the wrong direction. For this reason, TRAP is called a strongly deceptive function. Droste, Jansen, and Wegener [4] have proved that the expected number of steps of the (1+1) EA with $p(n) = 1/n$ on TRAP is $\Theta(n^n)$ which is by the exponential factor $\Theta((n/2)^n)$ larger than the expected time of random search. Random search is impractical even for rather small n. Hence, if random search beats an evolutionary algorithm for some function, both algorithms are too bad for this function. The algorithms will be stopped in applications before the global optimum is found.

We are looking for examples where the running time of the (1+1) EA can be drastically reduced to some polynomial number of steps when setting the mutation probability $p(n)$ to some other value than $1/n$. Such an example is given in the following section.

3 A Non-Trivial Example Function

Here, we introduce an objective function with the following properties. Setting $p(n) = 1/n$ implies that the (1+1) EA will almost surely need a superpolynomial number of steps to find a global maximum If $p(n)$ is set to another value, a global maximum is found within a polynomially bounded number of steps with high probability. The objective function we discuss combines some ideas that can partly be identified in other functions that were already subject of theoretical investigations. The first basic idea is that the need for a large "jump", i.e., the mutation of several bits simultaneously, causes difficulties for standard $1/n$-mutations. This is the key property of JUMP used by Jansen and Wegener [7] to provide an example where crossover is helpful. The second observation is, that analyzing the behavior of an EA may become substantially easier, if the function enforces that certain paths are followed almost surely. An example for that are ridge functions introduced (in a different context and for different reasons) by Quick, Rayward-Smith, and Smith [10].

For the sake of simplicity, we assume that $n = 2^k > 32$ implying that $n/4 \geq 2\log n$. Otherwise, appropriate rounding leads to substantially the same definitions and results. Throughout this paper by $\log n$ we denote the logarithm to the base 2, i. e. $\log_2 n$. In order to be able to present a well-structured and understandable definition we partition the search space $\{0,1\}^n$ into five disjoint sets, namely

$$
\begin{aligned}
A &:= \{x \in \{0,1\}^n \mid n/4 < ||x||_1 < 3n/4\}\,, \\
B &:= \{x \in \{0,1\}^n \mid ||x||_1 = n/4\}\,, \\
C &:= \left\{x \in \{0,1\}^n \mid \exists i \in \{0,1,\ldots,(n/4)-1\} : x = 1^i 0^{n-i}\right\}
\end{aligned}
$$

$$D := \left\{ x \in \{0,1\}^n \mid (||x||_1 = \log n) \wedge \left(\sum_{i=1}^{2\log n} x_i = 0 \right) \right\}, \text{ and}$$
$$E := \{0,1\}^n - (A \cup B \cup C \cup D),$$

where $||x||_1$ denotes the number of ones in x and $1^i 0^{n-i}$ denotes the string with i consecutive ones followed by $n-i$ consecutive zeros. Now, we can define the objective function that serves as our example.

Definition 1. *The function* PathToJump: $\{0,1\}^n \to \mathbb{R}$ *is defined by*

$$\text{PathToJump}(x) := \begin{cases} n - ||x||_1 & \text{if } x \in A, \\ (3/4)n + \sum_{i=1}^{n/4} x_i & \text{if } x \in B, \\ 2n - i & \text{if } x \in C \text{ and } x = 1^i 0^{n-i}, \\ 2n+1 & \text{if } x \in D, \\ \min\{||x||_1, n - ||x||_1\} & \text{if } x \in E. \end{cases}$$

All strings in D are globally optimal. In the rest of the paper we use the notation $X' < X''$ (X'' has a higher rank than X') to indicate that $f(x') < f(x'')$ for all $x' \in X'$ and $x'' \in X''$. The set X'' is globally better than X'. By definition, we obtain

$$E < A < B < C < D.$$

A typical run of the (1+1) EA whose mutation probability is not too large looks as follows. We start in A and, therefore, never reach E. The (1+1) EA almost works like on simple linear functions and has a good chance to reach B. Then it is unlikely to jump to $C \cup D$. Hence, improvements are made in B until the best string $1^{n/4}0^{3n/4}$ is reached there. Here we are close to C and follow the "path" given by the strings of C to the string 0^n. Finally, a jump to D is necessary. Here we need a mutation probability which is essentially larger than $1/n$.

First, we prove that with mutation probability $p(n) = 1/n$ with a probability close to 1 the (1+1) EA will not find a global optimum within a polynomial number of steps. In fact, we prove something stronger. We show that if the mutation probability $p(n)$ substantially differs from $\log n/n$, then the (1+1) EA needs a superpolynomial number of steps for optimizing PathToJump almost surely.

Theorem 1. *Let $\alpha(n) := p(n) \cdot n/\log n$. If $\alpha(n) \to 0$ as $n \to \infty$ or $\alpha(n) \to \infty$ as $n \to \infty$, the probability that the (1+1) EA with mutation probability $p(n)$ needs a superpolynomial number of steps to find a global optimum of* PathToJump *converges to 1.*

Proof. By Chernoff's bounds (Hagerup and Rüb [6]), the probability that the initial string belongs to A equals $1 - e^{-\Omega(n)}$. Hence, we can assume that the (1+1) EA starts in $A \cup B \cup C \cup D$ and never reaches strings in E.

Case 1: $\alpha(n) \to \infty$ as $n \to \infty$ We only investigate the last step where a string $x \in A \cup B \cup C$ is changed by mutation into a string $y \in D$ and estimate the probability of such a mutation. We know be definition that x contains at least $n/4$ zeros and in order to reach some $y \in D$ it is necessary that at most $\log n$ of these bits flip. The expected number of flipping bits among the chosen positions equals $(n/4) \cdot p(n) = (1/4)\alpha(n)\log n$. By Chernoff's bounds, the probability that at most $\log n$ of these bits flip is bounded above by $e^{-\Omega(\alpha(n)\log n)} = n^{-\Omega(\alpha(n))} \leq n^{-c\alpha(n)}$ for some $c > 0$. Hence, the expected running time of the (1+1) EA is at least $n^{c\alpha(n)}$ which grows superpolynomially in this case. Also the probability that $n^{c\alpha(n)/2}$ steps are sufficient is bounded above by $n^{-c\alpha(n)/2}$ which is superpolynomially small.

Case 2: $\alpha(n) \to 0$ as $n \to \infty$ Again we estimate the probability to obtain $y \in D$ by a mutation step from $x \in A \cup B \cup C$. By case inspection, we conclude that the Hamming distance between x and y is at least $\log n$. Here the assumption that $y_1 + \cdots + y_{2\log n} = 0$ for $y \in D$ is essential. Hence, it is necessary that at least $\log n$ bits flip simultaneously. The expected number of flipping bits equals $\alpha(n)\log n$. By Chernoff's bounds, the probability that the number of successes is by a factor of $\alpha(n)^{-1}$ larger than the expected value is bounded above by $\alpha(n)^{\Omega(\log n)} = n^{\Omega(\log \alpha(n)} \leq n^{c\log\alpha(n)}$ for some $c < \infty$. Remember that $\alpha(n) \to 0$ as $n \to \infty$. Hence, the expected running time of the (1+1) EA is at least $n^{-c\log\alpha(n)}$ which is superpolynomially increasing. Also, the probability that $n^{-(c/2)\log\alpha(n)}$ steps are sufficient is bounded above by $n^{(c/2)\log\alpha(n)}$ which is superpolynomially small. □

We remark that these lower bounds also hold for the $(\mu + \lambda)$ EA and the (μ, λ) EA. As long as $\mu = e^{o(n)}$, in particular, for populations of polynomial size, all initial strings belong to A with overwhelming probability and at least one string has to mutate in the way described in the two cases of the proof of Theorem 1.

In the following we investigate the case that $\alpha(n)$ is a constant. In order to simplify some calculations we switch from $\log n$ to $\ln n$ and assume that $p(n) = (c\ln n)/n$. We analyze the random number of steps of the (1+1) EA until an optimal string $y \in D$ is found and then we look for that constant c leading to the best result.

Theorem 2. *Let $p(n) = (c\ln n)/n$. The expected number of steps until the (1+1) EA with mutation probability $p(n)$ finds a global optimum of the function* PathToJump *is bounded above by $O(n^{2+c}\ln^{-1} n + n^{c-\log c-\log\ln 2})$. For $c = 1/(4\ln 2) \leq 0.361$ the expected running time is bounded by $O(n^{2.361})$.*

Proof. In the following we work with the following partitions of A, B, C, and E.

- $A_i = \{x \in A \mid ||x||_1 = i\}$, $n/4 < i < 3n/4$
- $B_i = \left\{x \in B \mid x_1 + \cdots + x_{n/4} = i\right\}$, $0 \leq i \leq n/4$
- $C_i = \left\{1^i 0^{n-i}\right\}$, $0 \leq i < n/4$
- $E_i = \{x \in E \mid ||x||_1 = i \text{ or } n - ||x||_1 = i\}$, $0 \leq i < n/4$

Then we have

$$E_0 < E_1 \cdots < E_{(n/4)-1} < A_{(3n/4)-1} < \cdots < A_{(n/4)+2} < A_{(n/4)+1} < B_0$$

$$< B_1 < \cdots < B_{(n/4)-1} < B_{n/4} < C_{(n/4)-1} < C_{(n/4)-2} < \cdots < C_1 < C_0 < D.$$

Our aim is to derive lower bounds α_i, β_i, γ_i, and ε_i for the probabilities that we reach within one step from $X \in A_i$, B_i, C_i, and E_i resp. a string y which is element of a set with a higher rank. These probabilities may be called success probabilities, since they measure the probability of increasing the fitness of the current string.

Claim 1: $\alpha_i = \Omega((\ln n)/n^c)$, $\varepsilon_i = \Omega((\ln n)/n^c)$

Proof. If $x \in A_i$, we have a success if exactly one of the $i > n/4$ 1-bits and none of the 0-bits flip. Hence,

$$\begin{aligned}\alpha_i &\geq i\frac{c\ln n}{n}\left(1-\frac{c\ln n}{n}\right)^{n-1} > \frac{1}{4}c\ln n\left(1-\frac{c\ln n}{n}\right)^{n-1} \\ &= \Omega\left(\ln n e^{-c\ln n}\right) = \Omega\left(\frac{\ln n}{n^c}\right).\end{aligned}$$

If $x \in E_i$, even at least $(3n/4)$ 1-bit mutations lead to a success.

Claim 2: $\beta_i = \Omega(((n/4)-i)^2(\ln^2 n)/n^{2+c})$, if $i < n/4$

Proof. If $x \in B_i$, $i < n/4$, x contains $(n/4)-i$ 0-bits among the first $n/4$ positions and $(n/4)-i$ 1-bits among the last $3n/4$ positions. We have a success if exactly one of these 0-bits and one of these 1-bits flip. Hence,

$$\begin{aligned}\beta_i &\geq \left(\frac{n}{4}-i\right)^2\left(\frac{c\ln n}{n}\right)^2\left(1-\frac{c\ln n}{n}\right)^{n-2} \\ &= \Omega\left(\left(\frac{n}{4}-i\right)\frac{\ln^2 n}{n^2}e^{-c\ln n}\right) = \Omega\left(\left(\frac{n}{4}-i\right)^2\frac{\ln^2 n}{n^{2+c}}\right).\end{aligned}$$

Claim 3: $\beta_{n/4} = \Omega((\ln n)/n^{1+c})$, $\gamma_i = \Omega((\ln n)/n^{1+c})$, if $i > 0$

Proof. If $x \in B_{n/4}$, $x = 1^{n/4}0^{3n/4}$ and, if $x \in C_i$, $x = 1^i 0^{n-i}$. We have a success if exactly the last 1-bit flips. Hence

$$\frac{c\ln n}{n}\left(1-\frac{c\ln n}{n}\right)^{n-1} = \Omega\left(\frac{\ln n}{n^{1+c}}\right)$$

is a lower bound for $\beta_{n/4}$ and γ_i.

Claim 4: $\gamma_0 = \Omega(n^{\log c + \log\ln 2 - c})$

Proof. If $x \in C_0$, $x = 0^n$. Hence, the success event contains exactly all events where exactly $\log n$ of the last $n - 2\log n$ bits flip. Hence

$$\begin{aligned}\gamma_0 &= \binom{n-2\log n}{\log n}\cdot\left(\frac{c\ln n}{n}\right)^{\log n}\cdot\left(1-\frac{c\ln n}{n}\right)^{n-\log n} \\ &\geq \left(\frac{n-2\log n}{\log n}\right)^{\log n}\cdot\left(\frac{c\ln n}{n}\right)^{\log n}\cdot\left(1-\frac{c\ln n}{n}\right)^{n}\cdot\left(1-\frac{c\ln n}{n}\right)^{-\log n}\end{aligned}$$

$$= \left(1 - \frac{2\log n}{n}\right)^{\log n} \cdot \left(\frac{c \ln n}{\log n}\right)^{\log n} \cdot \Omega\left(\frac{1}{n^c}\right) \cdot \Omega(1)$$
$$= \Omega(1) \cdot (c \ln 2)^{\log n} \cdot \Omega\left(\frac{1}{n^c}\right) \cdot \Omega(1) = \Omega\left(n^{\log c + \log \ln 2 - c}\right).$$

If the success probability of some event equals q, the expected waiting time for a success equals q^{-1}. In the worst case we start in E_0 and each success leads to a string of the next rank class. Hence, the expected running time $\mathrm{E}\,(T(c))$ can be estimated above by

$$\begin{aligned}
\mathrm{E}\,(T(c)) &\le \sum_{i=0}^{(n/4)-1} \varepsilon_i^{-1} + \sum_{i=(n/4)+1}^{(3n/4)-1} \alpha_i^{-1} + \sum_{i=0}^{(n/4)-1} \beta_i^{-1} + \beta_{n/4} + \sum_{i=1}^{(n/4)-1} \gamma_i^{-1} + \gamma_0^{-1} \\
&= O\left(n^{1+c} \ln^{-1} n\right) + O\left(n^{2+c}\left(\sum_{i=1}^{(n/4)-1} \frac{1}{i^2}\right) \ln^{-2} n\right) + O\left(n^{1+c} \ln^{-1} n\right) \\
&\quad + O\left(n^{2+c} \ln^{-1} n\right) + O\left(n^{c - \log c - \log \ln 2}\right) \\
&= O\left(n^{2+c} \ln^{-1} + n^{c - \log c - \log \ln 2}\right),
\end{aligned}$$

since the series $\sum(1/i^2)$ is converging.

In order to choose the best value for c we set

$$2 + c = c - \log c - \log \ln 2$$

which is equivalent to

$$\log c = -2 - \log \ln 2$$

or

$$c = \frac{1}{4 \ln 2} \le 0.361.$$

□

Theorem 2 shows that the appropriate mutation probability leads to an expected running time which is a polynomial of reasonable degree.

Theorem 3. *Let $p(n) = (c \ln n)/n$. The probability that the (1+1) EA with mutation probability $p(n)$ finds a global maximum of* PathToJump *within $O(n^{3+c} \ln^{-1} n + n^{1+c-\log c - \log \ln 2})$ steps ($O(n^{3.361})$ if $c = 1/(4 \ln 2)$) is bounded below by $1 - e^{-\Omega(n)}$.*

Proof. If the success probability of some event equals q, the probability of having no success within $\lceil q^{-1} n \rceil$ steps, equals

$$(1-q)^{\lceil q^{-1} n \rceil} \le (1-q)^{qn} = e^{-\Omega(n)}.$$

Here we have to wait for at most $(5/4)n$ successes with different success probabilities. The probability that we have to wait for at least one of these successes longer then n times the expected waiting time can be bounded by $(5/4)ne^{-\Omega(n)} = e^{-\Omega(n)}$. □

Our results show that we can gain a lot by choosing the appropriate mutation probability. The running time (the expected one and even a bound for the running time which holds with the overwhelming probability $1 - e^{-\Omega(n)}$) can decrease from superpolynomial to polynomial (with reasonable degree). But our results do not answer the question how to choose the right mutation probability. In black-box scenarios we do not know enough about the objective function to start any analysis and even in other scenarios the analysis is much too difficult to be carried out before starting the evolutionary algorithm.

4 The Dynamic (1+1) EA

We have seen that the choice of an appropriate mutation probability is advantageous and that we have in general no idea to compute or to estimate the appropriate mutation probability. Moreover, it can be even better to have different mutation probabilities in different phases of the algorithm. Hence, the algorithm has to "try" different mutation probabilities. Bäck [2] distinguishes a static parameter setting from three types of dynamic parameter settings. We choose the simplest one namely dynamic parameter control where the mutation probability only depends on the number of steps performed before. The more general schemes are adaptive parameter control which is guided by the success during the optimization process and self-adaptive parameter control where the change of the mutation probability is guided by an evolutionary algorithm.

Dynamic parameter control is easy to describe and implement. We choose $1/n$ as lower bound on the mutation probability since otherwise the expected number of flipping bits is less than one. We choose $1/2$ as upper bound since otherwise we flip on average more than half of the bits and do not search in the neighborhood of the current string. A phase consists of $\lceil \log n \rceil$ steps t, $1 \leq t \leq \lceil \log n \rceil$. The mutation probability in the t-th step of a phase equals $2^{t-1}/n$.

Algorithm 2 (The dynamic (1+1) EA). *The algorithm works as Algorithm 1 but the mutation probability in step s where $s = r \lceil \log n \rceil + t$, $1 \leq t \leq \lceil \log n \rceil$, equals $p_t(n) = 2^{t-1}/n$.*

Theorem 4. *The expected number of steps until the dynamic (1+1) EA finds a global optimum of the function* PathToJump *is bounded above by $O(n^2 \log n)$.*

Proof. We use an approach similar to that of the proof of Theorem 2. By α_i', β_i', γ_i', and ε_i' we denote the corresponding success probabilities for a whole phase of $\lceil \log n \rceil$ steps of the dynamic (1+1) EA.

Claim 1: $\alpha_i' = \Omega(1)$, $\varepsilon_i' = \Omega(1)$

Proof. We only consider the step with mutation probability $1/n$. Hence (compare the proof of Theorem 2),

$$\alpha_i' \geq i \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} > \frac{1}{4} \left(1 - \frac{1}{n}\right)^{n-1} = \Omega(1).$$

The result on ε_i' follows in a similar way.

Claim 2: $\beta'_i = \Omega(((n/4) - i)^2/n^2)$, if $i < n/4$

Proof. As in the proof of Theorem 2, we have $((n/4) - i)^2$ pairs such that the event that exactly the bits of a pair flip is a success. Hence, using the mutation probability $1/n$

$$\beta'_i \geq \left(\frac{n}{4} - i\right)^2 \left(\frac{1}{n}\right)^2 \left(1 - \frac{1}{n}\right)^{n-2} = \Omega\left(\left(\frac{n}{4} - i\right)^2 \frac{1}{n^2}\right).$$

Claim 3: $\beta'_{n/4} = \Omega(1/n)$, $\gamma_i = \Omega(1/n)$, if $i > 0$

Proof. There is a 1-bit mutation leading to a success. The claim follows for the mutation probability $1/n$.

Claim 4: $\gamma'_0 = \Omega(n^{\log \ln 2 - 1}) = \Omega(1/n^{1.53})$

Proof. The success event contains exactly all events where exactly $\log n$ of the last $n - 2\log n$ bits flip. Hence, mutation probabilities of the order $(\log n)/n$ are most promising. In the proof of Theorem 2 we have proved the bound $\Omega(n^{\log c + \log \ln 2 - c})$ for the mutation probability $(c \ln n)/n$. This bound is maximal, if $c = 1/\ln 2 \leq 1.45$. Then $\log c = -\log \ln 2$ and the bound equals $\Omega(n^{-1.45})$. But the dynamic (1+1) EA does not necessarily use exactly this mutation probability. We only now that we have for each interval $[(c \ln n)/n; (2c \ln n)/n]$ one mutation probability falling into this interval. Here it is optimal to choose $c = 1$ leading to a lower bound of $\Omega(n^{\log \ln 2 - 1}) = \Omega(n^{-1.53})$.

These claims lead to the following upper bound on the expected number $\mathrm{E}(P)$ of phases

$$\mathrm{E}(P) = O(n) + O\left(n^2 \sum_{i=1}^{(n/4)-1} \frac{1}{i^2}\right) + O\left(n^2\right) + O\left(n^{1.53}\right) = O\left(n^2\right).$$

The run time is bounded above by $\lceil \log n \rceil \cdot P$ which proves the claim. □

In the same way as we have obtained Theorem 3 from Theorem 2 we get the following result.

Theorem 5. *The probability that the dynamic (1+1) EA finds a global optimum of* PathToJump *within $O(n^3 \log n)$ steps is bounded below by $1 - e^{-\Omega(n)}$.*

The upper bounds of Theorem 4 and Theorem 5 are better than the corresponding bounds of Theorem 2 and Theorem 3. The dynamic (1+1) EA wastes some steps by using inappropriate mutation probabilities. But this only leads to an additional factor $O(\log n)$. The dynamic (1+1) EA does not always choose the optimal mutation probability (see the proof of Claim 4 in the proof of Theorem 4). This is not essential for PathToJump. But the dynamic (1+1) EA tries different mutation probabilities while the "optimal" static mutation probability is a compromise between optimal mutation probabilities for the Path-phase and the JUMP-phase. In order to rigorously prove that the dynamic (1+1) EA is better than each static (1+1) EA on PATHTOJUMP we need lower bounds for the static (1+1) EA. This implies bounds for the probability of jumping to strings whose rank is much larger than the current string. Such calculations are possible with our methods but we have not performed them yet.

5 Conclusions

We have presented the function PATHTOJUMP which has some interesting properties. The static (1+1) EA with mutation probabilities whose growth order is larger or smaller than $(\log n)/n$ needs superpolynomial time with overwhelming probability. But there are mutation probabilities such that the static (1+1) EA finds an optimum within an expected number of only $O(n^{2.361})$ steps. Since in general there is no idea how to compute an optimal or good value for the mutation probability, we have presented a dynamic (1+1) EA with a simple dynamic control of the mutation probability. This variant of the (1+1) EA finds an optimum of PATHTOJUMP within an expected number of $O(n^2 \log n)$ steps.

References

1. Bäck, T. (1993). Optimal mutation rates in genetic search. In Forrest, S. (Ed.): Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA '93), 2–8. Morgan Kaufmann.
2. Bäck, T. (1998). An overview of parameter control methods by self-adaptation in evolutionary algorithms. Fundamenta Informaticae 34, 1–15.
3. Droste, S., Jansen, T., and Wegener, I. (1998). A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs. Evolutionary Computation 6(2), 185–196.
4. Droste, S., Jansen, T., and Wegener, I. (1998). On the analysis of the (1+1) evolutionary algorithm. Tech. Report CI-21/98, Univ. Dortmund, Germany.
5. Garnier, J. Kallel, L., and Schoenauer, M. (1999). Rigorous hitting times for binary mutations. Evolutionary Computation 7(2), 173–203.
6. Hagerup, T., and Rüb, C. (1989). A guided tour of Chernoff bounds. Information Processing Letters 33, 305–308.
7. Jansen, T. and Wegener, I. (1999). On the analysis of evolutionary algorithms — A proof that crossover really can help. In Nešetřil, J. (Ed.): Proceedings of the 7th Annual European Symposium on Algorithms (ESA '99), 184–193. Springer.
8. Juels, A. and Wattenberg, M. (1994). Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Tech. Report CSD-94-834 Univ. of California.
9. Mühlenbein, H. (1992). How Genetic Algorithms Really Work. I. Mutation and Hillclimbing. In Männer, R. and Manderik, R. (Eds.): Parallel Problem Solving From Nature (PPSN II), 15–25. North-Holland.
10. Quick, R. J., Rayward-Smith, V. J., and Smith, G. D. (1998). Fitness distance correlation and ridge functions. In Eiben, A. E. Bäck, T., and Schwefel, H.-P. (Eds.): Parallel Problem Solving from Nature (PPSN V), 77–86. Springer.
11. Rudolph, G. (1997). Convergence Properties of Evolutionary Algorithms. Verlag Dr. Kovač.