Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem

Süntje Böttcher, Benjamin Doerr, and Frank Neumann

Algorithms and Complexity, Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. We reconsider a classical problem, namely how the (1+1) evolutionary algorithm optimizes the LeadingOnes function. We prove that if a mutation probability of p is used and the problem size is n, then the optimization time *is*

$$\tfrac{1}{2p^2}((1-p)^{-n+1} - (1-p)).$$

For the standard value of $p = 1/n$, this is approximately $0.86n^2$. As our bound shows, this mutation probability is not optimal: For $p \approx 1.59/n$, the optimization time drops by more than 16% to approximately $0.77n^2$.

Our method also allows to analyze mutation probabilities depending on the current fitness (as used in artificial immune systems). Again, we derive an exact expression. Analysing it, we find a fitness dependent mutation probability that yields an expected optimization time of approximately $0.68n^2$, another 12% improvement over the optimal mutation rate. In particular, this is the first example where an adaptive mutation rate provably speeds up the computation time. In a general context, these results suggest that the final word on mutation probabilities in evolutionary computation is not yet spoken.

1 Introduction

Evolutionary algorithms [1] are a class of randomized algorithms [2] that have found many applications in different problem domains. Understanding the behavior of this kind of algorithms is a challenging and difficult task due to different random components that are involved. Considering evolutionary algorithms as randomized algorithms from a theoretical point of view allows to analyze them with respect to their runtime behavior in a rigorous way.

Analyzing the runtime behavior of evolutionary algorithms has become a major branch in the theoretical analysis of these algorithms. Starting with results on simple pseudo-Boolean functions (see e. g. [3,4]), different results have been obtained for classical combinatorial optimization problems such as shortest paths, minimum spanning trees, or maximum matchings (see [5] for an overview).

Almost all results mentioned are asymptotic ones. In particular, most of the results give little information about the leading constants, which are of high interest when using such methods for realistic input sizes. There are only a few results that give at least give a bound on the leading constant.

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 1–10, 2010.

It is folklore that the (1+1) evolutionary algorithm finds the optimum of the most simple test function OneMax, counting the number of 1-bits in the bit-string of length n, in time at most $(1 + o(1))en \ln(n)$. This follows immediately from an elementary proof using coupon collector type arguments. However, the corresponding lower bound was only recently given in [6].

For linear functions, Jägersküpper was the first to give a bound including the constant, namely $(1+o(1))2.02en \ln(n)$. This was improved to $(1+o(1))1.39en \ln(n)$ in [7]. There also the $(1 - o(1))en \ln(n)$ lower bound for OneMax was extended to all linear functions with non-zero coefficients.

In this paper, we present an *exact* analysis for the function LeadingOnes introduced by Rudolph [8]. This function is one of the classical test problems and has been extensively studied (see e. g. [3,9,10]). Our analysis yields an exact formula for the expected number of iterations (*expected optimization time*) needed to find the optimum of the LeadingOnes function. Let n be the problem size, that is, the length of the bit-strings forming the search space, and let p be a mutation probability. The typical choice for the mutation probability is $p = 1/n$ [3], but our analysis works for all possible values.

Given n and p, we show that the expected optimization time is

$$\tfrac{1}{2p^2}((1-p)^{-n+1} - (1-p)).$$

From this formula, we see that the standard value for the mutation probability of $p = 1/n$ leads to an expected optimization time of less than $\frac{1}{2}(e-1)n^2 \approx 0.86n^2$, where the first expression is tight up to terms of order n. This improves the best known upper bound of en^2 and lower bound of $n^2/6$ given in [3].

Our formula also allows to determine the optimal mutation probability. This in particular shows that the standard mutation probability of $1/n$ is not optimal. For $p \approx 1.59/n$, the expected optimization time drops by more than 16% to approximately $0.77n^2$ (for n sufficiently large).

We should add that, while $p = 1/n$ is generally the preferred good choice for the mutation probability, there are examples known where mutation probabilities even having a different order of magnitude like $\Theta(\log(n)/n)$ are much better than $1/n$, see [11]. However, these example functions look custom-tailored to demonstrate this effect.

Our method also allows to analyze mutation probabilities depending on the current fitness. Such ideas have been used under the name *artificial immune systems* in [12,13]. However, contrary to the common belief that one should start the optimization process with a larger mutation probability and then successively reduce it, none of these works could show that varying the mutation probability yields an improvement. This is different in our analysis. If we choose the mutation probability to be the reciprocal of the current fitness value (that is, the number of leading ones), then the expected optimization time again reduces to a number T, which satisfies $\frac{1}{4}en^2 - \frac{1}{4}en \leq T \leq \frac{1}{4}en^2 + \frac{1}{4}en$. We do have an exact expression for T as well.

2 Problem and Algorithms

Our aim is to study the optimization behavior of a simple randomized search heuristic for the well-known pseudo-Boolean function $f = \textsc{LeadingOnes} : \{0,1\}^n \to \mathbb{N}_0$ defined by

$$f(x) = \textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

We investigate a simple baseline evolutionary algorithm called (1+1) EA. It works with a population size of 1 and produces in each iteration one offspring by mutation. If not worse, this offspring forms the new population. As mutation, we use standard bit mutation, that is, each bit is flipped independently with probability p. Note that many authors implicitly assume $p = 1/n$, whereas we do allow all values for p.

Algorithm 1 ((1+1) EA with mutation probability p)

1. *Choose $x \in \{0,1\}^n$ uniformly at random.*
2. *Flip each bit of x independently with probability p to produce an offspring y.*
3. *If $f(y) \geq f(x)$ then $x := y$.*
4. *Go to 2.*

For our theoretical investigations we consider the expected number of iterations until our algorithms produces an optimal solution for the LeadingOnes problem for the first time. This is called the *expected optimization time*. Note that the mutation rate p in Algorithm 1 is constant for given n, that is, it does not vary over time or with respect to the current fitness of the population.

It is a common belief that one can gain improvements by varying the mutation probability. In particular, it is said that in the early stages of the optimization process the mutation probability should be larger to faster approach the optimum, whereas at the end, it should be smaller to avoid large jumps that potentially lead away from the optimum.

Such a varying mutation probability could be implemented by making the mutation probability depend on the time the optimization process already lasts. A difficulty here is that one would need a good guess on the expected optimization time.

An alternative approach is to let the mutation probability depend on the current fitness value. Here, of course, one would need a guess on the maximum possible fitness. Still, this seems to be easier than guessing the expected optimization time.

To incorporate this concept into the (1+1) EA above, let $p : \{0, \dots, n\} \to (0,1)$ be a function. We think of $p_k := p(k)$ being the mutation probability used when we have a LeadingOnes-value of k. This leads to the following Algorithm 2, which we call Adaptive (1+1) EA.

Algorithm 2 (Adaptive (1+1) EA)
1. *Choose* $x \in \{0,1\}^n$ *uniformly at random.*
2. *Flip each bit of* x *independently with probability* $p_{f(x)}$ *to produce an offspring* y*.*
3. *If* $f(y) \geq f(x)$ *then* $x := y$*.*
4. *Go to 2.*

Algorithms such as artificial immune systems make use of such adaptive mutation rates. For some recent asymptotic results on the runtime of these algorithms, we refer to [12,13].

3 A Formula for the Expected Optimization Time

The key to our analysis (and the topic of this section) is noting that the expected optimization time can be fully described by the expected times needed to improve the fitness (conditional on a particular fitness level). The latter can easily be expressed via the (current) mutation probability.

3.1 Combining the States

We first exploit the fact that we start with a random individual. This allows a simplified view, namely that we do not have to regard the remaining expected optimization time for each possible individual, but only for a few random states.

Lemma 1. *Let* x *be a (random) individual produced by Algorithm 1 or 2 within a fixed number* t *of iterations. Let* $f(x)$ *denote its fitness. Then* $x_{f(x)+1} = 0$ *with probability one. For all* $i > f(x) + 1$*, we have* $\Pr[x_i = 1] = \Pr[x_i = 0] = \frac{1}{2}$ *independent from all other bits.*

Proof. Simple induction over the time t.

3.2 Improvement Times

As discussed above, the base of our analysis is (later) noting that the expected optimization time can be expressed in terms of the waiting times for an improvement. We now use Lemma 1 to, very elementarily, determine these waiting times relative to the current fitness.

Lemma 2. *Let* $0 \leq j < n$*. Let* $x \in \{0,1\}^n$ *be random subject to* $f(x) = \textsc{LeadingOnes} = j$*. Let* y *be the offspring of* x *obtained by mutation with some mutation probability* p_j*. Then the probability that* y *is strictly fitter than* x*, the improvement probability, is* $\Pr[f(y) > f(x)] = (1-p_j)^j p_j$*.*

Proof. Since $f(x) = j$, a mutation step increases the f-value if and only if the first j bits do not change and the $(j+1)$st bit does change. All other bits are irrelevant.

The actions of the different bits are independent. A bit flips with probability p_j and remains unchanged with probability $1-p_j$. Thus the probability of improving the f-value is $\Pr[f(y) > f(x)] = (1-p_j)^j p_j$.

Given this improvement probability, we can simply compute the waiting time for such an improvement. Denote by A_i the expected time needed to find an improvement given that the initial solution has a fitness of $n-i$ (that is, we are i levels from the optimum). We compute the A_i.

Theorem 1. *Let $x \in \{0,1\}^n$ be random with $f(x) < n$. Then the time $A_{n-f(x)}$ we need to wait for an improvement is*

$$A_{n-f(x)} = \frac{1}{\Pr[f(y) > f(x)]}.$$

Consequently, for all $1 \leq i \leq n$, we have

$$A_i = \frac{1}{(1-p_{n-i})^{n-i} p_{n-i}}.$$

Proof. Follows from elementary properties of the geometric distribution and the previous lemma.

3.3 Expected Optimization Time

We now express the expected optimization time in terms of the improvement times just determined. Note that since the latter depend on the (possibly varying) mutation probability, this immediately tells us how the mutation probability influences the expected optimization time.

We denote by T_{n-i} the expected number of steps needed to find the optimum starting from a random initial individual with f-value i. Obviously, at most $n-i$ improvements are necessary.

Lemma 3. *The time needed to find the optimum given a random solution with f-value $n-i$ is*

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j,$$

where A_i is the improvement time as defined in the previous subsection and $T_0 = 0$.

Proof. For $i = 0$, the time we need to finish obviously is $T_0 = 0$. For $0 < i \leq n$, the remaining time is given by the time for the next improvement, A_i, plus the expected remaining time conditional on this improvement. Let $\bar{x}$ denote the individual right after the improvement to an f-value of more than $n-i$. Since the bits $n-i+2, \ldots, n$ are still random, we have $\Pr[f(\bar{x}) = n-i+j] = 2^{-j}$ for $j = 1, \ldots, i-1$ and $\Pr[f(\bar{x}) = n] = 2^{-(i-1)}$. Since $T_0 = 0$, we may write

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j. \qquad \square$$

We now express the remaining optimization time T_i fully via the improvement times A_j.

Theorem 2. *The time needed to find the optimum given a random solution with f-value $n - i$ is*

$$T_i = A_i + \frac{1}{2}\sum_{j=1}^{i-1} A_j.$$

Proof. By induction we show that our claim

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j = A_i + \frac{1}{2}\sum_{j=1}^{i-1} A_j$$

holds for all $0 < i \leq n$.

For $i = 1$ we have $T_1 = A_1 + \frac{1}{2}\sum_{j=1}^{0} A_j$.

Assume that $T_k = A_k + \frac{1}{2}\sum_{j=1}^{k-1} A_j$ holds for all $0 < k \leq i$. Then

$$\begin{aligned}
T_{i+1} &= A_{i+1} + \sum_{j=0}^{i} 2^{j-(i+1)} T_j = A_{i+1} + \frac{1}{2}T_i + \frac{1}{2}\sum_{j=0}^{i-1} 2^{j-i} T_j \\
&= A_{i+1} + T_i - \frac{1}{2}T_i + \frac{1}{2}\sum_{j=0}^{i-1} 2^{j-i} T_j = A_{i+1} + T_i - \frac{1}{2}A_i \\
&= A_{i+1} + A_i + \frac{1}{2}\sum_{j=0}^{i-1} A_j - \frac{1}{2}A_i \\
&= A_{i+1} + \frac{1}{2}\sum_{j=1}^{i} A_j.
\end{aligned}$$

□

The bound of Theorem 2 matches our intuition that all improvements apart from the current one produce a waiting time only with probability 1/2.

4 The Optimal Fixed Mutation Rate

We first investigate the optimal choice for the mutation rate when working with a fixed mutation rate. Depending on the chosen mutation rate p, we compute the expected optimization time of the (1+1) EA. Remember that the time for an improvement for LeadingOnes is $A_i = \frac{1}{p}(1-p)^{i-n}$ and the overall expected optimization time is $T = \frac{1}{2}\sum_{i=1}^{n} A_i$. This leads to the following result.

Theorem 3. *The expected optimization time of (1+1) EA with fixed mutation rate p for* LeadingOnes *is*

$$T = \frac{1}{2p^2}\left[(1-p)^{1-n} - (1-p)\right].$$

Proof. Using our previous observation, we can calculate the expected optimization time directly and get

$$\begin{aligned} T &= \frac{1}{2}\sum_{i=1}^{n} A_i = \frac{1}{2}\sum_{i=1}^{n} \frac{1}{p}(1-p)^{i-n} \\ &= \frac{1}{2p}\sum_{i=0}^{n-1}(1-p)^{i-n+1} = \frac{1}{2p}(1-p)^{1-n}\sum_{i=0}^{n-1}(1-p)^{i} \\ &= \frac{1}{2p}(1-p)^{1-n}\cdot\frac{1-(1-p)^n}{1-(1-p)} \\ &= \frac{1}{2p^2}\left[(1-p)^{1-n}-(1-p)\right]. \end{aligned}$$

□

Based on the previous theorem, we can determine the optimal choice of p for (1+1) EA and LeadingOnes. To do this we compute the derivative of T with respect to p. In particular, we solve $\frac{d}{dp}T = 0$. This leads to

$$-\frac{1}{p^3}\left[(1-p)^{1-n}-(1-p)\right] + \frac{1}{2p^2}\left[-(1-n)(1-p)^{-n}+1\right] = 0.$$

We cannot solve this equation in an algebraic way. Therefore, we provide a numerical approximation. This gives that the optimal mutation rate p converges to a value around $\frac{1.5936}{n}$, which implies an expected optimization time of $\approx 0.77201n^2$ for n sufficiently large. Compared to this, the expected optimization time is $\approx 0.85914n^2$ when choosing $p = 1/n$. Therefore, the optimal mutation rate leads to an improvement of 16.1% compared to the standard choice.

5 The Optimal Adaptive Mutation Rate

After having investigated the optimal mutation rate for (1+1) EA, we want to examine whether an adaptive mutation rate which depends on the fitness of the currently best solution can lead to further improvements.

With our general framework we can optimize the remaining time T and examine the Adaptive (1+1) EA in the following.

As A_i is a function in p_i, the overall optimization time $T = \frac{1}{2}\sum_{i=1}^{n} A_i$ is a function in $(p_1, \ldots, p_n)$. Let $(p_1^*, \ldots, p_n^*)$ minimize T, thus $\frac{\partial}{\partial p_i}T(p_1^*, \ldots, p_n^*) = 0$. Note that the indices are renamed for the sake of simplicity. Here, p_i is the mutation probability if the currently best solution has fitness $f(x) = n - i$. In order to determine the optimal choice of the optimal adaptive mutation rates, we compute the partial derivatives according to the overall expected optimization time T.

We get

$$\begin{aligned}\frac{\partial}{\partial p_i}T(p_1,\ldots,p_n) &= \frac{\partial}{\partial p_i}\frac{1}{2}\sum_{i=1}^{n}A_i \\ &= \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial p_i}A_i\end{aligned}$$

In order to find the optimal value of p_i, we need to solve

$$\frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial p_i}A_i = 0.$$

Remember that A_i is the waiting time for one improvement, thus $A_i \geq 0$. Hence it follows that this equation holds if and only if $\frac{\partial}{\partial p_i}A_i = 0$ holds for all i. Thus we have to compute $\frac{\partial}{\partial p_i}A_i$.

Theorem 4. *Let $A_i = \frac{1}{p_i}(1-p_i)^{i-n}$. Then the optimal mutation rate is*

$$p_i = \frac{1}{n-i+1}.$$

Proof. We calculate the derivative

$$\frac{\partial}{\partial p_i}A_i = -\frac{1}{p_i^2}(1-p_i)^{i-n} - \frac{1}{p_i}(i-n)(1-p_i)^{i-n-1}$$

and get the optimal mutation rate by resolving

$$\begin{aligned}&-\frac{1}{p_i^2}(1-p_i)^{i-n} - \frac{1}{p_i}(i-n)(1-p_i)^{i-n-1} = 0 \\ \iff &-(1-p_i)^{i-n-1}\left[\frac{1-p_i}{p_i^2} + \frac{i-n}{p_i}\right] = 0 \\ \iff &\frac{1}{p_i^2} + \frac{i-n-1}{p_i} = 0 \\ \iff &1 + p_i(i-n-1) = 0.\end{aligned}$$

Hence, we get $p_i = \frac{1}{n-i+1}$. □

Thus we have shown that $p_i = \frac{1}{n-i+1}$ is an optimal mutation rate for the Adaptive (1+1) EA for optimizing LeadingOnes. We use this mutation rate for determining the expected optimization time for Adaptive (1+1) EA. First, we consider the time to reach an improvement in dependence of p_i.

The expected waiting time for an improvement is given by

$$\begin{aligned}A_i = \frac{1}{p_i}(1-p_i)^{i-n} &= \frac{1}{\frac{1}{n-i+1}}\left(1-\frac{1}{n-i+1}\right)^{i-n} \\ &= \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1)\,.\end{aligned}$$

Using this expression, we can compute almost matching upper and lower bounds on the expected optimization time.

Theorem 5. *Let $p_{f(x)} = \frac{1}{f(x)+1}$, then the expected optimization time of the Adaptive (1+1) EA is upper bounded by $\frac{e}{4}n^2 + \frac{e}{4}n$.*

Proof. Let $A_i = \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1)$. Then

$$\begin{aligned} T &= \sum_{i=0}^{n} 2^{i-n-1} T_i \\ &= \frac{1}{2}\sum_{i=1}^{n} A_i \\ &= \frac{1}{2}\sum_{i=1}^{n} \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1) \\ &\leq \frac{1}{2}\sum_{i=1}^{n} e(n-i+1) \\ &= \frac{e}{4}n^2 + \frac{e}{4}n. \end{aligned}$$

□

Similarly we can compute a lower bound for the expected optimization time.

Theorem 6. *Let $p_{f(x)} = \frac{1}{f(x)+1}$, then the expected optimization time of the Adaptive (1+1) EA is lower bounded by $\frac{e}{4}n^2 - \frac{e}{4}n$.*

Proof. Let $A_i = \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1)$. Then, as above,

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

□

Summarizing the results for the adaptive mutation rate, we get an improvement of 12.0% compared to the expected optimization time using the optimal constant mutation rate of approximately $p = \frac{1.59}{n}$ and an overall improvement of 20.9% to the expected optimization time with standard mutation rate $p = \frac{1}{n}$.

6 Conclusions

Most of the theoretical studies on the runtime behavior of evolutionary algorithms deal with asymptotic results. We have presented an exact analysis for

the (1+1) EA and the test function LeadingOnes. This in particular showed that one may speed up the computation by 16% when using the optimal mutation rate compared to the standard mutation rate of $1/n$. Furthermore, our investigations on the adaptive mutation rate show that a further improvement of 12% is possible when using such an approach. We are optimistic that further exact investigations will allow a deeper understanding of the right parameter setting for evolutionary algorithms.

References

1. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2007)
2. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
3. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. 276, 51–81 (2002)
4. Jansen, T., Wegener, I.: Evolutionary algorithms—how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans. Evolutionary Computation 5, 589–599 (2001)
5. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4, 281–293 (2007)
6. Doerr, B., Fouz, M., Witt, C.: Quasirandom evolutionary algorithms. In: Proceedings of GECCO 2010. ACM, New York (to appear 2010)
7. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: Proceedings of CEC 2010. IEEE, Los Alamitos (to appear 2010)
8. Rudolph, G.: Convergence properties of evolutionary algorithms. Kovac, Hamburg (1997)
9. Witt, C.: Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions. Evolutionary Computation 14, 65–86 (2006)
10. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence 3, 35–68 (2009)
11. Jansen, T., Wegener, I.: On the choice of the mutation probability for the (1+1) EA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 89–98. Springer, Heidelberg (2000)
12. Zarges, C.: Rigorous runtime analysis of inversely fitness proportional mutation rates. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 112–122. Springer, Heidelberg (2008)
13. Zarges, C.: On the utility of the population size for inversely fitness proportional mutation rates. In: Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009), pp. 39–46. ACM, New York (2009)