An Analysis of the Role of Offspring Population Size in EAs

Thomas Jansen
Krasnow Institute
George Mason University
Fairfax, VA 22030
tjansen@gmu.edu

Kenneth De Jong
Krasnow Institute
George Mason University
Fairfax, VA 22030
kdejong@gmu.edu

Abstract

Evolutionary algorithms (EAs) are general stochastic search heuristics often used to solve complex optimization problems. Unfortunately, EA theory is still somewhat weak with respect to providing a deeper understanding of EAs and guidance for the practitioner. In this paper we improve this situation by extending existing theory on the well-known (1+1) EA to cover the $(1+\lambda)$ EA, an EA that maintains an offspring population of size λ. Our goal is to understand how the value of λ affects expected optimization time. We compare the $(1+\lambda)$ EA with the (1+1) EA and prove that on simple unimodal functions no improvements are obtained for $\lambda > 1$. By contrast there are more complex functions for which sensible values of λ can decrease the optimization time from exponential to a polynomial of small degree with overwhelming probability. These results shed light on the role of λ and provide some guidelines for the practitioner.

1 INTRODUCTION

Evolutionary algorithms (EAs) are a broad class of stochastic search heuristics that include genetic algorithms (Goldberg 1989), evolution strategies (Schwefel 1995), and evolutionary programming (Fogel 1995). While there are countless reports of successful applications, EA theory is often concerned with either only isolated aspects of the algorithm or extremely simplified versions. One example are investigations of the so-called (1+1) EA, a very simple EA that uses a parent population of size 1 in each generation to create a single offspring by bit-wise mutation and replaces the parent by its offspring if the fitness if the offspring is not inferior to that of its parent (see, for example, Rudolph (1997), Garnier, Kallel, and Schoenauer (1999), and Droste, Jansen, and Wegener (2002)).

It has been known for some time that a simple (1+1) EA is often at least as efficient as much more sophisticated EAs (Juels and Wattenberg 1995; Mitchell 1995). This motivates the search for situations when the (1+1) EA is outperformed by more sophisticated EAs. For example, if we increase the parent population size μ, i.e., a $(\mu+1)$ EA and introduce multi-parent reproduction, there are problems for which the use of uniform or 1-point crossover can reduce the expected optimization time from exponential to a polynomial of small degree (Jansen and Wegener 2001).

In this paper we focus on the role of the offspring population size λ in $(1+\lambda)$ EAs. In particular, we would like to know when it makes sense to have $\lambda > 1$, and if so, what are reasonable values for λ. Our intuition suggests that, for simple unimodal functions, simple hill-climbing optimization procedures such as a (1+1) EA are efficient and not likely to be outperformed by $(1+\lambda)$ EAs. However, as the complexity of the functions to be optimized increases, there should be benefits for having $\lambda > 1$.

We formally address these issues by extending the analysis of the (1+1) EA as described in Droste, Jansen, and Wegener (2002). In particular, we are able to characterize the slowdown due increasing λ on two well-known unimodal test functions: OneMax and LeadingOnes. In addition, we exhibit a class of functions for which increasing λ in a quite sensible manner reduces the expected optimization time from e^n to n^2, where n is the dimension of the search space. The result of this analysis is an improved understanding of the role of offspring population size and a better sense of how to choose λ for practical applications.

In the next section, we give a formal description of the $(1+\lambda)$ EA and describe the two well-known test functions investigated. In Section 3 we present results on the expected optimization time of the $(1+\lambda)$ EA for these test problems. In Section 4 we present an example function that allows us to see when the use of an offspring population can reduce the optimization time from exponential to a polynomial of small degree. However, modifying this example function we see that also the opposite behavior can be observed, i. e., the use of a quite small population increases the optimization from $O(n^2)$ to exponential. In Section 5, we conclude and describe possible future research.

2 DEFINITIONS

The $(1+\lambda)$ EA to be analyzed is an evolutionary algorithm that maintains a parent population of a size one to produce in each generation λ offspring independently and replaces the parent by a best offspring if the fitness of this offspring is not inferior to the fitness of its parent. The internal representation is a fixed length binary string of length n and offspring variation is the result of bit-wise mutation. For convenience we assume the optimization goal is maximization. More formally, we wish to maximize $f\colon \{0,1\}^n \to \mathbb{R}$ using:

Algorithm 1 ($(1+\lambda)$ EA).

```
1.  Choose $x \in \{0,1\}^n$ uniformly at random.
2.  For $i := 1$ To $\lambda$ Do
        Create $y_i$ by flipping each bit
        in $x$ independently with
        probability $1/n$.
3.  $m := \max\{f(y_1), \dots, f(y_\lambda)\}$.
4.  If $m \geq f(x)$ Then
        Replace $x$ by one $y_i$ chosen
        uniformly at random from
        $\{y_i \mid (1 \leq i \leq \lambda) \wedge (f(y_i) = m)\}$.
5.  Continue at line 2.
```

Clearly, by setting $\lambda := 1$, we get the well-known (1+1) EA. Our goal is to analyze the effects that $\lambda > 1$ has on optimization performance. As usual in theoretical analysis we measure the optimization time in terms of the number of function evaluations needed before x becomes a globally optimal point for the first time. If G is the number of generations, i. e., the number of times lines 2–4 are executed, before this happens, we have $T = 1 + \lambda \cdot G$ for the optimization time T. Note that T is a fairer measure than the number of generations G. However, on a parallel machine a speed-up of the factor λ can be achieved. Since T obviously is a random variable, we are mostly concerned with $\mathrm{E}(T)$, the expected optimization time. Of course, other properties of T can be of interest, too.

3 PERFORMANCE ON SIMPLE UNIMODAL FUNCTIONS

Theoretical analyses of evolutionary algorithms often begin with simply structured example problems. The perhaps best known binary string test problem is OneMax, where the function value is defined to be the number of ones in the bitstring. Another slightly less trivial example is LeadingOnes, where the function value is given by the number of consecutive ones counting from left to right.

One way of analyzing algorithm performance on test problems is via "growth curve analysis" in which closed form expressions are sought that express the amount of time required to solve a problem as a function of the "size" of the problem (Corman, Leiserson, Rivest, and Stein 2001). For example, it is known that the expected optimization time $\mathrm{E}(T)$ of the (1+1) EA on OneMax is $\Theta(n \log n)$, i.e., as the size of the binary search space $\{0,1\}^n$ increases, the increase in $\mathrm{E}(T)$ is well-approximated by $n \log n$, and on LeadingOnes the expected optimization time is $\Theta\left(n^2\right)$ (Droste, Jansen, and Wegener 2002).

Both of these functions are examples of unimodal functions that can be optimized efficiently by simple hill-climbing algorithms such as a (1+1) EA. As a consequence, intuitively, one shouldn't expect a $(1+\lambda)$ EA to outperform a $(1+1)$ EA on such functions. Rather, one might expect a decrease in performance as we increase λ. In this section we show this more formally by extending the (1+1) EA growth curve analysis to $(1+\lambda)$ EAs. Interestingly, doing this for LeadingOnes is far simpler than for OneMax.

3.1 PERFORMANCE ON LEADINGONES

Theorem 1.1. *For the $(1+\lambda)$ EA on the function* LeadingOnes: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Theta\left(n^2 + n \cdot \lambda\right)$ *holds if λ is bounded above by a polynomial in n.*

Proof. We begin with the upper bound and distinguish two different cases with respect to the number of offspring λ. First, we assume that $\lambda \leq en$ holds. Note, that we want to prove that the expected optimization time is $\Theta\left(n^2\right)$. We begin with the upper bound. Obviously, it is sufficient if the current function value is increased by at least 1 at least n times in order to reach the unique global optimum 1^n. During the optimization the probability to create an offspring with larger function value is bounded below by $(1/n) \cdot (1-1/n)^{n-1} \geq 1/(en)$, since it is always sufficient to mutate exactly the left-most bit with value 0. The probability that in one generation no offspring has larger function value than x is therefore bounded above by $(1-1/(en))^{\lambda}$. Thus, with probability at least $1-(1-1/(en))^{\lambda} \geq 1-e^{-\lambda/(en)}$ the current string x is replaced by one of its offspring with larger function value. Since we have $\lambda \leq e \cdot n$, obviously $\lambda/(en) \leq 1$ holds. For $t \in \mathbb{R}$ with $0 \leq t \leq 1$ we have $1-e^{-t} \geq t/2$. Thus, the probability that in one generation at least one offspring improves upon its parent x is bounded below by $\lambda/(2en)$ for $\lambda \leq en$. Therefore, the expected number of generations for an improvement of the function value is bounded above by $2en/\lambda$. We see, that the expected number of generations the $(1+\lambda)$ EA needs to optimize LeadingOnes is bounded above by $2en^2/\lambda$. So, the expected optimization time is at most $\lambda \cdot 2en^2/\lambda = O\left(n^2\right)$ as claimed.

Now, assume that $\lambda > en$ holds. As we noticed above, the probability that in one generation the $(1+\lambda)$ EA creates an offspring with larger function value than its parent is bounded below by $1-e^{-\lambda/en}$. Since we assume $\lambda > en$, this is bounded below by $1-e^{-1}$. Thus, the expected number of generations until we have n such generations is bounded above by $O(n)$ implying $O(\lambda \cdot n)$ as upper bound on $\mathrm{E}(T)$.

In order to derive a lower bound, we remember that λ is bounded above by some polynomial. So, there exists a constant k, such that $\lambda \leq n^k$ holds. We see that the probability of producing an offspring where $k+2$ pre-specified bits are mutated is bounded above by $1/n^{k+2}$. Thus, the probability that such an individual is not created among $n \cdot \lambda$ offspring is bounded below by $1-\left(1/n^{k+2}\right) \cdot (n \cdot \lambda) > 1-1/n$. Now, we consider only the first $n/(4k+8) = \Theta(n)$ generations. There at most $n^{k+1}/(4k+8)$ offspring are produced. Thus, with probability at least $1-1/n$ no such individual is created. Now, we work under the assumption that this is the case in the observed run. With probability $1/2$, at least $n/2$ bits in the initial string are 0. It is known (Droste, Jansen, and Wegener 2002), that the bits that are to the right of the leftmost bit with value 0 are random and independent during the run. Thus, with probability at least $(1/2)-1/n$, after $n/12$ generations the global optimum is not reached. This implies $\Omega(\lambda n)$ as a lower bound on $\mathrm{E}(T)$. □

This theorem confirms our intuition that a $(1+\lambda)$ EA will not outperform a $(1+1)$ EA on LeadingOnes since no value of $\lambda > 1$ will reduce the expected optimization time below $\Theta(n^2)$. However, the more interesting implication of this theorem is that, from a growth curve analysis point of view, there is no significant slowdown if λ is anywhere in the range $1 \leq \lambda \leq n$, since $\Theta\left(n^2+n \cdot \lambda\right) = \Theta\left(n^2\right)$ for $\lambda = O(n)$.

This is due to the fact that, in each step of the LeadingOnes function, the probability of creating an offspring that is better than its parent is $\Theta(1/n)$ (i.e., in ES terminology, the mutation operator has an almost constant success probability of $\Theta(1/n)$). Thus, producing up to n offspring in each generation before checking for improvements results in no substantial waste of time.

3.2 PERFORMANCE ON ONEMAX

For OneMax things are less obvious. Although it is fairly straightforward to show that a $(1+\lambda)$ EA cannot do better than a $(1+1)$ EA, it is considerably more difficult to pin down precisely when increasing λ begins to significantly affect performance. This is due to the fact that the success probability of mutation for OneMax varies over time rather than remaining almost constant as in LeadingOnes. As we approach the global optimum of 1^n, the probability of creating an offspring that is better than its parents approaches $\Theta(1/n)$. Hence, as we saw with LeadingOnes, any $\lambda \leq n$ is acceptable.

However, at the beginning, starting with a randomly generated binary string, the probability of success is with overwhelming probability larger than $1/c$ for a constant c, much larger than $1/n$. Hence, unless λ is a constant independent of n, there will be at least an initial slowdown. Consequently, we see that there is no simple intuitive answer as to when an increase in λ degrades performance. But it is possible to bracket λ with useful upper and lower bounds. We begin with a theorem that establishes an upper bound on $\mathrm{E}(T)$ and indirectly sets a lower bound for λ:

Theorem 1.2. *For the $(1+\lambda)$ EA on the function* OneMax: $\{0,1\}^n \rightarrow \mathbb{R}$, $E(T) = O(n \log n + n\lambda)$ *holds.*

Proof. Assume that the current string x has Hamming distance d from the global optimum, i.e., $d = n - \text{OneMax}(x)$. The probability to create an offspring y with $\text{OneMax}(y) > \text{OneMax}(x)$ is bounded below by $d/n(1-1/n)^{n-1} \geq d/(en)$. Thus, the probability not to increase the function value of x in one generation is bounded above by $(1-d/(en))^{\lambda} \leq e^{-d \cdot \lambda/(en)}$. So, the probability to increase the function value of x in one generation is bounded below by $1-e^{-d \cdot \lambda/(en)} \geq 1-1/\left(1+d \cdot \lambda/(en)\right) = (d \cdot \lambda)/(en+d \cdot \lambda)$. Thus, $\mathrm{E}(T)$ is bounded above by

$$\lambda \cdot \sum_{d=1}^{n} \frac{en+d\lambda}{d\lambda} = \lambda\left(n+\frac{en}{\lambda}\sum_{d=1}^{n}\frac{1}{d}\right) = O(n\lambda+n\log n).$$

□

In particular, if λ is bounded above by $\log n$, i.e., $\lambda = O(\log n)$, then the expected optimization time is bounded above by $n \log n$, which is no worse than a $(1+1)$ EA on OneMax. Similarly, a useful lower bound on $\mathrm{E}(T)$ can be obtained that can be used to establish an upper bound on λ:

Theorem 1.3. *For the* $(1+\lambda)$ *EA on the function* OneMax: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Omega(\lambda \cdot n/\log n)$ *holds if* λ *is bounded above by a polynomial in* n.

Proof. The probability to create an offspring y by mutating x with $\text{OneMax}(y) \geq \text{OneMax}(x) + d$ is bounded above by

$$\binom{n}{d} \cdot \frac{1}{n^d} \leq \frac{1}{d!} < \left(\frac{e}{d}\right)^d$$

for all $x \in \{0,1\}^n$ and all $d \in \{1, 2, \ldots, n - \text{OneMax}(x)\}$. Therefore, the probability to create one such y within $n \cdot \lambda$ independent tries is bounded above by $n \cdot \lambda \cdot (e/d)^d$. We conclude that the probability to create one such y within $n \cdot \lambda$ independent tries with $d \in \{\log(n), \ldots, n - \text{OneMax}(x)\}$ is bounded above by $n \cdot \lambda \cdot (e/\log n)^{\log n} = e^{-\Omega(\log(n) \log\log n)}$. Thus, the probability that within n generations the Hamming distance to the optimum is not decreased by at least $\log(n)$ in one single generation is $1 - 2^{-\Omega(\log(n)\log\log n)}$. Since after random initialization the Hamming distance to the global optimum is at least $n/2$ with probability $1/2$, this implies $\Omega((n/\log n) \cdot \lambda)$ as a lower bound on $\mathrm{E}(T)$ as claimed. □

Hence, if we allow λ to increase faster than $\log^2 n$, we begin to see a significant slowdown relative to the $n \log n$ performance of a (1+1) EA on OneMax. Combining this with the previous theorem we see that keeping $\lambda < \log n$ means we'll do no worse than a $(1+1)$ EA. This still leaves open the possibility that there are values of $\lambda \leq \log^2 n$ that result in performance improvements over a (1+1) EA. However, as our intuition suggests, this is not the case. We show this formally by proving that, whenever λ grows more slowly than $\sqrt{n}/\log n$ (an upper bound much larger than $\log^2 n$), $\mathrm{E}(T)$ is bounded below by $\Omega(n \log n)$:

Theorem 1.4. *For the* $(1+\lambda)$ *EA on the function* OneMax: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Omega(n \log n)$ *holds if* $\lambda = o(\sqrt{n}/\log n)$.

Proof. It is easy to see that at some point of time the Hamming distance between the current string x and the global optimum is within $\{\lceil\sqrt{n}/2\rceil, \ldots, \lceil\sqrt{n}\rceil\}$ with probability very close to 1. We consider a run of the $(1+\lambda)$ EA after this point of time. Then, the probability to create an offspring y with $\text{OneMax}(y) \geq \text{OneMax}(x) + 3$ is bounded above by $\binom{\sqrt{n}}{3} \cdot 1/n^3 < 1/n^{3/2}$. Thus, the probability to create at least one such individual within $\lambda \cdot n \log n$ independent tries is bounded above by $1 - \left(1 - 1/n^{3/2}\right)^{\lambda \cdot n \log n} = O(\lambda \cdot \log n/\sqrt{n}) = o(1)$. Now, we continue under the assumption that no such individual is created. Then, the Hamming distance can only be decreased by 1 or 2 in each generation. Given that the current Hamming distance between x and the global optimum is d, the probability to create an offspring that decreases this Hamming distance is bounded above by $2d/n$. Thus, the probability to do so in one generation is bounded above by $1 - (1 - 2d/n)^\lambda < 4\lambda \cdot d/n$. Thus, the expected number of generations is bounded below by $\sum_{d=1}^{\sqrt{n}} n/(4\lambda \cdot d) = \Omega((n/\lambda) \cdot \log n)$. This implies $\mathrm{E}(T) = \Omega(n \log n)$. □

The previous 3 theorems collectively tell us that we can increase λ up to $\log n$ without a significant degradation in performance, but not beyond $\log^2 n$, leaving the interval between $\log n$ and $\log^2 n$ unresolved. The final two theorems of this section narrow that gap considerably by increasing the lower bound slightly and decreasing the upper bound to within a constant factor of the lower bound. We begin with the lower bound:

Theorem 1.5. *For the* $(1+\lambda)$ *EA on the function* OneMax: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Theta(n \log n)$ *holds if* $\lambda \leq (\ln n) \cdot (\ln\ln n)/(2 \ln\ln\ln n)$.

Proof. The lower bound follows from Theorem 1.4. For $\lambda = O(\log n)$ the upper bound follows from Theorem 1.2. So, we assume $\lambda \geq \ln n$ from now. In particular, we define $\gamma := \lambda/\ln n$ and have $\gamma \geq 1$.

We divide a run of the $(1+\lambda)$ EA into two disjoint phases and count the number of function evaluations separately for each phase. Let T_1 denote this number for the first phase and let T_2 denote this number for the second phase. The first phase starts after random initialization and continues as long as $\text{OneMax}(x) \leq n - n/\ln\ln n$ holds for the current string x. The second phase starts immediately after the end of the first phase and ends when the global optimum is reached. Obviously, for the expected optimization time $\mathrm{E}(T)$ we have $\mathrm{E}(T) = \mathrm{E}(T_1) + \mathrm{E}(T_2)$ with these definitions.

In order to get an upper bound on $\mathrm{E}(T_2)$, we reconsider the proof of Theorem 1.2. We see that $\mathrm{E}(T_2)$ is bounded above by $\lambda \cdot \sum_{d=1}^{n/\ln\ln n}(en + d \cdot \lambda)/(d \cdot \lambda) = \lambda \cdot ((n/ln \ln n) + (en/\lambda) \cdot \sum_{d=1} n/\ln\ln n 1/d) = O(\lambda \cdot (n/\ln\ln n) + n \log n) = O(n \log n)$. Here we need $\lambda = O(\log(n)/\log\log n)$.

Now we consider the first phase. Since we have $\textsc{OneMax}(x) \leq n - n/\ln\ln n$, the probability to create an offspring y by mutation of x with $\textsc{OneMax}(y) \geq \textsc{OneMax}(x) + \gamma$ is bounded below by

$$\begin{aligned} & \binom{n/\ln\ln n}{\gamma} \cdot \left(\frac{1}{n}\right)^{\gamma} \cdot \left(1 - \frac{1}{n}\right)^{n-\gamma} \\ \geq & \left(\frac{n}{\gamma \cdot n \cdot \ln\ln n}\right)^{\gamma} \cdot \frac{1}{e} = e^{-(1+\gamma\ln\gamma+\gamma\ln\ln\ln n)}. \end{aligned}$$

We conclude that the probability to create at least one such individual y in one generation is bounded below by $1 - \left(1 - e^{-(1+\gamma\ln\gamma+\gamma\ln\ln\ln n)}\right)^{\lambda} \geq \lambda/((e\ln n)+\lambda) \geq 1/(e+1)$ since we have $\gamma \leq (\ln\ln n)/(2\ln\ln\ln n)$ by assumption. Obviously, after less than n/γ such generations the first phase ends. This implies $O(\lambda \cdot n/\gamma) = O(n\log n)$ as upper bound on $\mathrm{E}(T_1)$. □

We cannot prove that $(\ln n)(\ln\ln n)/(2\ln\ln\ln n)$ is a sharp upper bound in the sense that any value for λ larger than this implies that the expected optimization is worse than $n\log n$. However, we can show that it is true when λ grows asymptotically faster than that.

Theorem 1.6. *For the* $(1+\lambda)$ *EA on the function* $\textsc{OneMax}\colon \{0,1\}^n \to \mathbb{R}$, $E(T) = \omega(n\log n)$ *holds if* $\lambda = \omega\left((\ln n)(\ln\ln n)/\ln\ln\ln n\right)$.

Proof. We use a different proof strategy in order to derive this tighter lower bound. We derive an upper bound on the expected decreasement in the Hamming distance in one generation. Then we use this upper bound in order to prove that it is not likely that the Hamming distance is decreased by a certain amount in a pre-defined number of generations.

In analogy to the proof of Theorem 1.4 it is easy to see that at some point of time the Hamming distance between the current string x and the global optimum is within $\{\lceil n/(2e)\rceil, \ldots, \lceil n/e\rceil\}$ with probability very close to 1. From this point of time on the probability to create an offspring y with $\textsc{OneMax}(y) \geq \textsc{OneMax}(x) + d$ is bounded above by $\binom{n/e}{d} \cdot 1/n^d < 1/d^d$.

Consider g generations of the $(1+\lambda)$ EA. Let x be the current string before the first generation and let x' be the current string after the g-th generation. Let $D_{\lambda,g,x}$ denote the advance in this g generations by means of Hamming distance, i. e. $D_{\lambda,g,x} := \textsc{OneMax}(x') - \textsc{OneMax}(x)$. Obviously, $D_{\lambda,g,x}$ depends on x and we have $\mathrm{Prob}\left(D_{\lambda,g,x} \geq d\right) \geq \mathrm{Prob}\left(D_{\lambda,g,y} \geq d\right)$ for all $d \in \{0,1,\ldots,n\}$ and all $x, y \in \{0,1\}^n$ with $\textsc{OneMax}(x) \leq \textsc{OneMax}(y)$. Since the function value of the current string of the $(1+\lambda)$ EA can never decrease, $\mathrm{E}\left(D_{\lambda,g,x}\right) \leq g \cdot \mathrm{E}\left(D_{\lambda,1,x}\right)$ holds for all λ, g and x. So, we concentrate on $\mathrm{E}\left(D_{\lambda,1,x}\right)$ now.

Obviously, $D_{\lambda,1,x}$ is a random variable that depends on λ and the current string x at the beginning of the generation. However, it is clear that $\mathrm{Prob}\left(D_{\lambda,1,x} \geq d\right) < \lambda/d^d$ holds for all x with $\textsc{OneMax}(x) \leq n - \lceil n/e\rceil$. From now on, we always assume that $\textsc{OneMax}(x)$ is bounded above in this way.

We are interested in $\mathrm{E}\left(D_{\lambda,1,x}\right)$. Since $D_{\lambda,1,x} \in \{0, 1, \ldots, n\}$, we have $\mathrm{E}\left(D_{\lambda,1,x}\right) = \sum_{i=1}^{n} \mathrm{Prob}\left(D_{\lambda,1,x} \geq d\right)$. For $d < (3\ln\lambda)/\ln\ln\lambda$ we use the trivial estimation $\mathrm{Prob}\left(D_{\lambda,1,x} \geq d\right) \leq 1$. For d with $(3\ln\lambda)/\ln\ln\lambda \leq d < (\lambda\ln\lambda)/\ln\ln\lambda$ we have

$$\frac{\lambda}{d^d} \leq \frac{e^{\ln\lambda}}{e^{((3\ln\lambda)/(\ln\ln\lambda))\cdot((\ln\ln\lambda)-\ln\ln\ln\lambda))}} < \frac{1}{\lambda}$$

and use the estimation $\mathrm{Prob}\left(D_{\lambda,1,x} \geq d\right) \leq 1/\lambda$. Finally, for $d \geq (\lambda\ln\lambda)/\ln\ln\lambda$ we have $\lambda/d^d \leq e^{\lambda}/e^{\lambda\ln\lambda} < e^{-\lambda}$ and use the estimation $\mathrm{Prob}\left(D_{\lambda,1,x} \geq d\right) < e^{-\lambda} < 1/n$. These three estimations yield $\mathrm{E}\left(D_{\lambda,1,x}\right) < 4\ln\lambda/\ln\ln\lambda$ for the expected advance in one generation.

Let G denote the number of generations the $(1+\lambda)$ EA needs for optimization of $\textsc{OneMax}$. Of course, $\mathrm{E}(G) \geq t \cdot \mathrm{Prob}\left(G \geq t\right)$ holds for all values of t. As we argued above, with probability at least $1/2$ at some point of time we have that some x with $\textsc{OneMax}(x) \in \{n - \lceil n/e\rceil, \ldots, n - \lceil n/(2e)\rceil\}$ as current string x. Thus, $\mathrm{Prob}\left(G \geq t\right) \geq \mathrm{Prob}\left(D_{\lambda,t,x} < n/e\right)$ holds, if x is some string with at least n/e zero bits. This yields $\mathrm{E}(G) \geq (t/2) \cdot \mathrm{Prob}\left(D_{\lambda,t,x} < n/e\right) = (t/2) \cdot (1 - \mathrm{Prob}\left(D_{\lambda,t,x} \geq n/e\right))$. By Markov inequation we have $\mathrm{E}(G) \geq (t/2) \cdot \left(1 - \mathrm{E}\left(D_{\lambda,t,x}\right)/(n/e)\right) \geq (t/2) \cdot \left(1 - e \cdot t \cdot \mathrm{E}\left(D_{\lambda,1,x}\right)/n\right)$. Together with our estimation for $\mathrm{E}\left(D_{\lambda,1,x}\right)$ we have $\mathrm{E}(G) \geq (t/2) \cdot (1 - 4e \cdot t \cdot \ln\lambda/(n \cdot \ln\ln\lambda))$. We set $t := (n\ln\ln\lambda)/(8e\ln\lambda)$ and get $\mathrm{E}(G) \geq n\ln\ln\lambda/(32e\ln\lambda)$ which implies $\mathrm{E}(T) = \Omega\left(n\lambda\ln\ln\lambda/\ln\lambda\right)$ for the expected optimization time $\mathrm{E}(T)$, since $T = G \cdot \lambda$ holds. It is easy to see, that this implies $\mathrm{E}(T) = \omega(n\log n)$ for $\lambda = \omega((\ln n)(\ln\ln n)/\ln\ln\ln n)$ as claimed. □

3.3 SUMMARY

The theorems in this section provide a clear picture of the performance of a $(1+\lambda)$ EA on $\textsc{OneMax}$. It never outperforms a $(1+1)$ EA. It's performance is about the same if λ doesn't get much bigger than $(\log n)(\log\log n)/2\log\log\log n$, and performance degrades significantly after that.

4 $(1+\lambda)$ EA Performance On More Complex Functions

The previous section showed that for simple functions like OneMax and LeadingOnes increasing λ does not improve $\mathrm{E}(T)$ over $(1+1)$ EA performance. In this section we focus on cases where increasing λ does improve performance. Intuitively, a necessary condition for it to be useful to invest more effort in the random sampling in the neighborhood of the current population is that the fitness landscape is to some degree misleading in the sense that a $(1+1)$ EA is more likely to get trapped on a local peak.

In this section we show that the performance improvement due to increased values of λ can be tremendous. We illustrate this with an artificial function for which we can prove that increasing λ from 1 to n reduces $\mathrm{E}(T)$ from e^n to n^2 with a probability that converges to 1 extremely fast.

The intuition for this function is quite simple as illustrated in Fig. 1. We want it to consist of a narrow path that leads to the optimum. However, while following that path, the algorithm is confronted multiple branch points, each with the property that from it there are a variety of paths leading uphill, but only the steepest one leads to the global optimum. Hence, as we increase λ we increase the likelihood of picking the correct branch point path.

The formal definition of this function is more complicated than the intuition. To simplify it somewhat, we divide the definition into two parts. First, we define a function $f\colon \{0,1\}^n \to \mathbb{R}$ that realizes the main idea but assumes that the initial string is 0^n.

Definition 1.1. *For $n \in \mathbb{N}$ we define $k := \lfloor\sqrt{n}\rfloor$. We use $|x| = \text{OneMax}(x)$ and define the function $f\colon \{0,1\}^n \to \mathbb{R}$ for all $x \in \{0,1\}^n$ by*

$$f(x) := \begin{cases} (2i+3)n + |x| & \text{if } x = y0^{n-ik}1^{(i-1)k} \text{ with } 1 \le i \le k/2 \text{ and } y \in \{0,1\}^k \\ (2i+4)n + |x| & \text{if } x = 0^{n-j}1^j \text{ with } i := \lceil j/k \rceil,\ 1 \le i \le k/2, \text{ and } i\cdot k \ne j \\ 0 & \text{otherwise} \end{cases}$$

The core function f contains one main path 0^i1^{n-i}. There are about $\sqrt{n}/2$ points on this path that are of special interest. At these points it is not only beneficial to add another one in the right side. The function value is also increased by flipping any of the left most $\lfloor\sqrt{n}\rfloor$ bits. Thus, at these points there are a variety of uphill paths to chose among. One path of the form 0^i1^{n-i} leads to the global optimum, while paths of the form $y0^{i'}1^{n-i'-\lfloor\sqrt{n}\rfloor}$ with $y \in \{0,1\}^{\lfloor\sqrt{n}\rfloor}$ all lead to local optima. Therefore we call these special points on the path branching points.

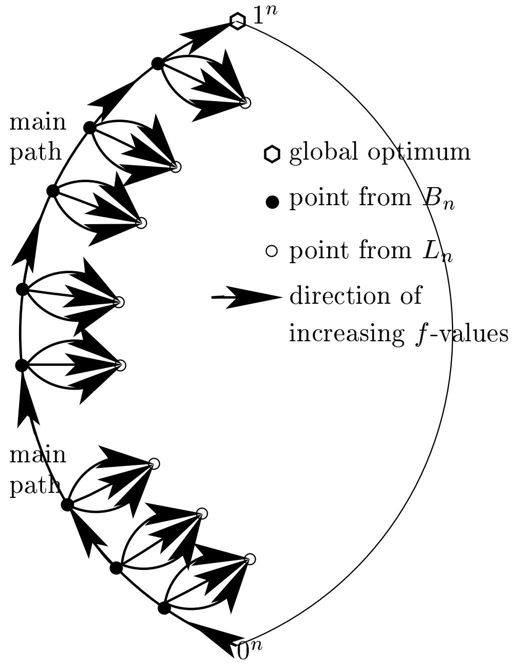

Figure 1: Core function f.

Definition 1.2. *For $n \in \mathbb{N}$ we define $k := \lfloor\sqrt{n}\rfloor$ and call a point $x \in \{0,1\}^n$ a* branching point *of dimension n iff $x = 0^{n-(i-1)k}1^{(i-1)k}$ holds for some $i \in \{1,2,\ldots,\lfloor k/2\rfloor - 1\}$. Let B_n denote the set of all branching points of dimension n.*

At the branching points a $(1+\lambda)$ EA may proceed toward a local optima or the global one. It is important to know what the probabilities are for the two different possibilities:

Lemma 1.1. *Let $n \in \mathbb{N}$, $k := \lfloor\sqrt{n}\rfloor$, B_n the set of branching points of dimension n as defined in Definition 1.2, $x \in B_n$, and $\lambda\colon \mathbb{N} \to \mathbb{N}$ be a function that has a polynomial upper bound. Consider a $(1+\lambda)$ EA with current string x optimizing the function $f\colon \{0,1\}^n \to \mathbb{R}$ as defined in Definition 1.1. Let $y = y_1y_2\cdots y_n$ be the first point with $f(y) > f(x)$ reached by the $(1+\lambda)$ EA. The probability that $\text{OneMax}(y_1y_2\cdots y_k) > 0$ holds is $1 - e^{-\Theta(\lambda/\sqrt{n})}$.*

Proof. We consider the $(1+\lambda)$ EA on f with current string $x = x_1x_2\cdots x_n \in B_n$. Let $x' = x'_1x'_2\cdots x'_n$ be one individual created by mutation of x. We consider several events concerning x'.

The points in B_n are ordered according to the function value. The probability that x' has a function value that is equal or greater to the next branching point is obviously $2^{-\Omega(\sqrt{n})}$. We neglect such an event and take care of this in our upper and lower bounds by adding or subtracting $2^{-\Omega(\sqrt{n})}$.

We denote the event that $\text{OneMax}(x'_1 \cdots x'_k) > 0$ holds by A. We have

$$\text{Prob}(A) \geq \binom{k}{1} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} > \frac{k}{en}$$

as a lower bound and

$$\begin{aligned} &\text{Prob}(A) \\ \leq\ & \left(\sum_{i=1}^{k} \binom{k}{i} \left(\frac{1}{n}\right)^i \left(1 - \frac{1}{n}\right)^{n-i}\right) + 2^{-\Omega(\sqrt{n})} \\ <\ & \left(\sum_{i=1}^{k} \left(\frac{k}{in}\right)^i\right) + 2^{-\Omega(\sqrt{n})} < \frac{k}{n-k} \end{aligned}$$

for an upper bound. We denote the event that $\text{OneMax}(x'_1 \cdots x'_k) = 0$ holds by B and have

$$\text{Prob}(B) \geq \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} > \frac{1}{en}$$

as a lower bound and

$$\text{Prob}(B) \leq \left(\sum_{i=1}^{k} \left(\frac{1}{n}\right)^i \cdot \left(1 - \frac{1}{n}\right)^{n-i}\right) + 2^{-\Omega(\sqrt{n})} < \frac{1}{n}$$

for an upper bound.

We are interested in the probability of A given that A or B occur. Since $A \cap B = \emptyset$, we have $\text{Prob}(A|A \cup B) = \text{Prob}(A) / (\text{Prob}(A) + \text{Prob}(B))$. Obviously, for each $a, b, c \in \mathbb{R}^+$ we have that $a/(a+c) \geq b/(b+c)$ holds iff $a \geq b$. This implies that

$$\frac{k/(en)}{k/(en) + 1/n} < \text{Prob}(A|A \cup B) < \frac{k/(n-k)}{k/(n-k) + 1/(en)}$$

follows from our upper and lower bounds on $\text{Prob}(A)$ and $\text{Prob}(B)$ and we have $1 - e/\sqrt{n} < \text{Prob}(A|A \cup B) < 1 - 1/(e\sqrt{n})$. Let C denote the event that we are really interested in, the event that $\text{OneMax}(y_1 \cdots y_k) > 0$ holds for the first point $y = y_1 \cdots y_n$ with $f(y) > f(x)$ reached by the $(1+\lambda)$. Since in each generation the λ offspring are independently generated, we have $\text{Prob}(C) = 1 - (1 - \text{Prob}(A|A \cup B))^{\lambda}$ and can conclude that

$$1 - \left(\frac{e}{\sqrt{n}}\right)^{\lambda} < \text{Prob}(C) < 1 - \left(\frac{1}{e\sqrt{n}}\right)^{\lambda}$$

holds. Since $(1 - 1/x)^x < e^{-1} < (1 - 1/x)^{x-1}$ holds for all $x \in \mathbb{N}$, this completes the proof. □

Assume that the $(1+\lambda)$ EA happens to continue in the direction of a $y0^i1^{n-i-\lfloor\sqrt{n}\rfloor}$ with $y \in \{0,1\}^{\lfloor\sqrt{n}\rfloor}$, $y \neq 0^{\lfloor\sqrt{n}\rfloor}$. Then it is quite likely to reach "the last point in this direction", i. e., $y0^i1^{n-i-\lfloor\sqrt{n}\rfloor}$ with $y = 1^{\lfloor\sqrt{n}\rfloor}$. It will turn out be convenient to define a set for those points similar to B_n.

Definition 1.3. *For $n \in \mathbb{N}$ we define $k := \lfloor\sqrt{n}\rfloor$ and*

$$L_n := \left\{1^k 0^{n-ik} 1^{(i-1)k} \mid i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}\right\}.$$

When considering the (1+1) EA on the core function f, it is essential that the algorithm is started with 0^n as initial string. Since the (1+1) EA choose the initial string uniformly at random this will not be the case with probability $1 - 2^{-n}$. Therefore, we now relax the assumption that the initial string is 0^n by extending f to g in a way that any starting point leads to the main path defined by f.

Definition 1.4. *For $n \in \mathbb{N}$ we define $m' := \lfloor n/2 \rfloor$, $m'' := \lceil n/2 \rceil$, and $g\colon \{0,1\}^n \to \mathbb{R}$ by*

$$g(x) := \begin{cases} n - \text{OneMax}(x'') & \text{if } x' \neq 0^{m'} \wedge x'' \neq 0^{m''} \\ 2n - \text{OneMax}(x') & \text{if } x' \neq 0^{m'} \wedge x'' = 0^{m''} \\ f(x'') & \text{if } x' = 0^{m'} \end{cases}$$

for all $x = x_1x_2 \cdots x_n \in \{0,1\}^n$ with $x' := x_1x_2 \cdots x_{m'} \in \{0,1\}^{m'}$ and $x'' := x_{m'+1}x_{m'+2} \cdots x_n \in \{0,1\}^{m''}$.

We double the length of the each bitstring and define the core function f only on the right half of the strings. The first half is used to lead a search algorithm towards the beginning of the main path of f. The construction will have the desired effect for all search heuristics that are efficient on OneMax.

Theorem 1.7. *The probability that the (1+1) EA optimizes the function $g\colon \{0,1\}^n \to \mathbb{R}$ within $n^{O(1)}$ steps is bounded above by $2^{-\Omega(\sqrt{n}\log n)}$.*

Proof. Our proof strategy is the following. First, we consider the expected optimization time of the (1+1) EA on the function $g\colon \{0,1\}^n \to \mathbb{R}$ under the condition that at some point of time the current string $x = x'x''$, with x' and x'' defined as in Definition 1.4, is of the form $x' = 0^{m'}$, $x'' \in L_{m''}$, with $m' = |x'|$, $m'' = |x''|$, $k = \left\lfloor\sqrt{m''}\right\rfloor$, and $i \in \{1, \ldots, \lfloor k/2 \rfloor\}$. Then, we prove that the probability that such a string becomes current string at some point of time is $\Omega(1)$.

First, assume that such a string x is current string of the (1+1) EA. Let A be the set of all such strings. Obviously, this string x is different from the unique global optimum. Moreover, due to the definition of g, all points with larger function value have Hamming distance at least k and there are less than n such points. Thus, the probability to reach such a point in one generation is bounded above by $n \cdot (1/n)^k \leq (1/n)^{\sqrt{n/2}-2}$. The probability that such an event occurs at least once in n^k generations is bounded above by $n^{-\sqrt{n/2}+k+1} = 2^{-\Omega\left(\sqrt{n}\log n\right)}$.

The initial current string $x = x'x''$ after random initialization is some string with $x' \neq 0^{m'}$ with probability $1-2^{-m'}$. Then, the function value is either given by $n - \text{ONEMAX}(x'')$ or $2n - \text{ONEMAX}(x')$. With probability $1-2^{-\Omega(n)}$ the all zero string 0^n is reached within $O\left(n^2 \log n\right)$ steps given that no other string y with $g(y) > g\left(0^n\right)$ is reached before. There are less than $2^k \cdot n$ such strings, thus, the probability to reach such a string within $O\left(n^2 \log n\right)$ steps is bounded above by

$$O\left(n^2 \log n \cdot \frac{2^k \cdot n}{2^n}\right) = 2^{-n+O\left(\sqrt{n}\right)}.$$

At each branching point, the (1+1) EA comes to a string $x = x'x''$ as new current string with some bit set to 1 within the first k bits of x'' with probability at least $1 - 1/(e\sqrt{n})$. This follows from the proof of Lemma 1.1. The probability to proceed with such strings instead of returning to a string with k zero at these bit positions increases. In fact, it is easy to see that with probability $1 - O\left(1/\sqrt{n}\right)$ the $(1+\lambda)$ EA reaches a point in A before reaching some point with k zeros at these positions. Since we have $\lfloor k/2 \rfloor - 1 > \sqrt{n}/5$ branching points, we conclude that the probability that the (1+1) EA does not reach some point in A is bounded above by $2^{-\Omega\left(\sqrt{n}\log n\right)}$ under the described circumstances. Combining all estimations completes the proof. □

Theorem 1.8. *For all constants $\varepsilon > 0$, the probability that the $(1+\lambda)$ EA with $\lambda = n \cdot \lambda'$, $\lambda' \in \mathbb{N}$ optimizes the function $g\colon \{0,1\}^n \to \mathbb{R}$ within $O\left(n^2 \cdot \lambda'\right)$ steps is bounded below by $1 - \varepsilon$.*

Proof. First, assume that the initial string $x = x'x''$ is some string with $x' \neq 0^{m'}$. Given that the all zero string is the first string $y = y'y''$ becoming current string with $y' = 0^{m'}$, we can conclude from Theorem 1.2 that the expected number of steps the $(1+\lambda)$ EA needs to reach the all zero string is $O\left(n^2 \cdot \lambda'\right)$. Similar to the proof of Theorem 1.7 we can conclude that the all zero string is reached within $cn^2 \cdot \lambda$ steps with probability at least $1 - \varepsilon$, where c and ε are positive constants. Note, that by enlarging c the failure probability ε can be made arbitrarily small. Given that for all following current strings $x = x'x''$ we have that x'' does not contain a bit different from 0 within the first k bits, it follows from Theorem 1.1, that the expected number of steps until the global optimum is reached is bounded above by $O\left(n^2 \cdot \lambda'\right)$. So, under the assumption that no such string is reached, we have similarly to the reasoning above that the global optimum is reached within $O\left(n^2 \cdot \lambda'\right)$ steps with probability at least $1 - \varepsilon$, where ε is an arbitrarily small positive constant. We know from Lemma 1.1 that the probability to reach some x where this assumption is not met is bounded above by $e^{-\Omega\left(\sqrt{n}\right)}$ at each branching point. Since there are less than $\sqrt{n}$ branching points, we have that with probability at least $1 - \sqrt{n} \cdot e^{-\Omega\left(\sqrt{n}\right)} = 1 - e^{-\Omega\left(\sqrt{n}\right)}$ no such point becomes current point x at any branching point. Combining these estimations completes the proof. □

At the same time, as has been seen in many other contexts, there is no "free lunch" here in the sense that the impressive speedup on g achieved by increasing λ is not true for all functions with local optima. For example, it is quite easy to modify the function of the previous section so that a (1+1) EA is efficient and a $(1+\lambda)$ EA with $\lambda \geq n$ fails with high probability. To see how, consider f and all strings $x = x'x''$ where x'' is a branching point. Let x''' be a string of length $|x''|$ with $x''' \in L_{|x'''|}$. that has k bits with the value 1 at the beginning and is equal to x'' on the rest of the bits. The proofs of Theorem 1.7 and Theorem 1.8 rely on the fact the the (1+1) EA reaches some string $x'x'''$ with high probability whereas the $(1+\lambda)$ EA does not. Defining a function f' that has all such point $x'x'''$ as global optimum and is equal to f on all other points is obviously optimized within $O\left(n^2\right)$ steps by the (1+1) EA with probability converging to 1, whereas the $(1+\lambda)$ fails to optimize f' within a polynomial number of steps with probability converging to 1, given that $\lambda = \Omega(n)$ holds.

5 CONCLUSIONS

We have used growth curve analysis techniques to better understand the role of offspring population size in $(1+\lambda)$ EAs. As we have seen, increasing λ on simple functions like ONEMAX and LEADINGONES does not lead to an improvement in the expected optimization time. However, increases in λ do not significantly degrade performance as long as $\lambda < \log n$ where n is the dimensionality of the search space. By contrast, for more complex functions with local optima, increasing λ can result in improvements in performance as illustrated in the previous section.

However, a note of caution needs to be raised. The growth curve analysis techniques used here focus on limiting behavior as n increases and view growth curves that differ only by a constant as equivalent. Hence, it may be the case that specific values of n and λ produce "contradictory" results to the conclusions in the previous paragraph.

In spite of this cautionary note we believe that these insights into the role of offspring population size provide some guidelines for the EA practitioner. Since practitioners apply EAs to functions about which they do not have detailed *a priori* knowledge, they can "hedge their bets" by choosing λ to be of order $\log n$. In doing so, they don't lose too much if the function turns out to be easy and may improve their performance on functions with local optima. An interesting observation here is that for most practical problems, $\log n$ is not likely to be too far away from the value of $\lambda = 7$ recommended by the ES community but derived in other contexts (Schwefel 1995, p. 148). On the other hand, if the function appears to be quite difficult in the sense that each run results in convergence to a different local optima, increasing λ to a value of order n is a plausible thing to try.

This work represents a modest step along to road to a more general EA theory. It extends some of the techniques and results known for the $(1 + 1)$ EA to $(1 + \lambda)$ EAs. Clearly, additional steps are required, including the extension of these results to $(\mu + \lambda)$ EAs. From a more practical perspective, the heuristic guidelines proposed here for choosing values for λ need to be experimentally evaluated and tied more closely to properties of functions.

Acknowledgments

The first author was supported by a fellowship within the post-doctoral program of the German Academic Exchange Service (DAAD).

References

T. Corman, C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to Algotrithms, Second Edition.* New York, NY: McGraw Hill.

S. Droste, T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*. Accepted for publication.

D. B. Fogel (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Piscataway, NJ: IEEE Press.

J. Garnier, L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

T. Jansen and I. Wegener (2001). Real royal road functions — where crossover provably is essential. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO 2001*), San Francisco, CA, 1034–1042. Morgan Kaufmann.

A. Juels and M. Wattenberg (1995). Hillclimbing as a baseline method for the evaluation of stochastic optimization algorithms. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems* (*NIPS 8*), Cambridge, MA, 430–436. MIT Press.

M. Mitchell (1995). *An Introduction to Genetic Algorithms.* MIT Press.

G. Rudolph (1997). *Convergence Properties of Evolutionary Algorithms.* Hamburg, Germany: Verlag Dr. Kovač.

H.-P. Schwefel (1995). *Evolution and Optimum Seeking.* New-York, NY: Wiley.