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 Offspring Population Size in Evolutionary Algorithms

Thomas Jansen, Kenneth A. De Jong
and Ingo Wegener

No. CI-181/04

Technical Report ISSN 1433-3325 September 2004

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 Offspring Population Size in Evolutionary Algorithms

Thomas Jansen Thomas.Jansen@udo.edu
FB Informatik, LS 2, Universität Dortmund, 44221 Dortmund, Germany

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

Ingo Wegener Ingo.Wegener@udo.edu
FB Informatik, LS 2, Universität Dortmund, 44221 Dortmund, Germany

Abstract

Evolutionary algorithms (EAs) generally come with a large number of parameters that have to be set before the algorithm can be used. Finding appropriate settings is a difficult task. The influence of these parameters on the efficiency of the search performed by an evolutionary algorithm can be very high. But there is still a lack of theoretically justified guidelines to help the practitioner find good values for these parameters. One such parameter is the offspring population size. Using a simplified but still realistic evolutionary algorithm, a thorough analysis of the effects of the offspring population size is presented. The result is a much better understanding of the role of offspring population size in an EA and suggests a simple way to dynamically adapt this parameter when necessary.

1 Introduction

A persistent issue that arises in evolutionary computation (EC) applications is the choice of an appropriate population size. Depending on the particular form of the evolutionary algorithm (EA) being used, this will require one or more parameters to be set. In canonical genetic algorithms (GAs) (Goldberg 1989) and evolutionary programming (EP) (Fogel 1995), the size of the parent and offspring populations are the same. However, for evolution strategies (ESs) (Schwefel 1995), the parent and offspring population sizes (μ and λ) are traditionally chosen independently for the (μ+λ)-variant. For the (μ, λ)-variant where parents cannot survive $\lambda \geq \mu$ is a necessary condition and most often $\lambda > \mu$.

An open question is whether there is any advantage to having λ much larger than μ, and if so what should it be. Most of the GA and EP literature assumes $\mu = \lambda$ and focuses on what that value should be, while the (1+1) ES plays a major role in the ES literature. On the other hand, there are a number of empirical studies that suggest advantages when λ is considerably larger than μ,

and recommend settings such that $\frac{\lambda}{\mu} = 7$ (see, for example, (Bäck 1996)). In this paper we undertake a systematic analysis of this issue with the goal of obtaining a better understanding of the role that λ plays in the overall performance of EAs, and improving our ability to choose appropriate values for it.

In order to be able to concentrate on the effects of the value of this single parameter, it makes sense to use a simple EA that supports analysis and avoids unnecessary complications due to effects of other EA components. Consequently, in this paper we investigate what is known as a (1+λ) EA. It uses a parent population size of 1 and creates λ offspring in each generation. Individuals are represented by binary strings of fixed length n. Since the parent population size is one, there is no reproductive selection or recombination. Reproductive variation is accomplished via a standard bit-flip mutation operator that flips each bit of an individual independently of the other bits with probability $1/n$. In the survival selection step, the parent is replaced by an offspring with maximal fitness if and only if the maximal offspring fitness is greater than or equal to the parent's fitness. This is repeated until some stopping criterion is fulfilled.

For the sake of clarity we give a formal definition of the (1+λ) EA for the maximization of a fitness function $f\colon \{0,1\}^n \to \mathbb{R}$ as Algorithm 1.

Algorithm 1 ((1+λ) EA).

1. **Initialization**
   Choose $x \in \{0,1\}^n$ uniformly at random.
2. **Mutation**
   For each $i \in \{1,\ldots,\lambda\}$:
   Create $y_i \in \{0,1\}^n$ by copying x and, independently for each bit,
   flip this bit with probability $1/n$.
3. **Selection**
   If $\max\{f(y_1),\ldots,f(y_\lambda)\} \geq f(x)$, replace x by some randomly chosen y_i with maximal f-value.
4. **"Stopping Criterion"**
   Continue at line 2.

Typically, one assumes that the number of function evaluations performed is an accurate measure for the run time of an evolutionary algorithm. We adopt this point of view and define the optimization time T, or more precisely T_λ, to be the number of function evaluations performed until $f(x) = \max\{f(x') \mid x' \in \{0,1\}^n\}$ holds. One iteration of the main loop of the (1+λ) EA (lines 2–4) is called a generation. If a global optimum of f is found after G_λ generations, then $T_\lambda = 1+G_\lambda \cdot \lambda$ since there is one function evaluation for the initial population and an additional λ function evaluations in each generation. Since EAs are stochastic algorithms, T_λ is a random variable and our main interest is in determining how its expected value, $\mathrm{E}(T_\lambda)$, is affected by changes in λ. We consider two cases: when λ is set to a fixed value independent of the dimension n of the solution space being searched (e.g., $\lambda = 7$), and when λ is a (polynomial) function of n (e.g., $\lambda = n \cdot \log n$).

If the fitness evaluation of an individual can be performed independently of the evaluation of other individuals, then a clear advantage to having $\lambda > 1$ is

that the fitness evaluation of the offspring population can be done in parallel. In this case the corresponding *parallel* optimization time is simply G_λ, the number of EA generations. If $\lambda > \lambda'$ but $G_\lambda \approx G_{\lambda'}$, there is a sense of computational wastefulness of the (1+λ) EA since a parallel optimization of approximately G_λ could be obtained with the smaller number of only λ' processors. Hence, we are interested in the smallest possible λ leading to the almost smallest values of G_λ.

The analysis reported in this paper consists of a three-pronged approach, reflecting the fact that there is no single approach to EA analysis that is capable of providing a complete picture. We begin by using standard algorithm analysis tools that allow us to answer questions about $\mathrm{E}(T_\lambda)$ asymptotically as the search space dimension n increases. This leaves open the question as to whether the results are relevant for practitioners using "normal" values of n. To address this issue a second set of analysis tools is used to give precise characterizations of $\mathrm{E}(T_\lambda)$ for values of n normally encountered in practice. Finally, since both of these tool sets can only be successfully applied to relatively simple fitness functions, we follow up with an empirical study that validates the theoretical results and extends them to more complex fitness landscapes.

The result of this three-pronged analysis is a much clearer understanding of the role of λ in EAs, and suggests a simple strategy for dynamically adapting λ during an evolutionary run. That mechanism was implemented and empirically analyzed in comparison with EAs using static values for λ. These results are presented in Section 5, followed by the final section in which we present our conclusions, discuss some open problems, and make a few remarks on possible future research.

2 Asymptotic Analysis

The analyses in this section are carried out using well-known concepts and notation for the comparison of the asymptotic growth of functions (see, for example, (Cormen, Leiserson, Rivest, and Stein 2001)). A brief summary is provided here.

Definition 1. *For functions* $f\colon \mathbb{N}_0 \to \mathbb{R}$ *and* $g\colon \mathbb{N}_0 \to \mathbb{R}$ *we say that*

- $f = O(g)$ *(*f *grows not faster than* g*), iff* $\exists n_0 \in \mathbb{N}_0, c \in \mathbb{R}^+ \colon \forall n \geq n_0 \colon f(n) \leq c \cdot g(n)$,
- $f = \Omega(g)$ *(*f *grows not slower than* g*), iff* $g = O(f)$,
- $f = \Theta(g)$ *(*f *grows as fast as* g*), iff* $f = O(g)$ *and* $f = \Omega(g)$,
- $f = o(g)$ *(*f *grows slower than* g*), iff* $\lim_{n\to\infty} f(n)/g(n) = 0$*, and*
- $f = \omega(g)$ *(*f *grows faster than* g*), iff* $g = o(f)$.

In addition, since our analyses focus on characterizing $\mathrm{E}(T_\lambda)$ and $\mathrm{E}(G_\lambda)$ for different algorithms and difference fitness functions, we adopt the following notation. For an algorithm A and a function f we denote the sequential optimization time (or optimization load) of A on f by $T_{A,f}$. Analogously, we denote the parallel optimization time of A on f by $G_{A,f}$. In the case of $A =$ (1+λ) EA, we use the abbreviations $T_{\lambda,f}$ and $G_{\lambda,f}$ and even T_λ and G_λ if the choice of f is unambiguous.

We start our investigation of the impact the choice of the offspring population size has on the optimization time required for two very simple and well-known fitness functions, OneMax and LeadingOnes. Both functions f are quite easy to optimize and we expect the (1+1) EA to have a sequential optimization time $\mathrm{E}(T_{1,f})$ that is difficult to beat. However, it may well be possible that the use of a larger offspring population may reduce the expected parallel optimization time $\mathrm{E}(G_{\lambda,f})$ considerably without $\mathrm{E}(G_{\lambda,f}) \cdot \lambda$ significantly exceeding $\mathrm{E}(T_{1,f})$.

We then consider the question as to whether there are fitness landscapes f for which $\mathrm{E}(T_{\lambda,f})$ for some large λ is much smaller than $\mathrm{E}(T_{1,f})$. We construct a function SufSamp for which this can be rigorously proved. The results presented in this section have partially been published in (Jansen and De Jong 2002).

2.1 Asymptotic Analysis of LeadingOnes

We begin our analysis with the LeadingOnes function since it turns out to be particularly easy to analyze. LeadingOnes is a pseudo-boolean function of n input bits that simply counts the number of leading ones from left to right, stopping when the first zero-bit is found. More formally,

Definition 2. *The function* LeadingOnes$\colon \{0,1\}^n \to \mathbb{R}$ *is defined by* $\textsc{LeadingOnes}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$ *for all* $n \in \mathbb{N}$ *and all* $x \in \{0,1\}^n$.

For this simple fitness function the following holds.

Theorem 1. *For* $\lambda = n^{O(1)}$*, the following holds on the expected optimization load and expected parallel optimization time of the (1+λ) EA on* LeadingOnes$\colon \{0,1\}^n \to \mathbb{R}$. $E(T_{\lambda,\textsc{LeadingOnes}}) = \Theta\left(n^2 + n\lambda\right)$. $E(G_{\lambda,\textsc{LeadingOnes}}) = \Theta\left(n^2/\lambda + n\right)$.

Proof. For the (1+λ) EA, $G_{\lambda,f} = \Theta(T_{\lambda,f}/\lambda)$ holds for any offspring population size λ and any fitness function f. Thus, it suffices to prove the result on either $G_{\lambda,f}$ or $T_{\lambda,f}$. In the following we investigate the function LeadingOnes.

We begin with the upper bound. If the optimum is not reached, the probability to increase the function value in one single mutation equals $(1/n) \cdot (1 - 1/n)^{\textsc{LeadingOnes}(x)} \geq 1/(en)$: It suffices to mutate the leftmost bit with value zero while leaving the leading ones unchanged. Thus, in each generation the probability to increase the function value is bounded below by $1 - (1 - 1/(en))^{\lambda} \geq 1 - e^{-\lambda/(en)}$. We distinguish two cases with respect to the offspring population size λ. If $\lambda \geq en$, the probability to increase the function value in one generation is bounded below by $1 - e^{-1}$. Application of Chernoff bounds (Motwani and Raghavan 1995) yields that with probability $1 - e^{-\Omega(n)}$ the global optimum is reached within the first $2(1 - e^{-1})n$ generations. This proves $\mathrm{E}(G_\lambda) = O(n)$ in the case $\lambda \geq en$. In the case $\lambda < en$, the probability for increasing the function value in one generation is bounded below by $\lambda/(2en)$. This holds since for each $x \in [0,1]$, $e^{-x} \leq 1 - x/2$ holds. We conclude that with probability $1 - e^{-\Omega(n)}$ the global optimum is reached within the first $4en^2/\lambda$ generations. This proves $\mathrm{E}(G_\lambda) = O(n^2/\lambda)$ in the case $\lambda < en$. Together we have $\mathrm{E}(G_\lambda) = O(n + n^2/\lambda)$ for any value of λ.

In order to prove a lower bound the following observation is crucial: At any time, all bits of the current search point x which are right of the leftmost bit

with value zero are independent and uniformly distributed (Droste, Jansen, and Wegener 2002). The offspring population size is polynomially bounded. Thus, there exists a constant $k \in \mathbb{N}$, such that $\lambda \leq n^k$ holds. In each generation we consider the current string x and its $k+2$ leftmost zero-bits. The probability that in $\varepsilon \cdot n$ generations ($\varepsilon > 0$ constant) with $\lambda \leq n^k$ offspring each there is at least one offspring mutating all these $k+2$ bits is bounded above by ε/n. Therefore, with probability $1 - \varepsilon/n$ we do not have such a generation during the first εn generations. A mutation of at most $k+1$ leftmost zero-bits increases the function value by at most $k+1+A$ where A is the random number of bits with value 1 following each of the mutating leftmost zero-bits. The increase is smaller if one of this one-bits is flipped. Remember that the bits considered are independent and uniformly distributed. We consider all εn generations together. Using Chernoff bounds it is easy to see that with probability $1-2^{-\Omega(n)}$ there are less than $2(k+1)\varepsilon n$ additional one-bits increasing the function value. Therefore, the initial function value is increased by at most $3(k+1)\varepsilon n$ with probability $1 - O(1/n)$. Choosing ε sufficiently small we see that the unique global optimum is not reached within the first εn generations with probability $1 - O(1/n)$. We denote this event by M. We have $\mathrm{E}(G_\lambda) \geq \mathrm{E}(G_\lambda \mid M) \cdot \mathrm{Prob}(M) = \Omega(\mathrm{E}(G_\lambda \mid M))$ and get a lower bound for $\mathrm{E}(G_\lambda)$ this way. We see that the optimum cannot be reached before $\Omega(n)$ mutations that increase the function value. This immediately implies $\Omega(n)$ as lower bound on the number of generations. Furthermore, such a mutation requires that the leftmost zero-bit flips and occurs therefore with probability at most $1/n$. Thus, the probability to have such a mutation within one generation is bounded above by λ/n. This implies $\Omega(n^2/\lambda)$ as lower bound on the expected number of generations. □

From this theorem we see that $\mathrm{E}(G_\lambda)$ only gets smaller when $\lambda = \Omega(n)$. At the same time we see that $\mathrm{E}(T_\lambda)$ does not increase significantly as long as $\lambda = O(n)$ holds. Thus, $\mathrm{E}(G_\lambda)$ is minimized when $\lambda = \Theta(n)$. The fact that $\mathrm{E}(T_\lambda)$ does not increase significantly as long as $\lambda = O(n)$ means that the performance of (1+λ) EAs for which $1 \leq \lambda \leq n$ are asymptotically equivalent when run on a sequential machine. This means that there is no particular reason in this case to set the value of λ to a constant (e.g., $\lambda = 1$) or to maintain a fixed ratio such as $\frac{\lambda}{\mu} = 7$.

Finally, the fact that the "cut-off point" $O(n)$, the value of λ where increasing λ only creates costs and has no benefits, is not an arbitrary one. It is in the same order of growth as the reciprocal of the mutation "success probability", the probability p that a mutation will result in an increase in fitness. In the case of LeadingOnes, increasing the fitness value is always advantageous. Hence, choosing $\lambda = \frac{1}{p}$ cannot increase the expected optimization load $\mathrm{E}(T_\lambda)$ significantly since, on average, $1/p$ mutations are needed to improve fitness regardless of whether these mutations are done sequentially or in parallel.

2.2 Asymptotic Analyses of OneMax

For our second asymptotic analysis we focus on one of the best-known simple fitness landscapes, OneMax, that simply counts the number of ones in an n-bit string. More formally,

Definition 3. *The function* $\textsc{OneMax}\colon \{0,1\}^n \to \mathbb{R}$ *is defined by* $\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i$ *for all* $n \in \mathbb{N}$ *and all* $x \in \{0,1\}^n$.

Interestingly, the "cut-off point" for the (1+λ) EA is much harder to predict for OneMax than for LeadingOnes. This is due to the fact that the success probability of mutation changes during a run. For LeadingOnes, the success probability is always bounded below by $1/(en)$ and bounded above by $1/n$. For OneMax it may be as small as almost $1/(en)$ (for $x \in \{0,1\}^n$ with $\textsc{OneMax}(x) = n-1$) or as large as $1-(1-1/n)^n$ (for the all zero string 0^n). Typically, it will be $\Theta(1)$ in the beginning and drop to $\Theta(1/n)$ in the end. Which maximal value for λ will not be harmful on average is not easy to see. Therefore, we approach our result in several steps. First of all, we show that $\mathrm{E}(T_{A,\textsc{OneMax}}) = \Omega(n \log n)$ holds for any evolutionary algorithm A that initializes its population uniformly at random and is based on standard mutations with mutation probability $1/n$ and selection only. We call such EAs *standard mutation* EAs.

Theorem 2. *Let* $f\colon \{0,1\}^n \to \mathbb{R}$ *be a function with unique global optimum* $x^* \in \{0,1\}^n$. *Let* A *be a standard mutation EA.* $E(T_{A,f}) = \Omega(n \log n)$.

Proof. Let μ be the initial population size of A. The probability to have x^* in the initial population is bounded above by $\mu/2^n$. For $\mu \geq n \log n$, A performs on average $\Omega(n \log n)$ function evaluations for the initial population without finding x^*. Thus, we only have to consider the case $\mu < n \log n$. With probability $1 - e^{-\Omega(n)}$ the minimal Hamming distance between x^* and a member of the population is bounded below by $n/3$. Thus, there are at least $n/3$ bits which all need to be mutated at least once in order to find x^*. The probability not to mutate a specific bit in t mutations equals $(1-1/n)^t \geq e^{-t/(n-1)}$. Thus, with probability at least $1/n$ a specific bit is not mutated at all within $t = (n-1)\ln n$ mutations. Therefore, with probability $1-(1-1/n)^{n/3} \geq 1 - e^{-1/3}$ there is a bit within $n/3$ bits that is not mutated at all in $(n-1)\ln n$ mutations. This implies $\mathrm{E}(T_{A,f}) \geq (1-e^{-\Omega(n)}) \cdot (1-e^{-1/3}) \cdot (n-1)\ln n = \Omega(n \log n)$. □

For the (1+λ) EA on OneMax, an upper bound of $O(n \log n + n\lambda)$ is easy to prove. We simply follow the same basic idea as for LeadingOnes.

Theorem 3. $E(T_{\lambda,\textsc{OneMax}}) = O(n \log n + n\lambda)$.
$E(G_{\lambda,\textsc{OneMax}}) = O((n \log n)/\lambda + n)$.

Proof. Let d denote the Hamming distance between the current string x and the global optimum 1^n. Obviously, the probability to increase the function value in one single mutation is bounded below by $(d/n)\cdot(1-1/n)^{n-1} \geq d/(en)$. Thus the probability to increase the function value in one generation is bounded below by

$$1 - \left(1 - \frac{d}{en}\right)^{\lambda} \geq 1 - e^{-d\lambda/(en)} \geq 1 - \frac{1}{1 + d\lambda/(en)} = \frac{d\lambda}{en + d\lambda}.$$

Thus,

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

□

From Theorem 3 we see that the "cut-off point" for λ is $\Omega(\log n)$. For offspring population sizes $\lambda = \omega(\log n)$ the expected optimization load may be $\omega(n \log n)$ — but we do not know. The proof of Theorem 3 only takes mutations of single bits into account. One may speculate that in the beginning when the success probability is still large simultaneous mutations of several bits may speed up the optimization. Thus, even larger values of λ may be helpful. We prove this idea to be correct for slightly larger offspring population sizes.

Theorem 4. *For $\lambda = O((\ln n)(\ln\ln n)/\ln\ln\ln n)$: $E(T_{\lambda,\textsc{OneMax}}) = O(n \log n)$ and $E(G_{\lambda,\textsc{OneMax}}) = O((n \log n)/\lambda)$.*

Proof. For $\lambda = O(\log n)$ the result follows from Theorem 3. So, we assume $\lambda \geq \ln n$ and define $\gamma := \lfloor \lambda / \ln n \rfloor$. Obviously, $\gamma \geq 1$ holds.

We divide a run of the (1+λ) EA into two disjoint phases. The first phase starts with random initialization and ends when $\textsc{OneMax}(x) > n - n/\ln\ln n$ holds for the first time. The second phase starts after the first phase is finished and ends when the global optimum is found. Let $G_{1,\lambda}$ denote the number of generations in the first phase. Let $G_{2,\lambda}$ be defined in the same way for the second phase. Let $T_{1,\lambda}$ and $T_{2,\lambda}$ denote the number of function evaluations in the first and second phase. Obviously, we have $G_\lambda = G_{1,\lambda} + G_{2,\lambda}$.

In order to get an upper bound on $\mathrm{E}(G_{2,\lambda})$ we can use the same arguments and estimates as in the proof of Theorem 3. Note that we have $d \leq n/\ln\ln n$ by the definition of the second phase. This yields

$$\mathrm{E}(G_{2,\lambda}) = \sum_{d=1}^{n/\ln\ln n} \frac{en + d\lambda}{d\lambda} = \frac{n}{\ln\ln n} + \frac{en}{\lambda} \sum_{d=1}^{n/\ln\ln n} \frac{1}{d} = O\left(\frac{n}{\ln\ln n} + \frac{n \log n}{\lambda}\right).$$

Since we have $\lambda = O((\ln n \ln\ln n)/\ln\ln\ln n)$, $\mathrm{E}(T_{2,\lambda}) = O(n \log n)$ follows.

Now, we give an upper bound for the first phase under the assumption that $\lambda \leq (\ln n)(\ln\ln n)/(2\ln\ln\ln n)$ holds. In the first phase we have $d \geq n/\ln\ln n$ by definition of the first phase. Thus, the probability to increase the function value by at least γ in one single mutation is bounded below by

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

Thus, the probability to increase the function value by at least γ in one generation is bounded below by

$$1 - \left(1 - e^{-(1+\gamma\ln\gamma+\gamma\ln\ln\ln n)}\right)^\lambda \geq \frac{\lambda}{e\ln n + \lambda} \geq \frac{1}{e+1}.$$

Here we need $\gamma \leq (\ln\ln n)/(2\ln\ln\ln n)$. Obviously, after at least n/γ such generations the first phase ends. This implies $\mathrm{E}(T_{1,\lambda}) = O(\lambda \cdot n/\gamma) = O(n \log n)$.

We still need an upper bound on the length of the first phase in the case $\lambda > (\ln n)(\ln\ln n)/(2\ln\ln\ln n)$. We define $\varepsilon := (\ln n)(\ln\ln n)/(2\lambda\ln\ln\ln n)$. According to our assumptions, we have $\varepsilon = \Theta(1)$. We redefine $\gamma := \lfloor \varepsilon\lambda/\ln n \rfloor$ and have $1 < \gamma \leq (\ln\ln n)/(2\ln\ln n)$. Now we can repeat the arguments from above and get $\mathrm{E}(T_{1,\lambda}) = O(\varepsilon^{-1} n \log n) = O(n \log n)$. □

We see that, with offspring population size $\lambda = O((\ln n)(\ln\ln n)/\ln\ln\ln n)$, a (1+$\lambda$) EA is still efficient. Now, we prove that $\Theta((\ln n)(\ln\ln n)/\ln\ln\ln n)$ is indeed the cut-off point: larger values of λ significantly increase the expected optimization load.

Theorem 5. *For* $\lambda = \omega((\ln n)(\ln\ln n)/\ln\ln\ln n)$*:* $E(T_{\lambda,\text{OneMax}}) = \omega(n\log n)$ *and* $E(G_{\lambda,\text{OneMax}}) = \omega((n/\lambda)\log n)$

Proof. The lower bound $\mathrm{E}(G_\lambda) = \Omega(n/\log n)$ is easy to prove. This is sufficient for $\lambda = \omega(\log^2 n)$, only.

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 $d \in \{1,2,\ldots,n - \text{OneMax}(x)\}$. The probability for such a mutation within n generations is bounded above by $\lambda n(e/d)^d$. The probability that the Hamming distance to the global optimum is decreased by at least $\log n$ in one single generation during the first n generations is therefore bounded above by

$$\sum_{d=\log n}^{n} \lambda n(e/\log n)^{\log n} = e^{-\Omega((\log n)(\log\log n))}.$$

With probability $1 - e^{-\Omega(n)}$, after random initialization the Hamming distance to the optimum is bounded below by $n/3$. We have seen that in a single generation the Hamming distance is decreased by at most $\log n$ with probability $1 - e^{-\Omega((\log n)(\log\log n))}$. This implies $\mathrm{E}(G_\lambda) = \Omega(n/\log n)$ as claimed.

Now we come to the tight result. 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. This corresponds to the lower bound method based on the expected advance used by Jansen and Wegener (2000).

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 $\text{OneMax}(y) \geq \text{OneMax}(x) + d$ is bounded above by $\binom{n/e}{d} \cdot 1/n^d < 1/d^d$.

Consider g generations of the (1+λ) 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 these g generations by means of Hamming distance, i. e. $D_{\lambda,g,x} := \text{OneMax}(x') - \text{OneMax}(x)$. Obviously, $D_{\lambda,g,x}$ depends on x and we have $\mathrm{Prob}(D_{\lambda,g,x} \geq d) \geq \mathrm{Prob}(D_{\lambda,g,y} \geq d)$ for all $d \in \{0,1,\ldots,n\}$ and all $x, y \in \{0,1\}^n$ with $\text{OneMax}(x) \leq \text{OneMax}(y)$. Since the function value of the current string of the (1+λ) EA can never decrease, $\mathrm{E}(D_{\lambda,g,x}) \leq g \cdot \mathrm{E}(D_{\lambda,1,x})$ holds for all λ, g and x. So, we concentrate on $\mathrm{E}(D_{\lambda,1,x})$ 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 $\mathrm{Prob}(D_{\lambda,1,x} \geq d) < \lambda/d^d$ holds for all x with $\text{OneMax}(x) \geq n - \lceil n/e\rceil$. From now on, we always assume that $\text{OneMax}(x)$ is bounded below in this way.

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

$$\begin{aligned}\mathrm{E}\left(D_{\lambda,1,x}\right) &< \left(\sum_{i=1}^{((3\ln\lambda)/\ln\ln\lambda)-1} 1\right) + \left(\sum_{i=(3\ln\lambda)/\ln\ln\lambda}^{((\lambda\ln\lambda)/\ln\ln\lambda)-1} 1/\lambda\right) + \left(\sum_{i=(\lambda\ln\lambda)/\ln\ln\lambda}^{n} 1/n\right) \\ &< (3\ln\lambda/(\ln\ln\lambda) - 1) + (\ln\lambda/\ln\ln\lambda) + 1 < 4\ln\lambda/\ln\ln\lambda\end{aligned}$$

for the expected advance in one generation.

Of course, $\mathrm{E}\left(G_\lambda\right) \geq t \cdot \mathrm{Prob}\left(G_\lambda \geq t\right)$ holds for all values of t. As we argued above, with probability at least $1/2$ at some point of time we have some x with $\textsc{OneMax}(x) \in \{n - \lceil n/e \rceil, \ldots, n - \lceil n/(2e) \rceil\}$ as current string x. Thus, $\mathrm{Prob}\left(G_\lambda \geq t\right) \geq \mathrm{Prob}\left(D_{\lambda,t,x} < n/e\right)$ holds, if x is some string with at least n/e zero-bits. This yields

$$\mathrm{E}\left(G_\lambda\right) \geq (t/2) \cdot \mathrm{Prob}\left(D_{\lambda,t,x} < n/e\right) = (t/2) \cdot \left(1 - \mathrm{Prob}\left(D_{\lambda,t,x} \geq n/e\right)\right).$$

By Markov's inequality we have

$$\mathrm{E}\left(G_\lambda\right) \geq (t/2) \cdot \left(1 - \mathrm{E}\left(D_{\lambda,t,x}\right)/(n/e)\right) \geq (t/2) \cdot \left(1 - e \cdot t \cdot \mathrm{E}\left(D_{\lambda,1,x}\right)/n\right).$$

Together with our estimation for $\mathrm{E}\left(D_{\lambda,1,x}\right)$ we have $\mathrm{E}\left(G_\lambda\right) \geq (t/2) \cdot (1 - 4e \cdot t \cdot \ln\lambda/(n \cdot \ln\ln\lambda))$. We set $t := (n \ln\ln\lambda)/(8e \ln\lambda)$ and get $\mathrm{E}\left(G_\lambda\right) \geq n \ln\ln\lambda/(32e \ln\lambda)$ which implies $\mathrm{E}\left(T_\lambda\right) = \Omega\left(n\lambda \ln\ln\lambda/\ln\lambda\right)$. It is easy to see that this implies $\mathrm{E}\left(T_\lambda\right) = \omega(n \log n)$ for $\lambda = \omega((\ln n)(\ln\ln n)/\ln\ln\ln n)$ as claimed. □

So, for OneMax we see that the cut-off point for λ is $\Theta((\ln n)(\ln\ln n)/\ln\ln\ln n)$. While smaller than the $\Theta(n)$ cutoff for LeadingOnes, there is still significant opportunity for speedup in a parallel computing environment as n increases, and no theoretical reason to set λ to a small constant value.

2.3 Asymptotic Analysis of SufSamp

For both of the previous fitness functions, the (1+1) EA can only be outperformed by the (1+λ) EA if a parallel computer is used. It is therefore interesting to see whether there exist fitness functions where the use of an offspring population with size $\lambda > 1$ reduces the expected optimization time significantly — even on a sequential computer. In order to construct an appropriate example function one needs an idea of where such an advantage might come from. One possibility is a fitness landscape in which "helpful hints" are more difficult to find than misleading ones. If sufficient sampling of the search space in the neighborhood of the current population is required to avoid misleading hints (e.g., paths to local optima), a (1+λ) EA may well be superior to a (1+1) EA. We prove that this idea is correct by first defining an example function, SufSamp that has exactly this property and second analyzing the performance of the (1+1) EA and the (1+λ) EA with sufficiently large offspring population size λ.

The intuition for this function is quite simple as illustrated in Figure 1. We want it to consist of a narrow main path that leads to the optimum. However, while following that path, the algorithm is confronted with multiple branch

points, each with the property that from it there is 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 path at the branch point.

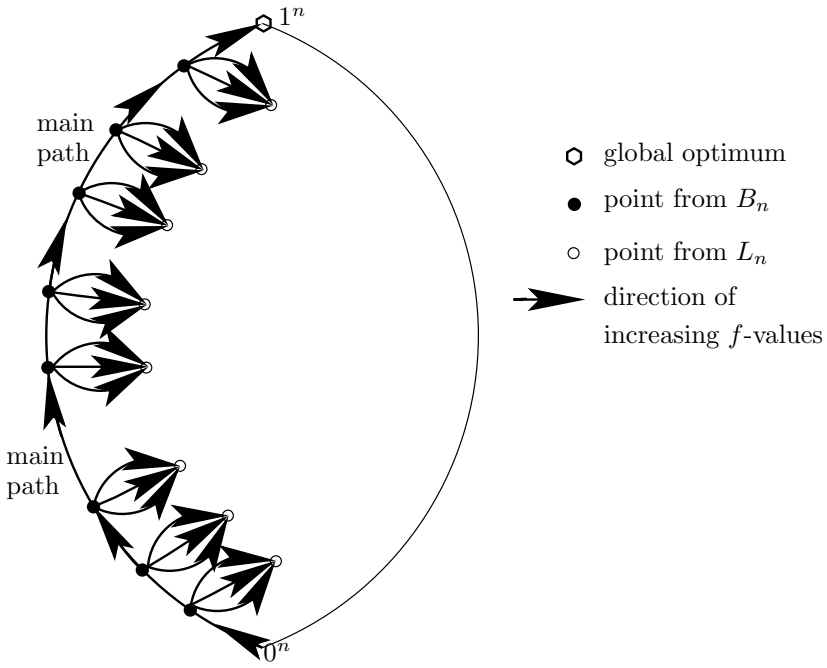

Figure 1: Core function f.

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

$$f(x) := \begin{cases} (i+3)n + |x| & \text{if } (x = 0^{n-i}1^i \text{ with } 0 \leq i \leq n) \text{ or} \\ & (x = y0^{n-i-k}1^i \text{ with } i \in \{k, 2k, \ldots, (k-2)k\},\ y \in \{0,1\}^k) \\ 0 & \text{otherwise.} \end{cases}$$

The core function f contains one main path 0^i1^{n-i}. There are about $\sqrt{n}$ points on this path that are of special interest. At these points it is not only beneficial to add another 1 on the right side. The function value is also increased by flipping any of the leftmost $\lfloor \sqrt{n} \rfloor$ bits. Thus, at these points there is a variety of uphill paths to choose 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 *branch points*.

Definition 5. *For $n \in \mathbb{N}$ we define $k := \lfloor \sqrt{n} \rfloor$ and call a point $x \in \{0,1\}^n$ a* branch point of dimension n *iff $x = 0^{n-i}1^i$ holds for some $i \in \{2k, 3k, \ldots, (k-1)k\}$. Let B_n denote the set of all branch points of dimension n.*

At the branch points a (1+λ) EA may proceed toward a local optimum or the global one. It is important to know what the probabilities are for the two different possibilities:

Lemma 1. *Let $n \in \mathbb{N}$, $k := \lfloor \sqrt{n} \rfloor$, B_n the set of branch points of dimension n as defined in Definition 5, $x \in B_n$, and $\lambda\colon \mathbb{N} \to \mathbb{N}$ be a function that has a polynomial upper bound. Consider a (1+λ) EA ($\lambda = \lambda(n)$) with current string x optimizing the function $f\colon \{0,1\}^n \to \mathbb{R}$ as defined in Definition 4. Let $y = y_1 y_2 \cdots y_n$ be the first point with $f(y) > f(x)$ reached by the (1+λ) EA. Let p be the probability that $\textsc{OneMax}(y_1 y_2 \cdots y_k) > 0$ holds, then:*

$$\frac{1 - e^{-\lambda/(en)}}{1 - e^{-2\lambda/k}} \leq p \leq \frac{1 - e^{-\lambda/n}}{1 - e^{-\lambda/(ek)}}.$$

Proof. Let $x \in B_n$ be the current string of the (1+λ) EA operating on f. The points in B_n are ordered according to the function value. We prove the claim under the assumption that no mutation of at least k bits occurs before we have a generation that produces an offspring with larger function value. This changes all probabilities involved by a factor of at most $1 + 2^{-\Omega(k)}$. It allows us to ignore mutations leading to points with function values equal to or greater than the function value of the next branch point.

Let $x^{(1)}, x^{(2)}, \ldots, x^{(\lambda)}$ be the offspring generated in one generation by independent mutations of x. We use the notation $x^{(i)} = x_1^{(i)} x_2^{(i)} \cdots x_n^{(i)}$ to denote the specific bits. Let A_i denote the event that $f(x^{(i)}) > f(x)$ and $\textsc{OneMax}(x_1^{(i)} \cdots x_k^{(i)}) = 0$ hold. In this case $x^{(i)}$ is on the main path behind x. Let B_i denote the event that $f(x^{(i)}) > f(x)$ and $\textsc{OneMax}(x_1^{(i)} \cdots x_k^{(i)}) > 0$ hold. In this case $x^{(i)}$ is on one of the misleading paths starting in x. With these definitions we have

$$p = \text{Prob}\left(\bigcup_{i=1}^{\lambda} A_i \mid \bigcup_{i=1}^{\lambda} (A_i \cup B_i)\right) = \frac{\text{Prob}\left(\bigcup_{i=1}^{\lambda} A_i\right)}{\text{Prob}\left(\bigcup_{i=1}^{\lambda} (A_i \cup B_i)\right)}. \tag{1}$$

Note that the events concerning different offspring are completely independent and have the same probability. Thus,

$$\text{Prob}\left(\bigcup_{i=1}^{\lambda} A_i\right) = 1 - \text{Prob}\left(\bigcap_{i=1}^{\lambda} \overline{A_i}\right) = 1 - \text{Prob}\left(\overline{A_1}\right)^{\lambda}$$

holds. Upper and lower bounds for $\text{Prob}\left(\overline{A_1}\right)$ are easy to find. The event A_1 requires that all the one-bits in x do not change their values and any positive number of consecutive zero-bits become one-bits in one mutation. This yields

$$\frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \leq \text{Prob}\left(A_1\right) \leq \sum_{i=1}^{k} \left(\frac{1}{n}\right)^{i} \left(1 - \frac{1}{n}\right)^{n-i}$$

and we have

$$1 - \frac{1}{en} \geq \text{Prob}\left(\overline{A_1}\right) \geq 1 - \frac{1}{n}$$

as bounds. Therefore,

$$1 - e^{-\lambda/(en)} \leq \text{Prob}\left(\bigcup_{i=1}^{\lambda} A_i\right) \leq 1 - e^{-\lambda/n}$$

holds.

Of course,

$$\text{Prob}\left(\bigcup_{i=1}^{\lambda} (A_i \cup B_i)\right) = 1 - \text{Prob}\left(\bigcap_{i=1}^{\lambda} (\overline{A_i} \cap \overline{B_i})\right)$$

holds and we see that the events $(\overline{A_i} \cap \overline{B_i})$ are completely independent and have the same probability. This leads to

$$\text{Prob}\left(\bigcup_{i=1}^{\lambda} (A_i \cup B_i)\right) = 1 - \text{Prob}\left(\overline{A_1} \cap \overline{B_1}\right)^{\lambda}.$$

For $A_1 \cup B_1$ it is easy to see that

$$\begin{aligned}
&\frac{k}{n}\left(1-\frac{1}{n}\right)^{n-1} \leq \text{Prob}\left(A_1 \cup B_1\right) \\
&\leq \left(\sum_{j=1}^{k}\binom{k}{j}\left(\frac{1}{n}\right)^{j}\left(1-\frac{1}{n}\right)^{n-j}\right)+\left(\sum_{i=1}^{k}\sum_{j=0}^{k}\binom{k}{j}\left(\frac{1}{n}\right)^{i+j}\left(1-\frac{1}{n}\right)^{n-i-j}\right)
\end{aligned}$$

holds and we have

$$1 - \frac{k}{en} \geq \text{Prob}\left(\overline{A_1} \cap \overline{B_1}\right) \geq 1 - \frac{2k}{n}$$

as bounds. Plugging these bounds into equation (1) completes the proof. □

Assume that the (1+λ) EA happens to continue in the direction of some $y0^{n-i-\lfloor\sqrt{n}\rfloor}1^i$ 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., $y0^{n-i-\lfloor\sqrt{n}\rfloor}1^i$ with $y = 1^{\lfloor\sqrt{n}\rfloor}$. The set of these points is called L_n.

Definition 6. *For $n \in \mathbb{N}$ we define $k := \lfloor\sqrt{n}\rfloor$ and*

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

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 chooses 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 another function SufSamp in a way that most starting points lead to the main path defined by f. The function is named SufSamp since the main idea (already of f) is that only sufficient sampling of the current search point's neighborhood allows an evolutionary algorithm to remain on the main path.

Definition 7. *For $n \in \mathbb{N}$ we define $m' := \lfloor n/2 \rfloor$, $m'' := \lceil n/2 \rceil$, and* SufSamp$\colon \{0,1\}^n \to \mathbb{R}$ *by*

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

for all $x = x_1 x_2 \cdots x_n \in \{0,1\}^n$ with $x' := x_1 x_2 \cdots x_{m'} \in \{0,1\}^{m'}$ and $x'' := x_{m'+1} x_{m'+2} \cdots x_n \in \{0,1\}^{m''}$.

We double the length of each bit string 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 6. *The probability that the (1+1) EA optimizes the function* SufSamp$\colon \{0,1\}^n \to \mathbb{R}$ *within $n^{O(1)}$ function evaluations is bounded above by $2^{-\Omega(\sqrt{n})}$.*

Proof. Our proof strategy is the following. First, we consider the expected optimization time of the (1+1) EA on the function SufSamp$\colon \{0,1\}^n \to \mathbb{R}$ under the condition that at some point of time the current string $x = x'x''$, with x' and x'' defined as in Definition 7, 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 SufSamp, 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 $n^{-\sqrt{n/2}+k+1} = 2^{-\Omega(\sqrt{n} \log n)}$.

The initial current string $x = x'x''$ after random initialization is some string with $x' \neq 0^{m'}$ and $m''/3 < \textsc{OneMax}(x'') < 2m''/3$ with probability $1 - 2^{-m'}$. Then, the function value is given by $n - \textsc{OneMax}(x'')$. With probability $1 - 2^{-\Omega(n)}$ then some point x on the main path below the first branch point $b = 0^{n-2k}1^{2k}$ is reached within $O\left(n^2 \log n\right)$ steps. If no other string y with $\textsc{SufSamp}(y) \geq \textsc{SufSamp}(b)$ is reached before, the statement holds by known results on OneMax. We estimate the probability to reach some y with $\textsc{SufSamp}(y) \geq \textsc{SufSamp}(b)$. We consider levels of strings that contain all strings with a given number of zero-bits. On each level, for symmetry reasons each string has the same probability to be reached as long as no string y with $\textsc{SufSamp}(y) \geq \textsc{SufSamp}\left(0^n\right)$ is reached. We are only interested in levels with less at least $m''/3$ and at most $m'' - 2k$ zeros. Such levels contain $2^{\Omega(k)}$ strings. Thus, the probability to reach a string on some path is $2^{-\Omega(k)} = 2^{-\Omega(\sqrt{n})}$.

At each branch 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 $1 - \Theta(1/\sqrt{n})$. This follows from the proof of Lemma 1. The probability

to proceed with such strings instead of returning to a string with k zeros at these bit positions increases. In fact, it is easy to see that with probability $1 - O(1/\sqrt{n})$ the (1+λ) EA reaches a point in A before reaching some point with k zeros at these positions. Since we have $k-1 > \sqrt{n}/2$ branch points, we conclude that the probability that the (1+1) EA does not reach some point in A is bounded above by $2^{-\Omega(\sqrt{n}\log n)}$ under the described circumstances. Combining all estimations completes the proof. □

Theorem 7. *For all constants $\varepsilon > 0$, the probability that the (1+λ) EA with $\lambda \geq cn \log n$, $c > 0$ a constant sufficiently large, optimizes the function* SufSamp$\colon \{0,1\}^n \to \mathbb{R}$ *within $O(n \cdot \lambda)$ function evaluations 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 3 that the expected number of steps the (1+λ) EA needs to reach the all zero string is $O(n \cdot \lambda)$. In a fashion similar to the proof of Theorem 6 we can conclude that the all zero string is reached within $c'n \cdot \lambda$ 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, that the expected number of steps until the global optimum is reached is bounded above by $O(n \cdot \lambda)$. So, under the assumption that no such string is reached, we have similarly to the reasoning above that the global optimum is reached within $O(n \cdot \lambda)$ steps with probability at least $1 - \varepsilon$, where ε is an arbitrarily small positive constant. We know from Lemma 1 that the probability to reach some x where this assumption is not met is bounded above by $e^{-\Omega(\lambda/n)}$ at each branch point. Since there are less than $\sqrt{n}$ branch points, we have that with probability at least $1 - \sqrt{n} \cdot e^{-\Omega(\lambda/n)}$ no such point becomes current point x at any branch point. Since we have $\lambda/n \geq c \log n$ with $c > 0$ sufficiently large, this probability is sufficiently close to 1. Combining these estimations completes the proof. □

So, our intuition can now be seen to be correct. Larger offspring populations can be of considerable benefit with (1+λ) EAs on fitness landscapes in which additional sampling is required to identify the "correct" direction in which to continue searching.

2.4 SufSamp′

The perceptive reader may have already noted that the same ideas used to construct SufSamp can also be used to construct a landscape that misleads the same (1+λ) EA that easily finds the global optimum of SufSamp. Consider a different fitness function f that can be described as a "small" modification of SufSamp. Consider SufSamp and all strings $x = x'x''$ where x'' is a branch 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 6 and Theorem 7 rely on the fact that the (1+1) EA reaches some string $x'x'''$ with high probability whereas the (1+λ) EA does not. We see that a function SufSamp′ that has all such points $x'x'''$ as global optimum and is equal to SufSamp on all other points is obviously optimized within $O(n^2)$

function evaluations by the (1+1) EA with probability converging to 1, whereas the (1+λ) EA fails to optimize SufSamp$'$ within a polynomial number of steps with probability converging to 1, given that $\lambda = \Omega(n \log n)$ holds.

Note that it is not the existence of local optima that makes the example functions SufSamp and SufSamp$'$ so difficult for some algorithms. It is not difficult to come up with unimodal versions of SufSamp and SufSamp$'$: the idea is to connect the local optima of SufSamp and SufSamp$'$ with a unique global optimum via very long paths that take the (1+λ) EA very long time (Droste, Jansen, and Wegener 1998)).

2.5 Asymptotic Analysis Summary

The asymptotic analysis in this section makes it very clear that there is no simple answer to the question of an appropriate value for λ in a (1+λ) EA. For simple landscapes such as OneMax and LeadingOnes, settings in the range $1 \leq \lambda \leq \log n$ yield optimization times that are all asymptotically equivalent on serial machines. Hence, conventional settings such as 1 or 7 seem adequate in such cases but not required, while on parallel machines there is a clear advantage to increase λ to the point at which additional parallelism provides no additional speedup. On the other hand, the SufSamp landscape illustrates that even on sequential machines there are situations in which large values of λ are required to find the global optimum with high probability.

3 Exact Analysis

All the results from the previous section are asymptotic ones. That means that they hold for sufficiently large values of n and they neglect multiplicative constants. Therefore, they may not be accurate for small values of n. For example, for OneMax and LeadingOnes, it may be the case that for some values of n and some values of λ the expected optimization load may in fact be less for the (1+λ) EA than for the (1+1) EA — in spite of our asymptotic results. However, there is nothing in our analysis that supports such a belief. In fact, for functions as simple as OneMax and LeadingOnes, an exact analysis is possible that shows that this is not the case. So, in this section we revisit the performance of the (1+λ) EA on OneMax and LeadingOnes and prove that larger offspring population sizes cannot decrease the optimization time for any value of n.

3.1 Exact Analysis for OneMax

Theorem 8. *For any $\lambda \geq 1$, any $t \in \mathbb{N}$, and any $k \in \mathbb{N}$:*
Prob $(T_{\lambda,\text{OneMax}} < t) \geq$ *Prob* $(T_{k\cdot\lambda,\text{OneMax}} < t)$.

Proof. Let x_t denote the current generation of the (1+1) EA after the t-th function evaluation (with x_1 denoting the initial population). The main observation is the following. For all $x, x' \in \{0,1\}^n$, all $t, k \in \mathbb{N}$

$$\begin{aligned} &\text{OneMax}(x) \geq \text{OneMax}(x') \\ &\Rightarrow \text{Prob}\left(\text{OneMax}(x_t) \geq k \mid x_1 = x\right) \geq \text{Prob}\left(\text{OneMax}(x_t) \geq k \mid x_1 = x'\right) \end{aligned} \tag{2}$$

holds. This can be proved via induction on t. Let $t = 2$. Because of symmetry of OneMax with respect to the bit positions we can assume w. l. o. g. that $x = 1^i 0^{n-i}$ and $x' = 1^{i+d} 0^{n-i-d}$ with $d \geq 0$. If $d = 0$, then $x = x'$ and (2) is obvious. Let $d > 0$. Then we may consider a so-called "coupled mutation", i. e., the decision whether bit i of x and bit i of x' flips is done by the same experiment. This leads to the mutants $m(x)$ and $m(x')$ of x and x' to the same probability distributions than usual mutations. Hence, $m(x)$ and $m(x')$ agree in the prefix of length i and in the suffix of length $n-i-d$. The number of ones in the middle part of $m(x)$ is binomially distributed with the parameters d and $1 - 1/n$ and for $m(x')$ we have the binomial distribution with the parameters d and $1/n$. If $n \geq 2$, the probability of at least r ones, $0 \leq r \leq d$, is for $m(x)$ at least as large as for $m(x')$. Hence,

$$\text{Prob}\left(\textsc{OneMax}(m(x)) \geq k\right) \geq \text{Prob}\left(\textsc{OneMax}(m(x')) \geq k\right).$$

Since $\textsc{OneMax}(x) \geq \textsc{OneMax}(x')$ and x_2 is the better one among x and $m(x)$, similarly for x'_2, (2) holds for $t = 2$.

For the induction step, we investigate the probabilities p_l, $0 \leq l \leq n$, that x_{t-1} contains exactly l ones, similarly for p'_l and x'_{t-1}. We arrange the probabilities as shown in Fig. 2 for $n = 5$. The crucial point is the following observation.

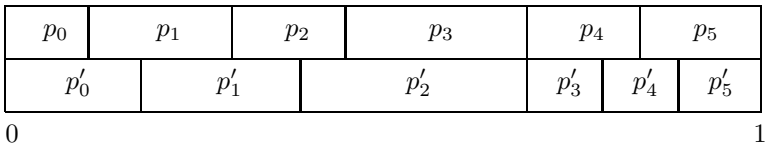

Figure 2: Comparing p_l and p'_l.

Consider some $r \in [0, 1]$. If $r \in I_i$ and $r \in I'_j$, then by the induction hypothesis $j \leq i$. In order to pick x_{t-1} and x'_{t-1} we pick $r \in [0, 1]$ uniformly at random. If $r \in I_i$ and $r \in I'_j$, then we choose some x_{t-1} with i ones and some x'_{t-1} with j ones. For each r we can apply arguments from the induction base and, finally, we have proved (2).

A direct consequence is that

$$\begin{aligned}&\textsc{OneMax}(x) \geq \textsc{OneMax}(x')\\ &\qquad \Rightarrow \text{Prob}\left(T_{1,\textsc{OneMax}} \leq t \mid x_0 = x\right) \geq \text{Prob}\left(T_{1,\textsc{OneMax}} \leq t \mid x_0 = x'\right)\end{aligned}$$

holds.

To prove the theorem we start with the special case $\lambda = 1$ and compare a (1+1) EA and a (1+λ) EA. Due to the definition of the (1+λ) EA, we have $x_i = x_{i+1} = \cdots = x_{i+\lambda}$ for all $i \not\equiv 1 \mod \lambda$. Only for $i \equiv 1 \mod \lambda$ the current population x_i is replaced by an offspring (from the last λ created) with maximal function value. Since $x_t \neq x_{t+1}$ implies $\textsc{OneMax}(x_t) \geq \textsc{OneMax}(x_{t+1})$, the claim follows.

The same holds for the comparison of the (1+λ) EA and the (1+$k\lambda$) EA, since the (1+λ) EA always updates earlier than the (1+$k\lambda$) EA. □

We see that increasing λ by some factor can never decrease the expected optimization load. Furthermore, it is not difficult to see that for $t > 1$ the

inequalities in the first statement can all be made strict. The same holds for the second statement and $t > \lambda$.

We do not make a similar statement for arbitrary offspring population sizes $\lambda < \lambda'$. It is not clear that an analogy to Theorem 8 holds in this general case. Updating the current string later may deliver an advantage in the beginning. Since this seems to vanish in later generations we make the following conjecture.

Conjecture 1. *For any $\lambda, \lambda' \in \mathbb{N}$ with $\lambda \leq \lambda'$, there exists an $t_0 \in \mathbb{N}$, such that for all $t \in \mathbb{N}$ with $t \geq t_0$*

$$Prob\left(T_{\lambda,\textsc{OneMax}} < t\right) \geq Prob\left(T_{\lambda',\textsc{OneMax}} < t\right)$$

holds.

3.2 Exact Analysis for LeadingOnes

Theorem 9. *For any $\lambda \geq 1$, any $t \in \mathbb{N}$, and any $k \in \mathbb{N}$:*

$$Prob\left(T_{\lambda,\textsc{LeadingOnes}} < t\right) \geq Prob\left(T_{k\cdot\lambda,\textsc{LeadingOnes}} < t\right).$$

Proof. The first observation is, that equation (2) does not hold for LeadingOnes. This is due to the missing correspondence between the function value and the Hamming distance to the optimum. Obviously, $\textsc{LeadingOnes}(10^{n-1}) = 1$ and $\textsc{LeadingOnes}(01^{n-1}) = 0$, but we expect the success probability after t steps to be greater if the initial population is 01^{n-1}. However, we already observed that for any (1+λ) EA the distribution of all bits right to the leftmost zero-bit is uniform (Theorem 1). Taking this into account the same correspondence between the function value and the probability of mutations to strings with larger fitness exists for LeadingOnes as for OneMax. Thus the proof follows in the same way as the proof of Theorem 8. □

Since the structural similarities between OneMax and LeadingOnes are so large, we make the same conjecture for LeadingOnes as we did for OneMax.

Conjecture 2. *For any $\lambda, \lambda' \in \mathbb{N}$ with $\lambda \leq \lambda'$, there exists an $t_0 \in \mathbb{N}$, such that for all $t \in \mathbb{N}$ with $t \geq t_0$*

$$Prob\left(T_{\lambda,\textsc{LeadingOnes}} < t\right) \geq Prob\left(T_{\lambda',\textsc{LeadingOnes}} < t\right)$$

holds.

3.3 Exact Analysis Summary

The exact analysis in this section nicely complements the asymptotic analysis in section 2 by showing that, regardless of the dimensionality n of simple landscapes such as OneMax and LeadingOnes, one cannot improve on the sequential optimization time of a (1+1) EA by increasing the offspring population size λ. Recall, however, that section 2 did exhibit an artifically constructed landscape for which $\lambda > 1$ resulted in significant improvements in optimization time. This leaves open the question as to whether there are "normally encountered" fitness landscapes for which $\lambda > 1$ is advantageous. We explore this possibility in the next section.

4 Empirical Analysis

The analyses of the previous two sections were performed with rigorous mathematical tools and involved very specific fitness landscapes. In this section we complement these mathematical analyses with a series of carefully chosen empirical studies designed to accomplish two goals: 1) to validate the mathematical results by comparing their predictions with the actual behavior of running algorithms, and 2) to see the extent to which the derived mathematical results apply to other fitness landscapes that are too complex for mathematical analysis.

4.1 Empirical Analysis Methodology

The design of our empirical analysis is motivated by the results and insights obtained from our theoretical analysis. First, since the dimensionality n of the fitness landscapes plays an important role in the theoretical analyses, for our empirical analyses we systematically varied n for each landscape over the set of values given by $n \in \{18, 24, 30, \ldots, 90\}$. A second important issue, of course, was the choice of appropriate values for λ. Clearly, we were interested in EA performance using small, dimension-independent values. For this aspect we chose $\lambda \in \{1, 2, 7\}$. In addition, the mathematical analyses from Section 2 suggest focusing on a number of dimension-dependent values for λ, specifically $\lambda = \Theta((\log n)(\log\log n)/\log\log\log n)$, $\lambda = \Theta(n)$, and $\lambda = \Theta(n \log n)$. However, the choice of the constant factors hidden in the Θ analyses is in some sense arbitrary. For the experiments reported here we used a constant factor of 10 throughout, i.e., $\lambda \in \{10 \cdot l(n) \cdot l(l(n))/l(l(l(n))), 10n, 10n \cdot l(n)\}$, with $l(n) = \lfloor \log_2 n \rfloor$. Combining this with our choices for n produced the additional λ-values in Table 1.

n	$10 \cdot l(n) \cdot l(l(n))/l(l(l(n)))$	$10n$	$10n \cdot l(n)$
18	80	180	720
24	80	240	960
30	80	300	1200
36	100	360	1800
42	100	420	2100
48	100	480	2400
54	100	540	2700
60	100	600	3000
66	120	660	3960
72	120	720	4320
78	120	780	4680
84	120	840	5040
90	120	900	5400

Table 1: Dimension-dependent values used for λ

We performed fifty independent runs for each combination of values for n and λ. We stopped each run after at most $10n^4$ function evaluations or when the global optimum was found. For those runs in which the optimum was located, we calculated the average number of function evaluations and the average number of generations needed to do so. Runs that failed to find the optimum were not

taken into account for the computation of the averages. Therefore, the number of runs which are averaged may vary and that number is reported for each experiment.

We began our empirical analysis of the (1+λ) EA by focusing on the three fitness landscapes used in the mathematical analyses, namely, OneMax, LeadingOnes, and SufSamp. The goal in the case was to see whether the empirical results support the theoretical results. We then selected three additional fitness functions that were not part of the theoretical analyses in order to get a sense of the predictive power of our results.

4.2 Empirical Analyses on LeadingOnes and OneMax

For both LeadingOnes and OneMax the global optimum was discovered in all runs without any exception, as one might expect from their simplicity. In Figures 3 and 4 we display the average number of function evaluations until the global optimum was discovered together with the confidence interval for each mean. For visual clarity we plot just a few representative values for λ.

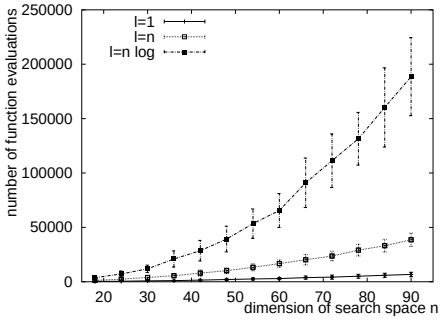

Figure 3: Average number of function evaluations required by various (1+λ) EAs on LeadingOnes as n increases.

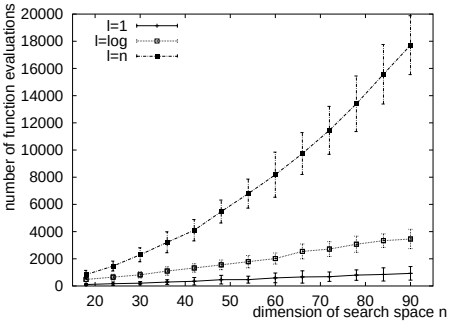

Figure 4: Average number of function evaluations required by various (1+λ) EAs on OneMax as n increases.

The results are exactly those predicted by the theoretical analyses, namely, that for these simple fitness landscapes, the performance of a (1+1) EA cannot be improved by increasing the value of λ. Moreover, performance systematically

degrades as λ increases. As long as λ is below the theoretical cutoff points ($O(n)$ for LeadingOnes and $O(\log n \cdot (\log\log n)/(\log\log\log n))$ for OneMax, the performance degradation is a constant factor c. However, as λ increases beyond these cutoff points, the performance degradation is polynomial in n.

4.3 Empirical Analyses on SufSamp

For the much more complex landscape SufSamp, the empirical results are quite different. Except for very small values of n, no algorithm is able to locate the global optimum in all 50 runs. The actual numbers of these successful runs are presented in Table 4 and require some additional discussion and interpretation.

n	$\lambda = 1$	$\lambda = 2$	$\lambda = 7$	$\lambda =$ "$\log n$"	$\lambda =$ "n"	$\lambda =$ "$n \log n$"
18	50	50	50	50	50	50
24	50	49	50	48	50	50
30	14	20	16	37	36	36
36	6	7	10	30	34	36
42	7	9	9	31	32	34
48	0	1	2	12	24	18
54	0	2	0	7	21	25
60	0	0	2	15	23	25
66	4	1	2	6	19	15
72	1	1	1	9	26	16
78	0	0	0	0	10	7
84	0	0	0	5	18	12
90	0	0	0	4	12	8

Table 2: Number of successful runs (out of 50) for different λ-values on SufSamp

First of all, note that for very small values of n ($n \leq 24$), the (1+λ) EA is successful almost regardless of the value of λ. This is mainly due to the fact that SufSamp does not really exhibit its intended character for such small dimensions of the search space. Our discussion of the properties of SufSamp was based on the fact that certain "large jumps" are very unlikely. This holds only for sufficiently large values of n.

For larger values of n and small constant values of λ ($\lambda \leq 7$), the empirical results match the theoretical predictions quite well in that the success rate drops drastically. As we increase λ as a function of n, the success rate improves until the final "$n \log n$" column. This is a bit surprising since the theoretical analysis of SufSamp indicated that $\lambda \geq \Theta(n \log n)$ was required for optimal performance. Since we observe this discrepancy only for the larger values of n, we believe this to be an artifact of the uniform upper limit of $10n^4$ function evaluations that we imposed on these empirical studies.

4.4 Empirical Analyses on Three Additional Fitness Functions

In order to get a sense as to whether the theoretical results of Sections 2 and 3 extend to other fitness landscapes, we performed our empirical analysis on three additional test functions.

4.4.1 De Jong's F1 Function

The first function is the well-known F1 function from the De Jong test suite (De Jong 1975). It is defined by $\text{F1}(x) = \sum_{i=1}^{3} x_i^2$ and is to be minimized. We use $-5.12 \leq x_i \leq 5.12$ for $i \in \{1, \ldots, 3\}$ and a binary representation. A bit string of length n is split into three components of equal length $n/3$. The i-th component represents x_i using standard binary representation. Therefore, $0^{n/3}$ represents -5.12, $1^{n/3}$ represents 5.12, $0^{(n/3)-1}1$ represents $-5.12 + 10.12/(2^{n/3} - 1)$ and so on.

F1 represents a discretized version of a simple continuous fitness landscape that presents no difficulties for any EA. As such, we would expect our results to closely match those of OneMax and LeadingOnes, and that is what we found. A global optimum was found in every run and the situation with respect to the average number of function evaluations (Figure 5) is very similar to Figures 3 and 4.

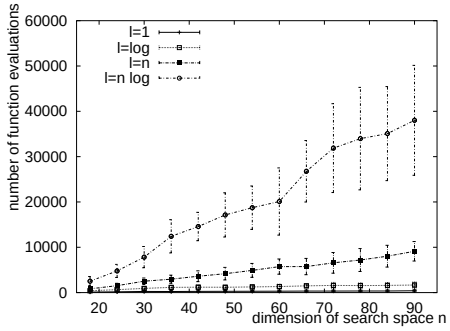

Figure 5: Average number of function evaluations required by various (1+λ) EAs on F1 as n increases.

4.4.2 Ackley's Concatenated Trap Function

The second function, Traps, is a concatenated 3-bit-trap function, defined by $\text{Traps}(x) = \sum_{i=1}^{n/3} f(x_{3(i-1)+1} x_{3(i-1)+2} x_{3i})$, where f is the well-known trap function (Ackley 1987), which is given as $f(0^k) = k + 1$ and $f(x) = \text{OneMax}(x)$ for $x \in \{0, 1\}^k$.

In this case the global optimum was also found in every run. However, unlike any of our earlier results, the particular value of λ had no significant affect on the performance of the (1+λ) EA (Figure 6). This can be understood by noting that the Traps function consists of a series of $n/3$ independent trap functions, thus producing a highly multi-modal fitness landscape. Improvements in performance on landscapes of this type are obtained by increasing the parent population size (μ) in order to support higher degrees of parallel search. Increasing λ when $\mu = 1$ does not increase this parallelism.

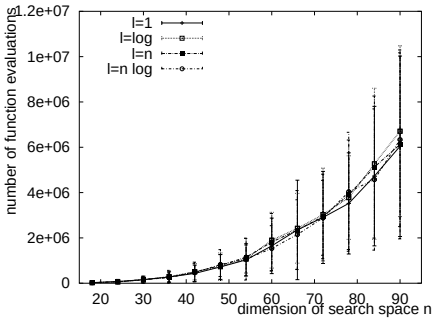

Figure 6: Average number of function evaluations for various (1+λ) EAs on TRAPS as n increases

4.4.3 A Matching Problem

For our third function, we selected a well-studied graph problem, namely, an instance of the maximal matching problem. In this particular formulation the graph is a line consisting of m nodes $\{1, \ldots, m\}$ and $m-1$ edges $\{\{1,2\},\{2,3\},\ldots,\{m-1,m\}\}$. A selection of the edges is represented by a bit string $x \in \{0,1\}^{m-1}$ in the following way. Each bit represents an edge and the edge is selected if the corresponding bit is set to 1. If the set of selected edges has the property that no two edges "collide" (i.e., share a common node), the edge set represents "a match". In that case, the fitness is defined to be the number of edges in the match set (i. e., ONEMAX(x)). Otherwise, the fitness is defined to be $-c$, where c is number of colliding edges. The objective is to find a maximal match. If m is even, there is a unique maximal match.

For MATCHING, the global optimum was not discovered in every run as indicated in Table 3. This is primarily due to the fact that the uniform upper bound on the number of function evaluations that we used for all experiments is of the same order of magnitude as the expected optimization time of $O(m^4)$ for a (1+1) EA (Giel and Wegener 2003). When the use of a larger offspring population size leads to wasting some function evaluations, this implies a decrease of the probability to find the global optimum. Accordingly, we observe a decreasing number of successful runs with increasing values of λ.

Figure 7 summarizes the effects of increasing λ on the number of function evaluations required to find the optimum. The results are consistent with the ONEMAX and LEADINGONES results, indicating the best performance is obtained by setting λ to small constant values.

4.5 Empirical Analysis Summary

The empirical studies in this section both complement and support the mathematical analyses of the earlier sections. Collectively, these analyses clarify our sense that the appropriate value for λ is landscape dependent, and is closely associated with the probability of producing an offspring that lies on a trajectory towards the global optimum. For simple landscapes, that probability is quite high. In such cases, setting λ to something larger than small constant values results in wasted computation effort. For complex landscapes in which that probability is quite low, much larger values of λ are required.

n	$\lambda = 1$	$\lambda = 2$	$\lambda = 7$	$\lambda =$ "$\log n$"	$\lambda =$ "n"	$\lambda =$ "$n \log n$"
18	50	50	50	50	44	38
24	50	50	50	50	47	29
30	50	50	50	50	37	28
36	50	50	50	50	38	21
42	50	50	50	49	37	21
48	50	50	50	48	35	16
54	50	50	50	49	30	16
60	50	50	50	50	36	12
66	50	50	50	49	26	13
72	50	50	50	47	23	13
78	50	50	50	49	27	10
84	50	50	50	50	22	9
90	50	50	50	50	21	6

Table 3: Number of successful runs on the MATCHING problem.

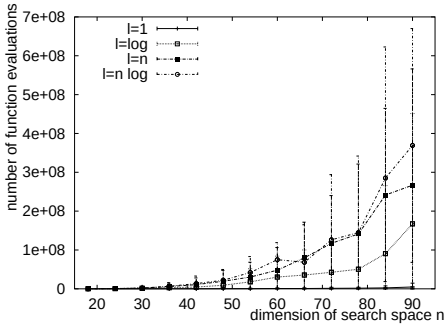

Figure 7: Average number of function evaluations for various (1+λ) EAs on the MATCHING problem with increasing n.

Unfortunately, the analyses presented here are not capable of making strong *a priori* predictions regarding the appropriate value of λ for an arbitrary landscape. For the practitioner, this means that some experimenting will be required to find effective λ-values for new landscapes. In addition, there is no reason to believe that holding the value of λ fixed during an evolutionary run is optimal. One possible improvement in this situation would be to have an EA self-adapt the value of λ. This possibility is explored in the next section.

5 An Adaptive (1+λ) EA

In this section we use the insights gained from the analyses presented in this paper to suggest an adaptive mechanism for having the offspring population size adjusted during a run by the EA itself. The idea for adapting the offspring population size is simple. We have seen in the proofs of Theorem 1 and Theorem 4 that we get maximal speed-up in a (1+λ) EA without significantly increasing the optimization load when we have the value of λ inversely proportional to the success probability, i.e., the probability of creating an offspring that replaces its parent. During an EA run, we can monitor this success rate and modify λ

accordingly. The particular algorithm we chose for doing so is as follows. If, in a particular generation, no offspring replaces its parent, we double the size of λ. If, in a particular generation, there are s offspring capable of replacing the parent, we reduce the value of λ to λ/s. Thus, we quickly increase the size of λ if it seems to be too small and reduce it if it appears to be too large based on each generation's estimate of the current success probability. For the sake of clarity we give a formal description of this adaptive (1+λ) EA:

Algorithm 2 (Adaptive (1+λ) EA).

1. **Initialization**
   `Choose` $x \in \{0,1\}^n \to \mathbb{R}$ `uniformly at random.`
   $\lambda := 1$
2. **Mutation**
   `For each` $i \in \{1,\ldots,\lambda\}$`:`
   `Create` $y_i \in \{0,1\}^n$ `by copying` x `and, independently for each bit,`
   `flip this bit with probability` $1/n$`.`
3. **Adaptation of λ**
   $s := |\{y_i \mid f(y_i) \geq f(x) \wedge y_i \neq x\}|$
4. **Selection**
   `If` $\max\{f(y_1),\ldots,f(y_\lambda)\} \geq f(x)$`, replace` x `by some randomly chosen` y_i `with maximal` f`-value.`
5. **Adaptation of λ**
   `If` $s > 0$ `Then` $\lambda := \lfloor \lambda/s \rfloor$ `Else` $\lambda := 2\lambda$`.`
6. **"Stopping Criterion"**
   `Continue at line 2.`

To study the effectiveness of this adaptive (1+λ) EA, we compared its performance with the non-adaptive (1+λ) EA on the six functions used in the empirical studies from Section 4. With one exception, the results were quite reasonable in the following sense. The number of function evaluations required to find the optimum were a bit higher, but comparable to that produced by the best choices for λ observed in the earlier empirical studies. Figure 8 illustrates this for the OneMax problem for which small static values of λ were shown to be best.

However, the way in which this adaptive (1+λ) EA achieves the comparable performance is strikingly different than an EA with a fixed λ-value. Figure 10 illustrates this by plotting how the adaptive EA dynamically changes the value of λ during a single run on OneMax. Recall that our earlier analyses suggested that, on landscapes like OneMax, the success probability is quite high early in an evolutionary run, but decreases significantly over time. Figure 10 suggests that our adaptive EA is tracking such changes quite nicely.

The one exception that we noted in the empirical studies regarding the performance of the adaptive (1+λ) EA was on the SufSamp function. Recall that it was carefully designed to require large values of λ in order to improve the likelihood that successful offspring would lie on a trajectory to the global

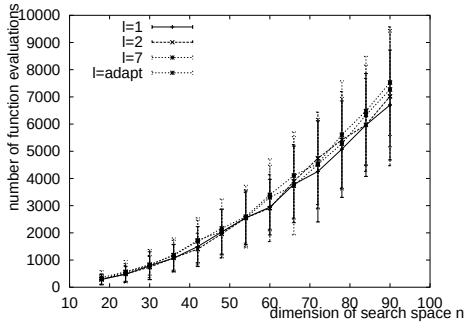

Figure 8: Average number of function evaluations for LeadingOnes with λ constant.

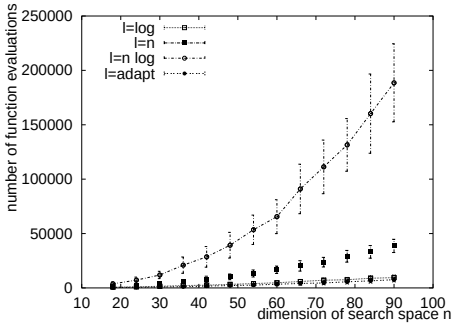

Figure 9: Average number of function evaluations for LeadingOnes with λ a function of n.

optimum. However, there is no way to monitor this success rate without *a priori* knowledge of the fitness landscape. As a consequence, the adaptive mechanism set λ to smaller suboptimal values, resulting in a much lower success rate for finding the optimum, as illustrated in Table 4.

6 Conclusions

Building on known results on the performance of the (1+1) EA, we have presented an analysis of the performance of the (1+λ) EA for different values of λ. One of the main reasons to introduce a non-trivial offspring population size is to significantly reduce the number of generations without significantly increasing the overall computational effort. When a parallel computing environment is available, this reduces the optimization time significantly. It is intuitively clear, that in this scenario it makes sense to increase the offspring population size λ roughly to the reciprocal of the success probability, i. e., the probability to produce an improving offspring. For simple landscapes like LeadingOnes and OneMax, these success probabilities are quite large. Hence, moderate static values for λ are shown to be optimal, and the penalty for modest increases in λ up to specified cutoffs points is only a modest decrease in performance. Beyond these cutoff points the performance penalties are much more severe.

By contrast, we have shown that there are more complicated landscapes for

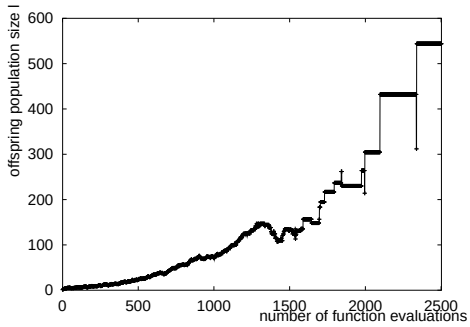

Figure 10: Values assigned to λ on OneMax during one run.

n	1	2	7	"log"	"n"	"$n \log n$"	adaptive
18	50	50	50	50	50	50	50
24	50	49	50	48	50	50	50
30	14	20	16	37	36	36	18
36	6	7	10	30	34	36	19
42	7	9	9	31	32	34	19
48	0	1	2	12	24	18	8
54	0	2	0	7	21	25	10
60	0	0	2	15	23	25	10
66	4	1	2	6	19	15	9
72	1	1	1	9	26	16	11
78	0	0	0	0	10	7	6
84	0	0	0	5	18	12	4
90	0	0	0	4	12	8	6

Table 4: Number of successful runs (out of 50) for different λ -values on SufSamp

which the necessary success probabilities require large offspring populations. We have illustrated such a situation by presenting a carefully constructed example function, SufSamp, where the offspring population size needs to be quite large in order to find a global optimum with high probability.

Our investigation has been carried out in three different ways. In Section 3, we have presented an exact analysis of the (1+λ) EA on OneMax and LeadingOnes. Since exact analyses give precise answers for any concrete value of n, they are obviously the most desirable. However, they are very difficult and can most often only be done in very simple cases. If an exact analysis is not possible, an asymptotic analysis can provide very valuable insights. We have presented such an analysis in Section 2. It yields rigorously proven results that hold for sufficiently large values of n and ignore the influence of constant factors and lower order terms. Therefore, it is not immediately clear whether these results give the right picture for small n, too. This is why an empirical study is a valuable addition. In Section 4, we have not only presented experiments supporting our analytical results. In addition, we have investigated some more fitness functions in order to provide further empirical indications on the validity of our general reasoning.

The insights from our analyses suggested that a simple adaptive scheme for the setting of the offspring population size λ might leave to a performance similar to that of a (1+λ) EA with an optimal value of λ without having to determine that optimal value *a priori*. We described such an adaptive (1+λ) EA in Section 5 and tested it on all of the test functions considered here. It turns out that this very simple adaptive variant performs well in those cases where a speed-up can be obtained by setting the offspring population size roughly to the reciprocal of the success probability. However, as could be predicted, it fails in the more complicated situation that a significantly larger value for λ is needed, namely on SufSamp.

We believe that we shed some light on questions that arise when good values for the choice of the offspring population size are needed. It is subject of future research to investigate whether the findings here are still valid when the evolutionary algorithm used is more complex.

References

D. H. Ackley (1987). *A Connectionist Machine for Genetic Hillclimbers*. Norwell, MA: Kluwer Academic Publishers.

T. Bäck (1996). *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press.

T. Cormen, C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to Algorithms, 2nd Edition*. New York, NY: McGraw Hill.

K. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph. D. thesis, University of Michigan.

S. Droste, T. Jansen, and I. Wegener (1998). On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), *Proceedings of the 5th Parallel Problem Solving from Nature* (*PPSN V*), 47–56. Springer. LNCS 1498.

S. Droste, T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science 276*, 51–81.

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

O. Giel and I. Wegener (2003). Evolutionary algorithms and the maximum matching problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science* (*STACS 2003*), 415–426. LNCS 2607.

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

T. Jansen and K. De Jong (2002). An analysis of the role of offspring population size in EAs. In W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honovar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO 2002*), 238–246. Morgan Kaufman.

T. Jansen and I. Wegener (2000). On the choice of the mutation probability for the (1+1) EA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton,

J. Merelo, and H.-P. Schwefel (Eds.), *Proceedings of the 6th Parallel Problem Solving From Nature* (*PPSN VI*), 89–98. Springer. LNCS 1917.

R. Motwani and P. Raghavan (1995). *Randomized Algorithms*. Cambridge University Press.

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