Money for Nothing: Speeding Up Evolutionary Algorithms Through Better Initialization

Axel de Perthuis de Laillevault
École Polytechnique
Univ. Paris-Saclay, France

Benjamin Doerr
École Polytechnique
Univ. Paris-Saclay, France

Carola Doerr
CNRS & Univ. Pierre et Marie Curie
Paris, France

ABSTRACT

That the initialization can have a significant impact on the performance of evolutionary algorithms (EAs) is a well known fact in the empirical evolutionary computation literature. Surprisingly, it has nevertheless received only little attention from the theoretical community.

We bridge this gap by providing a thorough runtime analysis for a simple *iterated random sampling* initialization. In the latter, instead of starting an EA with a random sample, it is started in the best of k search points that are taken from the search space uniformly at random. Implementing this strategy comes at almost no cost, neither in the actual coding work nor in terms of wall-clock time.

Taking the best of two random samples already decreases the $\Theta(n \log n)$ expected runtime of the (1+1) EA and Randomized Local Search on OneMax by an additive term of order $\sqrt{n}$. The optimal gain that one can achieve with iterated random sampling is an additive term of order $\sqrt{n \log n}$. This also determines the best possible mutation-based EA for OneMax, a question left open in [Sudholt, IEEE TEC 2013].

At the heart of our analysis is a very precise bound for the maximum of k independent Binomially distributed variables with success probability 1/2.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

Keywords

Theory; Runtime Analysis; Random Restarts; Initialization

1. INTRODUCTION

A widely applied initialization technique in evolutionary computation is *random initialization*. An evolutionary algorithm (EA) using this method samples its first search point uniformly at random from the search space. Clearly, this method is useful only if no a priori or just little knowledge about the fitness landscape is available. Otherwise, one would start the search close to the known areas of attraction.

For many practical applications it has proven beneficial to generate not only one random search point at the beginning but to sample instead some k random search points and to take the best one of these as starting point of the main optimization routine. The simplest such approach takes the random samples independently and uniformly at random from the search space, an approach that we call *iterated random sampling* in the following. Iterated random sampling is a very easy to implement initialization that can reduce the runtime of an EA significantly.

Interestingly, for discrete search spaces the initialization phase of EAs has not been thoroughly looked at from a theoretical perspective. With this work we contribute such a mathematical investigation for iterated random sampling. More precisely, we analyze for the well-known OneMax function the expected number of function evaluations that one can save by equipping the random hill climber Randomized Local Search (RLS) or the (1+1) EA with this initialization technique. Our results show that one can gain an additive number of $\Theta(\sqrt{n \log n})$ function evaluations by investing $k = \sqrt{n}/\log n$ initial samples.

We recall that OneMax is defined by $\textsc{Om}(x) := \sum_{i=1}^{n} x_i$, $x \in \{0,1\}^n$, thus counting the number of ones in the bit string x. This function is easily optimized by the all-ones string. However, the optimization behavior of RLS is identical on all strictly monotone functions, our result for OneMax naturally extends to that class of functions, which has a rather complex structure that is not well understood to date (e.g., for the (1+1) EA, an algorithm which is very well studied for a large range of problems, its expected runtime on the class of strictly monotone functions is not known [5]). However, for the (1+1) EA, too, the result naturally extends to a more complex class of functions. Indeed, the (1+1) EA does not distinguish between the so-called generalized OneMax functions $\textsc{Om}_z(\cdot) : x \mapsto |\{i \in \{1,\dots,n\} \mid x_i = z_i\}|$ (cf. Section 2 for a brief discussion of these functions) and the result for OneMax therefore applies to any function from that class.

From a technical perspective, we believe that our result is interesting for the evolutionary computation community for the following three reasons.

1. Our result determines the best possible mutation-based EA for OneMax, a question that has been repeatedly ad-

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ISBN 978-1-4503-3472-3/15/07…$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754760

dressed in the runtime analysis literature [3, 12, 14]. Sudholt [12, Theorem 12] proves that the best such EA is a version of the $(1+1)$ EA with standard bit mutation $1/n$ and k iterated random samples, for some k that he does not determine further than noting that it is of order at most $\sqrt{n}\log n$. Our analysis shows that the optimal k is between $\sqrt{n}/\log n$ and $\sqrt{n/\log n}$.

2. Our bound for RLS with iterated random sampling improves the best known upper bound for the unary unbiased black-box complexity of OneMax, a problem that is studied in the black-box complexity literature, which itself has recently attracted substantial attention in the theory of evolutionary computation. While we know by a result of Lehre and Witt [10] that this complexity is of order $\Theta(n\log n)$, we do not know to date the best unary unbiased algorithm for this function. Based on the results from this paper, we strongly suspect that this algorithm is a variant of RLS with iterated random sampling and fitness-dependent mutation steps, but proving that this is indeed the best one can do remains an open problem. Still our result reduces the size of the interval in which this unary unbiased black-box complexity of OneMax is located. For the reader who is unfamiliar with black-box complexity we briefly recall that, intuitively, the unary unbiased black-box complexity of a function f is the expected number of function evaluations that a best possible purely mutation-based black-box algorithm can exhibit on f. This algorithm does not necessarily have to be "evolutionary". We refer the interested reader to [10] for a more extensive description.

3. We give a very precise estimate for the maximum of k independently sampled binomially distributed variables with success probability $1/2$ (Theorem 3). We believe that the techniques used here, albeit not new (but not or at least not intensively used in the runtime analysis community) are very useful for a precise runtime analysis of other functions, a problem that is gaining importance in the recent literature [4, 7, 9, 12, 14].

In his work [12] Sudholt claims that "As a word of caution, we remark that it is clearly not worth optimizing for k in practice as the differences in the expected running time only concern additive terms of small orders." Naturally, we disagree with this view. In practical applications, saving runtime can be very important, even if it concerns only lower order terms. In particular when this comes at almost no extra cost there is no reason not to use iterated random sampling. Note that for optimizing OneMax with RLS or the $(1+1)$ EA sampling just two random initial search points instead of one saves already $\Theta(\sqrt{n})$ function evaluations in expectation at the negligible cost of sampling and evaluating one more random search point.

We recall that our main result shows that we can reduce the expected runtime of RLS and the $(1+1)$ EA by starting them in the best of k independently sampled search points. One may wonder if we cannot do even better if we smartly design the initial search points instead of just sampling them independently and uniformly at random. We address this question in Section 6. Interestingly, we can show that this is not the case. More precisely, regardless of how we choose the $k=\omega(1)$ initial search points, the best one of them has OneMax value at most $n/2+O(\sqrt{n\log k})$ with high probability. Here, of course, we regard the OneMax value with respect to an arbitrary OneMax function OM_z and not with respect to the classic OneMax function $\mathrm{OM}_{(1,\dots,1)}$ (which is a trivial problem anyway). The claimed bound is much more precise in the actual statement (cf. Theorem 8) and shows that for $k=\omega(1)\cap o(\sqrt{n})$ random initial search points are essentially best possible.

While the technical details of our analysis as well as the statements of the main results (Theorems 1 and 2) are rather tedious, we illustrate the expected gain from iterated random sampling in a series of figures, cf. Section 4. We also provide in that section, for n varying between 2^5 and 2^{20}, the optimal values k^* of random initial samples and the resulting expected gain in terms of function evaluations.

2. ALGORITHMS WITH ITERATED RANDOM SAMPLING

We regard in this work RLS (Algorithm 1) and the $(1+1)$ EA (Algorithm 2) with iterated random sampling. Our presentation focuses on the maximization of pseudo-Boolean functions $f:\{0,1\}^n\to\mathbb{R}$, though the concept of iterated random resampling is easily extensible to other search spaces.

The initialization phase **BestOf**(k) (cf. Algorithm 3) for both algorithms is identical. In the version with k random initial samples, k bit strings are taken from the search space independently and uniformly at random. The best one of these (ties broken uniformly at random) serves as the parent for the next generation. Both algorithms maintain a population size of one always. While RLS creates in each iteration a new search point y from the parent x by flipping in x a random bit, the $(1+1)$ EA applies to x so-called standard bit mutation, i.e., it creates y by flipping each bit in x independently with probability $1/n$. In the selection step of both algorithms the parent x is replaced by its offspring y if and only if $f(y)\geq f(x)$.

For convenience we abbreviate in the following by RLS_k ($(1+1)$ EA_k) the RLS ($(1+1)$ EA) variant with a **BestOf**(k) initialization.

As a side note we remark that these algorithms differ from a $(k+1)$ EA in that they can only store one search point at a time (i.e., they follow a (1+1) scheme). This, together with the typically employed uniform or fitness-proportional selection for reproduction, can make a crucial difference. Indeed, the performance of a $(k+1)$ EA can substantially differ from that of a $(1+1)$ EA_k, see [13, Section 6] for an example function that the $(1+1)$ EA fails to optimize efficiently but that, for large enough k, is efficiently optimized by a $(k+1)$ EA.

We use the standard notation $[n]:=\{i\mid i\in\mathbb{N}, i\leq n\}$. All logarithms in this work are natural logarithms, i.e., logarithms to base $e:=\exp(1)$.

Algorithm 1: Randomized Local Search with iterated random sampling for maximizing $f\colon\{0,1\}^n\to\mathbb{R}$.

```
Initialization:
x ← BestOf(k);
Optimization:
for t = 1, 2, 3, ... do
    Choose j ∈ [n] uniformly at random;
    Set y ← x ⊕ e_j^n ; //mutation step
    if f(y) ≥ f(x) then x ← y ; //selection step
```

Algorithm 2: The (1+1) EA with iterated random sampling for maximizing $f\colon \{0,1\}^n \to \mathbb{R}$.

$x \leftarrow$ **BestOf**(k);
Optimization: for $t = 1, 2, 3, \ldots$ **do**
  Sample $y \in \{0,1\}^n$ by flipping each bit in x with probability $1/n$ and query $f(y)$; //mutation step
  if $f(y) \geq f(x)$ **then** $x \leftarrow y$; //selection step

Algorithm 3: The **BestOf**(k) initialization operator.

for $i = 1, 2, \ldots, k$ **do** Sample $x^{(i)} \in \{0,1\}^n$ uniformly at random;
Select $j \in [k]$ with $f(x^{(j)}) = \max\{f(x^{(i)}) \mid i \in [k]\}$ uniformly at random;
$x \leftarrow x^{(j)}$;

2.1 Known Runtime Bounds

As commented in the introduction RLS behaves identical on all strictly monotonic functions, i.e., on all functions f for which $f(y) > f(x)$ whenever $y_i \geq x_i$ for all $i \in [n]$ and $y_j > x_j$ for at least one index j. This is due to the fact that in every iteration RLS flips exactly one bit. It is well known that the runtime of RLS on any such strictly monotonic function is $\Theta(n \log n)$. This follows from the coupon collector's problem, a classic problem in the analysis of algorithms community, cf. [1]. Tight bounds for the runtime of RLS on such functions (up to a $\pm o(1)$ term) have recently been proven in [7]. More precisely, it is shown there that the runtime is $nH_{n/2} - 1/2 \pm o(1)$, where $H_{n/2}$ denotes the $(n/2)$th harmonic number when n is even, and $H_{n/2} := (1/2)H_{\lfloor n/2 \rfloor} + (1/2)H_{\lceil n/2 \rceil}$ when n is odd.

Note that the $(1+1)$ EA may flip more than one bit per iteration. Its optimization behavior therefore crucially depends on the exact combination of bits flipped (and, typically, on the state of all other bits as well). Results for OneMax do therefore not extend to the class of monotonic functions. In fact, it is not even known to date whether or not the asymptotic runtimes are of the same order, see [5] for a discussion. Multiple bit flips make analyzing the runtime of the $(1+1)$ EA much more difficult than analyzing local search algorithms such as RLS.

Still, results that hold for the (1+1) EA on OneMax hold also for every generalized OneMax function

$$\mathrm{OM}_z : \{0,1\}^n \to \{0,1,\ldots,n\}, x \mapsto \sum_{i=1}^{n} 1(x_i = z_i), \qquad (1)$$

where $z \in \{0,1\}^n$ is the *target string* of OM_z and the binary indicator function $1(x_i = z_i)$ is one if and only if x_i equals z_i. Function OM_z therefore counts the number of positions in which x and z agree. The traditional OneMax function counting the number of ones is $\mathrm{OM}_{(1,\ldots,1)}$. It is easy to see that for all z the fitness landscape of OM_z is isomorphic to that of $\mathrm{OM}_{(1,\ldots,1)}$. This also explains that the optimization behavior of the (1+1) EA is identical on all generalized OneMax functions. Furthermore, the unique global optimum of OM_z is z, hence the name *target string*. For simplicity, we regard in the following mainly the classic OneMax function counting the ones in a bit string. All results extend to any other OM_z function in a trivial way.

For the $(1+1)$ EA, too, it is well known that its expected runtime on OneMax is of order $n \log n$. This result has first been proven in [11]. The known upper and lower bounds have been improved in a sequence of papers. The currently best known, and actually very tight, bound can be found in the preprint [9]. It states that the expected runtime of the $(1+1)$ EA on OneMax is $en \log n + c_1 n + \frac{1}{2} e \log(n) + c_2 + O(\log(n)/n)$ for some explicitly computable constants c_1 and c_2.

While the above mentioned bounds hold for the RLS and the $(1+1)$ EA without iterated random sampling, not much can be found in the evolutionary computation literature about the versions that we are interested in. Only Sudholt [12, Section 7] observes that the performance of the $(1+1)$ EA can be improved through iterated random sampling. He shows that for $k = \Theta(\log n)$ the $(1+1)$ EA with a **BestOf**(k) initialization is faster than the standard $(1+1)$ EA by an additive term of order $\sqrt{n}$. He also observes that the best trade-off between random initial samples and saved runtime bounds is achieved for some $1 < k = O(\sqrt{n} \log n)$.

3. OVERVIEW OVER MAIN RESULT

We aim at analyzing the trade-off between the number k of iterated random samples that **BestOf**(k) evaluates and the expected gain that this gives for the running times of RLS and the $(1+1)$ EA, respectively. Let $\Delta_{k,\mathrm{RLS}}$ ($\Delta_{k,(1+1)\ \mathrm{EA}}$) be total expected time that we save if we start RLS (the (1+1) EA) with the **BestOf**(k) search point. Then the **BestOf**(k) initialization saves in expectation $\Delta_{k,\mathrm{RLS}}$ ($\Delta_{k,(1+1)\ \mathrm{EA}}$) iterations at the cost of $k-1$ additional random samples. We therefore aim at maximizing the quantities $\Delta_{k,\mathrm{RLS}} - k + 1$ and $\Delta_{k,(1+1)\ \mathrm{EA}} - k + 1$, respectively. We state below the theorems for both algorithms. For reasons of space, however, we cannot describe the full proof details for both RLS and the $(1+1)$ EA. We therefore provide in this extended abstract the proof for the expected savings on RLS (see Section 5.2), while we defer the analysis for the $(1+1)$ EA to the journal version. We will remark, however, where and how the analysis for the $(1+1)$ EA differs from that of RLS. Furthermore, we would like to note that the statement below is rather technical. We will explain all the details in the subsequent sections, and perhaps more importantly for those less interested in the mathematical details, we will illustrate our findings with the help of several numeric examples in Section 4.

THEOREM 1. *Let $k = O(\sqrt{n})$ and let $\phi : \mathbb{R} \to \mathbb{R}, x \mapsto \frac{1}{\sqrt{\pi}} \int_{-\infty}^{x} e^{-t^2} \mathrm{d}t$ denote the cumulative distribution function (CDF) of the Normal distribution with mean zero and variance 1/2. Then*

$$\Delta_{k,\mathrm{RLS}} = 2\sqrt{\frac{n}{2}} \int_{-\infty}^{+\infty} \phi(t) \left(1 - \phi(t)^{k-1}\right) \mathrm{d}t + O\left(k\sqrt{\log n}\right).$$

In particular, $\Delta_{k,\mathrm{RLS}}$ equals

$$2\sqrt{\frac{n}{2}\left(\log k - \frac{1}{2}\log\log k\right)} + O\left(k\sqrt{\log n} + \sqrt{\frac{n}{\log k}}\right).$$

Numerical approximations of the term $\int_{-\infty}^{+\infty} \phi(t) \left(1 - \phi(t)^{k-1}\right) \mathrm{d}t$ for $k = 2^i$, $i = 1, \ldots, 33$,

are provided in Table 1. The claimed approximation of this integral can be found in Theorem 7.

The second statement of the theorem is in general less accurate than the first, but much easier to compute. For $k \in \Omega\left(\frac{\sqrt{n}}{\log n}\right)$, both statements are equally accurate. The term $2\sqrt{\frac{n}{2}}$ was not simplified to respect the analogy with Theorem 2.

THEOREM 2. *For k and ϕ as in Theorem 1*

$$\Delta_{k,(1+1)\text{ EA}} = \eta\sqrt{\frac{n}{2}} \int_{-\infty}^{+\infty} \phi(t)\left(1-\phi(t)^{k-1}\right) \mathrm{d}t + O\left(k\sqrt{\log n}\right),$$

where $\eta \approx 2.969$ is an absolute constant.

In particular, $\Delta_{k,(1+1)\text{ EA}}$ equals

$$\eta\sqrt{\frac{n}{2}\left(\log k - \frac{1}{2}\log\log k\right)} + O\left(k\sqrt{\log n} + \sqrt{\frac{n}{\log k}}\right).$$

From Theorems 1 and 2 we see that the respective bounds on Δ_k differ only in the leading constant. In fact, this constant corresponds to the time that it takes on average to gain one fitness level, provided that the algorithms start in a search point of fitness $n/2$. That this value is two for RLS is easily seen, since the probability that a one-bit flip changes a zero into a one equals $1/2$ when the number of ones equals the number of zeros. For the $(1+1)$ EA, one can show that the expected progress of one iteration starting in a search point of ONEMAX-value $n/2$ is approximately $0.337 = 1/2.969$.

3.1 Discussion of the Main Result For RLS

We will show in Theorem 3 that the expected fitness value of the search point returned by **BestOf**(k) is by an additive term of $\sqrt{\frac{n}{2}\left(\log k - \frac{1}{2}\log\log k + O(1)\right)}$ better than the expected fitness $n/2$ of a random search point. Intuitively, it is logical that the gain $\Delta_{k,\mathrm{RLS}}$ should be the product of this expected fitness gain and the expected cost of going from one fitness level to the next. This is what the second statement in Theorem 1 shows.

From this second expression in Theorem 1 one can see that choosing $k = \sqrt{n}/\log n$ guarantees that $\Delta_k - k + 1$ is

$$\begin{aligned}&\tfrac{c}{2}\sqrt{n(\log n - 3\log\log n)} + O(\sqrt{n/\log n})\\ &= \tfrac{c}{2}\sqrt{n\log n} + O\left(\frac{\sqrt{n}\log\log n}{\sqrt{\log n}}\right).\end{aligned}$$

Unfortunately, we see that the error term in Theorem 1 is quite large. Indeed, it is larger than the cost of sampling k random bitstrings. This makes it difficult to estimate precisely the optimal value of k, i.e., the value which maximizes $\Delta_{k,\mathrm{RLS}} - k + 1$. Still it is interesting to compute the optimal value of k assuming that there is no error term, i.e. assuming that $\Delta_k = c\sqrt{\frac{n}{2}\left(\log k - \frac{1}{2}\log\log k + \lambda\right)}$ exactly for some constant λ.[1] Using this equality we can differentiate

[1]Here again we replace the 2 preceding the $\sqrt{\frac{n}{2}\left(\log k - \frac{1}{2}\log\log k + O(1)\right)}$ term by c to get a result that is applicable also to other search heuristics, in particular the $(1+1)$ EA.

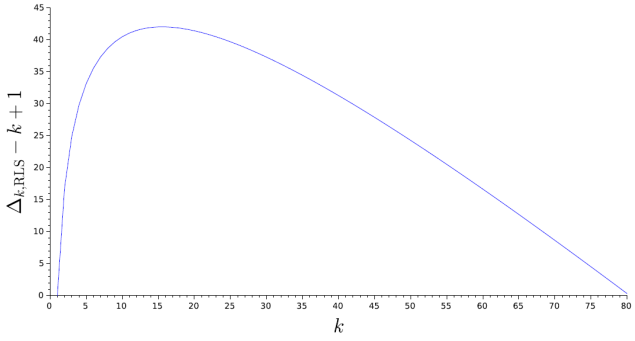

Figure 1: The expected gain of RLS_k over RLS_1 (in number of function evaluations) for $n = 1000$ and different values of k.

$\Delta_k - k$ to find the optimal value of k, which we denote by k^*. This yields

$$k^* = \frac{c}{2}\sqrt{\frac{n}{\log n}}\left(1 + \frac{\log\log n + O(1)}{\log n}\right), \tag{2}$$

where the $O(1)$ term is a function of λ and c. For $c = 2$ (i.e., in particular, for RLS), this $O(1)$ term is experimentally between -0.3 and $+0.3$. For this k^* we get an expected gain in runtime of size

$$\Delta_{k^*} - k^* + 1 = \frac{c}{2}\sqrt{n\log n} + O(\sqrt{n}). \tag{3}$$

See Figures 2 and 3 for Graphs illustrating the accuracy of this estimation.

4. NUMERIC EXAMPLES

While for the runtime analysis community the analysis of Theorems 1 and 2 may be of independent interest, the practitioner certainly cares more about the absolute gains that can be achieved with iterated random sampling. We therefore provide in this section a few numeric examples helping us understand the rather complex terms of the main theorems.

We start with Figure 1 which plots for different values of k the expected runtime gain of the RLS_k on a strictly monotonic function of length $n = 1,000$. The graph nicely illustrates that the diminishing rate of return. More precisely, while for $k = 10$ we gain an expected number of approximately 40 function evaluations, this expected gain is only approximately 24 for $k = 50$. For $k > 80$ the expected gain is negative, that is, each additional random sample saves in expectation less than one fitness value.

We see from Figure 1 that for $n = 1,000$ the optimal number of random initial samples is around 16. After that number, the expected gain decreases. We use Table 2 to displays the optimal number $k*$ of initial random samples for RLS on ONEMAX with varying solution length. Table 2 also lists the resulting gains in the expected runtime of the RLS_k over the standard RLS. Figures 2 and 3 show that the estimations (2) and (3) from Section 3.1 are quite accurate.

Finally, we present one typical run of the **BestOf**(k) routine for $n = 100$. That is, we plot in Figure 4 the outcome of 100 independently sampled binomially distributed variables with success probability $1/2$ (these are the dots in

k	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}
	0.3967507	0.7276378	1.0066125	1.2487419	1.4634766	1.6572698	1.8346574	1.9988942	2.1523646	2.2968522	2.4337195
k	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}
	2.5640273	2.6886174	2.8081689	2.9232380	3.0342866	3.1417030	3.2458170	3.3469117	3.4452321	3.5409920	3.6343791
k	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}
	3.7255593	3.8146802	3.9018738	3.9872584	4.0709409	4.1530181	4.2335780	4.3127008	4.3904599	4.4669230	4.5421517

Table 1: Numerical approximations of $\int_{-\infty}^{+\infty} \phi(t) \left(1-\phi(t)^{k-1}\right) \mathrm{d}t$ for k varying from 2^1 to 2^{33}.

n	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
$k*$	4	5	7	9	12	16	21	28	38	51	69	93	127	173	237	325
$\Delta_{k^*} - k^* + 1$	3.0	5.7	9.9	16.5	26.9	42.7	66.9	104	159	241	364	546	815	1212	1796	2653

Table 2: Optimal value of $\Delta_k - k + 1$ for n varying from 2^5 to 2^{20}.

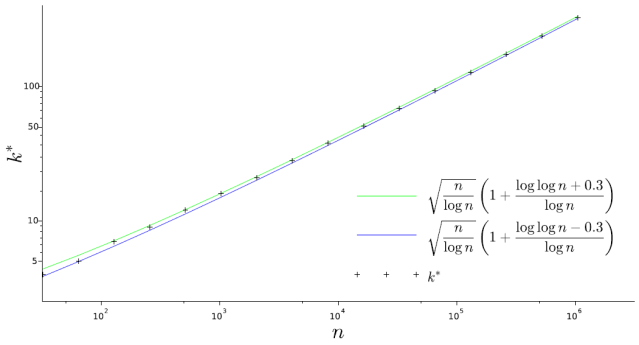

Figure 2: Comparison of k^* with the bounds from our estimation (2) for different sizes of n.

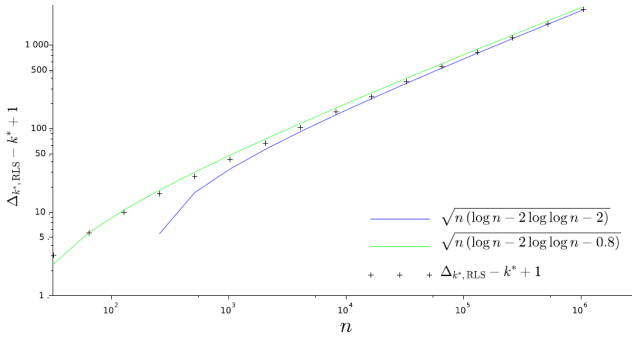

Figure 3: Comparison of $\Delta_{k^*,\mathrm{RLS}} - k^* + 1$ with the bounds from estimation (3) for different sizes of n. The upper line plots $\sqrt{n(\log n - 2\log\log n - 0.8)}$, while the lower line plots $\sqrt{n(\log n - 2\log\log n - 2)}$.

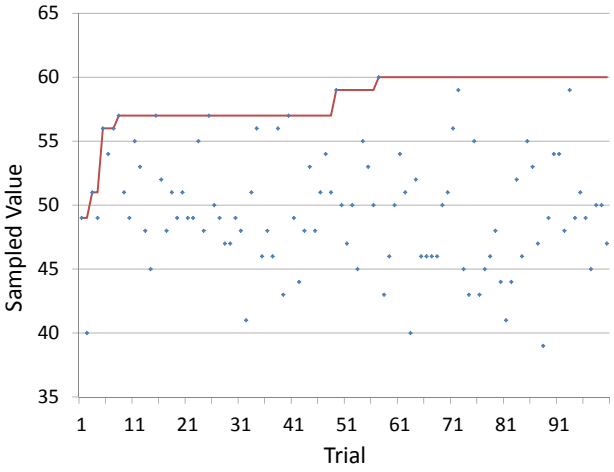

Figure 4: The dots represent the values of 100 independent Binomially distributed variables with 100 trials and success probability 1/2 each. The solid line at value k displays the maximum of the first k values.

Figure 4). The line represents the maximum of the sampled values, and thus the OneMax-value of the individual that the **BestOf**(k) initialization would return if it had sampled search points according to the plotted points in that graph. E.g., for $k = 20$ it would have returned a value of 57. This figure illustrates that in the beginning we typically gain quite a bit from resampling while we have to wait the longer the more samples we already have.

5. PROOF OF THEOREM 1

As mentioned above we present the proof of Theorem 1 and point out differences to the proof of Theorem 2. As a preliminary step, we analyze the maximum of k independent binomially distributed variables.

5.1 Bounding the Starting Fitness

Intuitively, to bound the runtime of RLS_k on OneMax we need to know the fitness value of the starting point re-

turned by the **BestOf**(k) initialization routine. Since the k samples of the **BestOf**(k) routine are sampled independently, and since the OneMax fitness value of a random search point follows a binomial distribution with n trials (corresponding to the n positions) and success probability $1/2$ (corresponding to the fact that in a uniform search point each bit position is one with probability exactly $1/2$), analyzing the fitness of the starting point corresponds to analyzing the maximum of k independent binomially distributed variables. Our analysis presented below hence addresses also a classic problem in the theory of computation literature.

We provide in this section an analysis of the expected fitness value of the starting search point. This will give us insights needed in the later runtime analysis of RLS_k. However, as a word of caution, we recall that the expected runtime $\text{E}[T(\text{RLS}_k)]$ of RLS_k on OneMax is

$$\sum_{i=0}^{n} \text{E}[T(\text{RLS}_k) \mid f_k = i] \Pr[f_k = i], \tag{4}$$

where f_k denotes the fitness of the starting point returned by the **BestOf**(k) routine. Hence, to bound this runtime, we need not only analyze the *expected* value f_k of the starting point, but also its distribution. Still we will gain from this section a first intuition of how to analyze the distribution f_k and we can also describe the main intuition for the proof of Theorem 1. Furthermore, the rate with which RLS progresses in the range where most random samples are found is more or less constant. Therefore if we manage to calculate the difference between the fitness of a random sample, f_1 and that of the best of k samples, f_k, the gain obtained by starting in the output of the **BestOf**(k) is approximately $c(f_k - f_1)$ for some suitably chosen constant c.

The main result of this section is the following.

Theorem 3. *Let $n \in \mathbb{N}$, $k \in \omega(1) \cap o(\sqrt{n})$. Let $X_1, \dots, X_k$ be independently sampled from the Binomial distribution $\text{Bin}(n, 1/2)$ with n trials and success probability $1/2$. Let X be the maximum of $X_1, \dots, X_k$.*

With high probability (i.e., with probability $1 - o(1)$), X is between

$$i_- = \frac{n-1}{2} + \sqrt{\frac{n}{2}}\sqrt{\log k - \frac{1}{2}\log\log k - c_k}$$

and

$$i_+ = \frac{n-1}{2} + \sqrt{\frac{n}{2}}\sqrt{\log k - \frac{1}{2}\log\log k + c_k},$$

where c_k is an arbitrary sequence that tends to infinity.

To prove Theorem 3 we make use of the well-known approximation of Binomially distributed variables by the Normal distribution, which is often called *Gaussian* distribution. The Berry-Essen theorem bounds the difference of these two distributions. We recall from Theorem 1 that ϕ denotes the CDF of the Normal distribution with mean zero and variance $1/2$. An important theorem bounding the error of this approximation is the following theorem, which was proven independently by Berry [2] and Esseen [8] in 1941 and 1942, respectively. Note here that a binomially distributed variable $X = \text{Bin}(n, p)$ can be decomposed into the sum of n independent Bernoulli variables $\text{Ber}(p)$.

Theorem 4. *[2, 8] Let $n \in \mathbb{N}$. Let $X_1, \dots, X_n$ be n independently identically distributed (i.i.d.) random variables of mean $E(X_1) = \mu$, variance $E\left(|X_1 - \mu|^2\right) = \sigma^2$ and third centralized moment $E\left(|X_1 - \mu|^3\right) = \rho^3$.*

Let

$$Y_n = \frac{1}{\sigma\sqrt{2n}}\left(X_1 + \dots + X_n - n\mu\right)$$

and let F_n be the cumulative distribution function (CDF) of Y_n.

Then there exists a constant C such that for all $x \in \mathbb{R}$

$$|F_n(x) - \phi(x)| \leq \frac{C\rho^3}{\sigma^3\sqrt{n}}.$$

The constant C has been refined over the years. For the sum of Bernoulli variables of parameter $1/2$—this is the case we are interested in in our applications—it is known that $\frac{C\rho^3}{\sigma^3} \leq 1$.

We recall that the cumulative distribution function of the Binomial distribution $\text{Bin}(n, p)$ is given by $F(i) = \sum_{t=0}^{\lfloor i \rfloor} \binom{n}{t}/2^n$. Clearly we have $\Pr[\max(X_1, \dots, X_k) \leq i] = \Pr[X_1, \dots, X_k \leq i] = \Pr[X_1 \leq i]^k = F(i)^k$. Though this is a nice-looking formula, it is not straightforward to bound the individual terms precisely. In the proof of Theorem 3 we shall estimate these expressions using Theorem 4 for values i that are close to the mean, and using Theorem 5 below for values that are far from the mean value. Here, by "close" and "far" we mean within, respectively outside, of an $\sqrt{n \log n}$ interval around the mean $n/2$. As for the Berry-Esseen theorem, the important fact is that it allows us to write

$$F(i) = \left(\phi(x_i) + \varepsilon(x_i)\right)^k,$$

for some function ε satisfying $\sup_{x \in \mathbb{R}} |\varepsilon(x)| = O(1/\sqrt{n})$ and $x_i := \sqrt{\frac{2}{n}}\left(i + \frac{1}{2} - \frac{n}{2}\right)$. This is the approximation that we use in the proofs of Theorems 3, 1, and 2.

Theorem 5. *Let $n \in \mathbb{N}, k \in \mathbb{N}$. Let F be the CDF of $\text{Bin}(n, 1/2)$, the Binomial distribution with n trials and success probability $1/2$. Then,*

$$F(i) \leq \exp\left(-\frac{2}{n}\left(i - \frac{n}{2}\right)^2\right),$$

and

$$1 - F(i)^k \leq k \exp\left(-\frac{2}{n}\left(i - \frac{n}{2}\right)^2\right).$$

Furthermore, for $\alpha > 0$, $n_+ := \frac{n}{2} + \alpha\sqrt{\frac{n}{2}}$, and $n_- := \frac{n}{2} - \alpha\sqrt{\frac{n}{2}}$, it holds that

$$0 \leq \sum_{i=n_+ + 1}^{n-1} 1 - F(i)^k \leq k\sqrt{\frac{n}{2}}\frac{e^{-\alpha^2}}{2\alpha}$$

and

$$0 \leq \sum_{i=0}^{n_- - 1} F(i) \leq \sqrt{\frac{n}{2}}\frac{e^{-\alpha^2}}{2\alpha}.$$

This theorem can be proven by Hoeffding's inequality, which is sometimes called the additive version of Chernoff's bound, cf. [6].

Using the two approximations from Theorems 4 and 5 one can show Theorem 3. However, the proof remains quite

technical, for which reason we do not give the full proof details here.

We remark that experimental results suggest that the bounds provided by Theorem 3 are, unfortunately, not as accurate as they could be. This possible inaccuracy causes us to choose $k = O(\sqrt{n})$ throughout this paper, though we strongly suspect that the statements in Theorem 1 and 2 hold for much more general (that is, larger) values of k as well.

5.2 Proof Details

To prove Theorem 1, we need to bound the terms in equation (4). As indicated above, this will be done by approximating the cumulative distribution function of the Binomial distribution by a Normal distribution. Useful tools in this analysis are Theorems 6 and 7 below.

Theorem 6 gives an approximation of the weighted sum $\sum_{i=0}^{n-1} F(i)\left(1-F(i)^{k-1}\right)c_i$. In our applications of this theorem in the proofs of Theorem 1 and 2 the weights c_i correspond to the time that it takes to advance one fitness level, provided that you start in a search point of fitness i, while $F(i)\left(1-F(i)^{k-1}\right)$ measures the difference of the probabilities that level i is visited, where the difference is with respect to the standard RLS vs. RLS_k with a **BestOf**(k) initialization.

THEOREM 6. *Let $n \in \mathbb{N}$, $k \in O(\sqrt{n})$, and let F be the CDF of* $\text{Bin}(n,1/2)$. *Furthermore, let $(c_i)_{i=0}^{n-1}$ be a sequence of positive numbers satisfying the following conditions. (i) There exist positive constants p and K such that, for all $i \in [n]$, $c_i \le Kn^p$ and (ii) there exists a constant c such that, for $i = n/2 \pm \sqrt{\frac{p+1}{2} n \log n}$, $c_i = c\left(1+O(\sqrt{\log(n)/n})\right)$.*

Then

$$\sum_{i=0}^{n-1} F(i)\left(1-F(i)^{k-1}\right)c_i = c\sqrt{\frac{n}{2}}\int_{-\infty}^{+\infty}\phi(t)\left(1-\phi(t)^{k-1}\right)\mathrm{d}t + O\left(k\sqrt{\log n}\right).$$

The next theorem gives an evaluation of the integral of the first statement in Theorem 1.

THEOREM 7. *For $k \in \mathbb{N}$ the integral $\int_{-\infty}^{+\infty}\phi(t)\left(1-\phi(t)^{k-1}\right)\mathrm{d}t$ equals*

$$\sqrt{\log k - \frac{1}{2}\log\log k} + O\left(\frac{1}{\sqrt{\log k}}\right).$$

Note that the equality from Theorem 7 can be rewritten as

$$\int_{-\infty}^{+\infty}\phi(t)\left(1-\phi(t)^{k-1}\right)\mathrm{d}t = \sqrt{\log k - \frac{1}{2}\log\log k + O(1)}\,.$$

Figure 5 shows that this $O(1)$-term is between -0.6 and -0.8 for k between two and 2^{33}.

PROOF OF THEOREM 1. We recall that RLS needs $\frac{n}{n-i}$ function evaluations on average to increase the fitness by one, provided that it is started in a search point of fitness i, $0 \le i \le n-1$. We can thus rewrite expression (4) to

$$\mathrm{E}[T(\text{RLS}_k)] = \mathrm{E}\left(\sum_{i=0}^{n-1} 1_{\{X \le i\}}\frac{n}{n-i}\right),$$

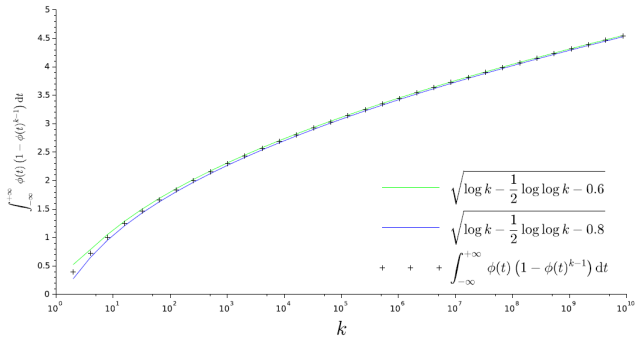

Figure 5: $\int_{-\infty}^{+\infty}\phi(t)\left(1-\phi(t)^{k-1}\right)\mathrm{d}t$ **for** $k \in [2, 2^{33}]$.

where X is the number of ones of the search point returned by **BestOf**(k). By the linearity of expectation this can be written as

$$\sum_{i=0}^{n-1}\mathrm{E}\left(1_{\{X\le i\}}\right)\frac{n}{n-i} = \sum_{i=0}^{n-1}\Pr\left[X \le i\right]\frac{n}{n-i}.$$

As we have noted above, $\Pr[X \le i] = \Pr[X_1,\dots,X_k \le i] = F(i)^k$, for F being the CDF of the binomial distribution with parameters n and $1/2$. Clearly, $F(i) = \sum_{t=0}^{\lfloor i\rfloor}\binom{n}{t}/2^n$.

This shows that we can write

$$\begin{aligned}\Delta_{k,\text{RLS}} &= \mathrm{E}[T(\text{RLS}_1)] - \mathrm{E}[T(\text{RLS}_k)]\\ &= \sum_{i=0}^{n-1}F(i)\frac{n}{n-i} - \sum_{i=0}^{n-1}F(i)^k\frac{n}{n-i}\\ &= \sum_{i=0}^{n-1}F(i)\left(1-F(i)^{k-1}\right)\frac{n}{n-i}.\end{aligned}$$

Clearly we have $\frac{n}{n-i} \le n$ for all $0 \le i \le n-1$, while for $i = n/2 \pm \sqrt{n\log n}$ we also have $\frac{n}{n-i} = 2(1\pm O(\sqrt{\log(n)/n}))$, so that, according to Theorem 6,

$$\Delta_{k,\text{RLS}} = 2\sqrt{\frac{n}{2}}\int_{-\infty}^{+\infty}\phi(t)\left(1-\phi(t)^{k-1}\right)\mathrm{d}t + O\left(k\sqrt{\log n}\right).$$

Finally, Theorem 7 yields that $\Delta_{k,\text{RLS}}$ equals

$$2\sqrt{\frac{n}{2}}\sqrt{\log k - \frac{1}{2}\log\log k} + O\left(\sqrt{\frac{n}{\log k}}\right) + O\left(k\sqrt{\log n}\right).$$

□

6. OPTIMALITY OF RANDOM RESAMPLING

Now that we have seen that we can gain an additive number of $\Theta(\sqrt{n\log n})$ function evaluations by repeated random initialization, we may ask ourselves if a more clever way of choosing the initial search points cannot do even better. Interestingly, this is not possible, as we will show in this section.

We shall use here an assumption implicitly made in most analyses regarding the ONEMAX test function, namely that we are not interested in solely optimizing the particular function ONEMAX, but in algorithms that perform well on any

generalized OneMax function (as defined in equation (1)). Otherwise, of course, there would be better initializations, including taking as first (and last) search point the all-ones string.

THEOREM 8. *Let $k \in \mathbb{N}$. Let $x^{(1)}, \dots, x^{(k)} \in \{0,1\}^n$ and $C > 0$. Then the ratio of $z \in \{0,1\}^n$ such that*

$$\max\{\mathrm{OM}_z(x^{(1)}), \dots, \mathrm{OM}_z(x^{(k)})\} \geq \frac{n}{2} + C\sqrt{n \log k} \quad (5)$$

is at most k^{-2C^2+1}.

The theorem, for example, implies the following special case: Let $k = \omega(1)$ and $C = \frac{1+\varepsilon}{\sqrt{2}}$ for any $\varepsilon > 0$. Then any choice of $x^{(1)}, \dots, x^{(k)} \in \{0,1\}^n$ as set of initial search points will for a $1 - o(1)$ fraction of the generalized OneMax functions not give a better initial fitness than $\frac{n}{2} + \sqrt{(1+\varepsilon)(n/2)\log k}$. Recall from Theorem 3 that $k \in \omega(1) \cap o(\sqrt{n})$ random initial search points for a $1 - o(1)$ fraction of the generalized OneMax functions do give an initial fitness of at least $\frac{n}{2} + \sqrt{(1-\varepsilon)(n/2)\log k}$, $\varepsilon > 0$ any constant. Hence in this range of k, essentially random initial search points give the best possible initial fitness.

Although Theorem 8 may come as a surprise, it is actually not difficult to prove. We replace the counting implicit in talking about the ratio by a probabilistic view and then a simple Chernoff and union bound give the answer.

PROOF OF THEOREM 8. Let us call a $z \in \{0,1\}^n$ satisfying (5) *good*. Let $Z \subseteq \{0,1\}^n$ be the set of all good $z \in \{0,1\}^n$. Now the ratio $|Z|/|\{0,1\}^n| = 2^{-n}|Z|$ of good z is nothing more than the probability that a random $z \in \{0,1\}^n$ is good. We compute this probability and thus prove the theorem.

Let $z \in \{0,1\}^n$ be chosen uniform at random. This is equivalent to saying that each z_i is chosen independently and uniformly at random from $\{0,1\}$. For each $i \in [k]$, regardless of how $x^{(i)}$ is chosen, the additive Chernoff bound (see, e.g., [6, Theorem 1.11]) yields

$$\Pr[\mathrm{OM}_z(x^{(i)}) > \tfrac{n}{2} + C\sqrt{n \log k}] \leq \exp\left(-2C^2 \log k\right) = k^{-2C^2}.$$

Therefore, by a simple union bound, we have

$$\begin{aligned}
&\Pr[\max\{\mathrm{OM}_z(x^{(1)}), \dots, \mathrm{OM}_z(x^{(k)})\} \geq \tfrac{n}{2} + C\sqrt{n \log k}] \\
&\quad = \Pr[\exists i \in [k] : \mathrm{OM}_z(x^{(i)})\} \geq \tfrac{n}{2} + C\sqrt{n \log k}] \\
&\quad \leq k \Pr[\mathrm{OM}_z(x^{(1)})\} \geq \tfrac{n}{2} + C\sqrt{n \log k}] \leq k^{-2C^2+1}.
\end{aligned}$$

□

7. CONCLUSIONS

We have shown that the expected optimization time of search heuristics can be reduced by a simple initialization routine which, instead of sampling just one search point, returns the best of k points that are sampled from the search space independently and uniformly at random. We have shown that for RLS and the $(1+1)$ EA on the generalized OneMax problem we gain an expected number of $\Theta(\sqrt{n \log n})$ fitness evaluations for $k = O(\sqrt{n}/\log n)$. We have also seen that we can not do substantially better than this; not even if we allow ourselves to specifically design k search points for the OneMax problem.

Our result answers a problem left open in [12]. In addition, it is a first step towards a tight bound for the unary unbiased black-box complexity of OneMax.

While it would be interesting to study the resampling initialization for other, more complex, optimization problems, we remark that even for the simple OneMax function our analysis can possibly be tightened. In fact, as commented in the main part, we believe that the condition $k = O(\sqrt{n})$ is not needed. Removing this, however, seems to require an improved version of the Berry-Esseen theorem (Theorem 4).

8. REFERENCES

[1] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics*. World Scientific, 2011.

[2] A. C. Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49:122–136, 1941.

[3] P. A. Borisovsky and A. V. Eremeev. Comparing evolutionary algorithms to the (1+1)-EA. *Theoretical Computer Science*, 403:33–41, 2008.

[4] S. Böttcher, B. Doerr, and F. Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. PPSN'10*, pages 1–10. Springer, 2010.

[5] S. Colin, B. Doerr, and G. Férey. Monotonic functions in EC: anything but monotone! In *Proc. GECCO'14*, pages 753–760. ACM, 2014.

[6] B. Doerr. Analyzing randomized search heuristics: Tools from probability theory. In A. Auger and B. Doerr, editors, *Theory of Randomized Search Heuristics*, pages 1–20. World Scientific Publishing, 2011.

[7] B. Doerr and C. Doerr. The impact of random initialization on the runtime of randomized search heuristics. In *Proc. GECCO'14*, pages 1375–1382. ACM, 2014.

[8] C.-G. Esseen. On the liapunoff limit of error in the theory of probability. *Arkiv för matematik, astronomi och fysik*, A28:1–19, 1942.

[9] H. Hwang, A. Panholzer, N. Rolin, T. Tsai, and W. Chen. Probabilistic analysis of the (1+1)-evolutionary algorithm. *CoRR*, abs/1409.4955, 2014. Available online from http://arxiv.org/abs/1409.4955.

[10] P. K. Lehre and C. Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[11] H. Mühlenbein. How genetic algorithms really work: Mutation and hillclimbing. In *Proc. PPSN'92*, pages 15–26. Elsevier, 1992.

[12] D. Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435, 2013.

[13] C. Witt. Runtime analysis of the (mu + 1) EA on simple pseudo-boolean functions. *Evolutionary Computation*, 14:65–86, 2006.

[14] C. Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22:294–318, 2013.