

Fixed-Target Runtime Analysis

Maxim Buzdalov, Benjamin Doerr, Carola Doerr, Dmitry Vinokurov

▶ **To cite this version:**

Maxim Buzdalov, Benjamin Doerr, Carola Doerr, Dmitry Vinokurov. Fixed-Target Runtime Analysis. Algorithmica, 2022, 84 (6), pp.1762–1793. 10.1007/s00453-021-00881-0 . hal-03377095

HAL Id: hal-03377095

https://hal.sorbonne-universite.fr/hal-03377095

Submitted on 14 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2004.09613v3 [cs.NE] 5 Oct 2021

Fixed-Target Runtime Analysis

Maxim Buzdalov[1], Benjamin Doerr[2], Carola Doerr[3], and Dmitry Vinokurov[1]

[1]ITMO University, Saint Petersburg, Russia
[2]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[3]Sorbonne Université, CNRS, LIP6, Paris, France

October 6, 2021

Abstract

Runtime analysis aims at contributing to our understanding of evolutionary algorithms through mathematical analyses of their runtimes. In the context of discrete optimization problems, runtime analysis classically studies the time needed to find an optimal solution. However, both from a practical and from a theoretical viewpoint, more fine-grained performance measures are needed to gain a more detailed understanding of the main working principles and their resulting performance implications. Two complementary approaches have been suggested: fixed-budget analyses and fixed-target analyses.

In this work, we conduct an in-depth study on the advantages and the limitations of fixed-target analyses. We show that, different from fixed-budget analyses, many classical methods from the runtime analysis of discrete evolutionary algorithms yield fixed-target results without greater effort. We use this to conduct a number of new fixed-target analyses. However, we also point out examples where an extension of existing runtime results to fixed-target results is highly non-trivial.

1 Introduction

The classic performance measure in the theory of evolutionary computation [Jan13, DN20] is optimization time, that is, the number of fitness evaluations that an algorithm uses to find an optimal solution for a given optimization problem. Often only expected optimization times are analyzed and reported, either for reasons of simplicity or because some analysis methods like certain drift theorems [Len20] only yield such bounds.

Some works give more detailed information, e.g., the expectation together with a tail estimate [Köt16, Wit14, DG10]. In some situations, only runtime bounds that hold with some or with high probability are given, either because these are easier to prove or more meaningful (see, e.g., the discussion in [Doe19b] on such statements for estimation-of-distribution algorithms), or because the expectation is unknown [DL17] or infinite. The

use of the notion of stochastic domination [Doe19a] is another way to give more detailed information on runtimes of algorithms.

Nevertheless, all these approaches reduce the whole optimization process to a single point in time: the moment in which an optimal solution is found. For various reasons, more detailed information on the whole process is also desirable, including the following:

(1) Evolutionary algorithms, different from most classic algorithms, are so-called *anytime algorithms*. This means that they can be interrupted at essentially any point of time and they still provide some valid solution (possibly of a low quality). The optimization time as the only performance measure gives no information on how good an algorithm is as an anytime algorithm. Such information, however, is of great interest in practice. It can be used, for instance, if one does not know in advance how much time can be allocated to the execution of an algorithm, or when it is important to report whenever a certain milestone (e.g., quality threshold) has been reached.

(2) When several iterative optimization heuristics are available to solve a problem, one can try to start the optimization with one heuristic and then, at a suitable time, switch to another one which becomes more powerful at that time. To decide which heuristic to use up to a certain point of time or solution quality, more detailed information than the optimization time is needed.

We note that the importance of reporting *runtime profiles* instead of only optimization times has for a long time been recognized in algorithm benchmarking [HAR+20, DWY+18]. These fine-grained performance analyses have helped to advance our understanding of evolutionary computation methods, and have contributed significantly to algorithm development. It is therefore surprising that such more fine-grained notions play only a marginal role in the runtime analysis literature. The following two notions have been used in the runtime analysis community.

- *Fixed-budget analyses:* For a fixed number ("budget") of fitness evaluations, one studies the (usually expected) quality of the best solution found within this budget.
- *Fixed-target analyses:* For a fixed quality threshold, one studies the (usually expected) time (often measured in terms of function evaluations) needed to find a solution of at least this quality.

The main goal of this work is a systematic fixed-target runtime analysis. We provide, in particular, a comparison of different more fine-grained performance measures (Section 2), a survey of the existing results (Section 4), an analysis how the existing methods to prove runtime analysis results can be used to also give fixed-target results (Sections 5 and 6) together with several applications of these methods, some to reprove existing results, others to prove new fixed-target results. The main insight here is that fixed-target results often come almost for free when one can prove a result on the optimization time, so it is a waste to not report them explicitly. However, in Section 7 we also point out situations in which the runtime is well understood, but the derivation of fixed-target results appears very difficult.

The preliminary version of this work has been published in proceedings of the GECCO conference [BDDV20]. In this present version, apart for implementing properly the pieces that have been shortened or omitted during to the conference page limit and extending some of the results to evolutionary algorithms with *fast* mutation operators, we have

introduced new versions of variable and multiplicative drift theorems that explicitly use the expected potential at the moment of stopping and hence yield better bounds in the fixed-target settings.

2 Fine-Grained Runtime Analysis: Fixed-Budget and Fixed-Target Analyses

Among the notions other than the time required to find an optimum, the first notion to become the object of rigorous mathematical analysis is *fixed-budget analysis* [JZ14a]. Fixed-budget analysis asks, given a computational budget $b \in \mathbb{N}$, for the expected fitness of the best solution seen within b fitness evaluations. In the first paper devoted to this topic (extending results presented at GECCO 2012), Jansen and Zarges [JZ14a] investigated the fixed-budget behavior of two simple algorithms, randomized local search (RLS) and the $(1+1)$ evolutionary algorithm (the $(1+1)$ EA), on a range of frequently analyzed example problems. For these two elitist algorithms, fixed budget analysis amounts to computing or estimating $f(x_b)$, where f is the objective function and x_b is the b-th search point generated by the algorithm. Jansen and Zarges considered small budgets, that is, budgets b below the expected optimization time, and argued that instead of larger budgets, one should rather regard the probability to find the optimum within the budget.

Jansen and Zarges [JZ14a] obtained rather simple expressions for the fixed-budget fitness obtained by RLS, but those for the $(1+1)$ EA were quite complicated. In [JZ14b], the same authors evaluated artificial immune systems (AIS). In terms of the classic optimization time, AIS are typically worse than evolutionary algorithms. Interestingly, in the fixed-budget perspective with relatively small budgets, AIS were proven to outperform evolutionary algorithms, confirming our claim that fine-grained runtime results can lead to insights that cannot be (easily) obtained from studying optimization times only.

These first results were achieved using proof techniques highly tailored to the considered algorithms and problems. Given the innocent-looking definition of fixed-budget fitness, the proofs were quite technical even for simple settings like RLS optimizing LeadingOnes. The analyses were even more complicated for the $(1+1)$ EA and many analyses could not cover the whole range of budgets (e.g., for LeadingOnes, only budgets below $0.5n^2$ were covered, whereas the (strongly concentrated) optimization time is around $0.86n^2$, see [BDN10]).

In [DJWZ13], a first general approach to proving fixed-budget results was presented. Interestingly, it works by estimating fixed-target runtimes and then using strong concentration results to translate the fixed-target result into a fixed-budget result. This might be the first work that explicitly defines the fixed-target runtime, without however using this name. The paper [LS15] also uses fixed-target runtimes (called *approximation times*, see [LS15, Corollary 3]) for a fixed-budget analysis, but most of the results in the paper are achieved by employing drift arguments. An explicit collection of drift theorems designed for fixed-budget analyses, along with an application to derive fixed-budget results for the well-studied LeadingOnes problem, can be found in [KW20].

The first fixed-budget analysis for a combinatorial optimization problem was conducted in [NNS17]. Subsequently, several papers more concerned with classical optimization time also formulated their results in the fixed-budget perspective, among

them [DDY16b, DDY20, DDY16a].

A similar notion of fine-grained runtime analysis, called the *unlimited budget analysis* [HJZ19], was recently proposed. It can be seen as either a complement to the fixed-budget analysis (as its primary goal is to measure how close an algorithm gets to the optimum of the problem in a rather large number of steps) or as an extension of fixed-budget analysis which goes beyond using small budgets only.

For the second main fine-grained runtime notion, *fixed-target analysis*, due to it being a direct extension of the optimization time, it is harder to attribute a birthplace. As we argue also in Section 5, the fitness level method is intimately related to the fixed-target view. As such, many classic papers can be seen as fixed-target works, which is particularly true for papers where the fitness level method is not used as a black box, but one explicitly splits a runtime analysis into the time to reach a particular fitness and the another time to find the optimum, as done, e.g., in [Wit06]. The first, to the best of our knowledge, explicit definition of the fixed-target runtime in a runtime analysis paper can be found in the above-mentioned fixed-budget work [DJWZ13, Section 3]. There, for a given algorithm A, a given objective function f (to be maximized), and a given target $k \in \mathbb{R}$, the fixed-target runtime $T_{A,f}(k)$ is defined as the number of fitness evaluations after which a search point of fitness at least k is found. Since this notion was merely used as a tool in a proof, the name *fixed-target runtime* was not used yet. The paper [CD18] argued that fixed-target results, coined *runtime profiles* in [CD18], should be made explicit, to provide more information to practitioners. The name *fixed-target analysis* was, in the context of runtime analysis, first used in the GECCO 2019 student workshop paper [VBB+19], the only other work putting fixed-target analysis into its center. It is also worth noting that certain papers combine theoretical results for optimization time and experimental fixed-target results, [DYvR+18] being a good example, which suggests that there is a demand for a method for an easy translation of the results between these two areas.

In summary, we see that there are generally not too many runtime results that give additional information on how the process progresses over time. Since fixed-budget analysis, as a topic on its own, was introduced earlier into the runtime analysis community, there are more results on fixed-budget analysis. At the same time, by looking over all fixed-budget and fixed-target results, it appears that the fixed-budget results tend to be harder to obtain.

From the viewpoint of designing dynamic algorithms, that is, algorithms that change parameter values or sub-heuristics during the runtime, it appears to us that fixed-budget results are more useful for time-dependent dynamic choices, whereas fixed-target results aid the design of fitness-dependent schemes. If algorithm A with a budget of b computes better solutions than algorithm B with the same budget, then in a time-dependent scheme one would rather run algorithm A for the first b iterations than B. However, if the runtime to the fitness target x of algorithm A is lower than that of B, then a fitness-dependent scheme would use rather A than B to reach a fitness of at least x.

Since we do not see that the increased difficulty of obtaining fixed-budget results is compensated by being significantly more informative or easier to interpret and since we currently see more fitness-dependent algorithm designs (e.g., [BDN10, DDE15, DDY16b, LS20] than time-dependent ones (where, in fact, we are only aware of a very rough proof-of-concept evolutionary algorithm in the early work [JW06]), we advocate in this work

Algorithm 1 The $(\mu + \lambda)$ EA to maximize $f : \{0,1\}^n \to \mathbb{R}$

Require: mutation strength distribution $\mathcal{M}$, initialization distribution $\mathcal{D}$
  for $i \in [\mu]$ **do**
    $x_i \leftarrow$ sample from $\mathcal{D}$
    query $f(x_i)$
  end for
  $X \leftarrow \{x_1, \dots, x_\mu\}$
  while true do
    for $i \in [\lambda]$ **do**
      $j \leftarrow$ sample uniformly from $[\mu]$
      $\ell \leftarrow$ sample from $\mathcal{M}$
      $y_i \leftarrow$ flip ℓ pairwise different, uniformly chosen bits in x_j
      query $f(y_i)$
    end for
    $Y \leftarrow \{y_1, \dots, y_\lambda\}$
    $X \leftarrow \mu$ best solutions from $X \cup Y$, breaking ties randomly,
      preferring offspring in the case of ties
  end while

to rather focus on fixed-target results. We support this view by further elaborating how the existing analyses methods for the optimization time, in particular, the fitness-level methods and drift, can easily be adapted to also give fixed-target results.

3 Preliminaries

Throughout the paper we use the notation $[a..b]$ to denote a set of integer numbers $\{a, a+1, \dots, b-1, b\}$, and we denote the set $[1..n]$ as $[n]$. We write H_n for the n-th harmonic number, that is, $H_n = \sum_{i=1}^{n} 1/i$, and $H_0 = 0$ for convenience. Finally, we use the shorthand $[\mathcal{E}]$ to denote the function that returns 1 if event $\mathcal{E}$ holds true, and which returns 0 otherwise.

We consider simple algorithms, such as the $(1+1)$ EA, the $(\mu+1)$ EA, and the $(1+\lambda)$ EA, which solve optimization problems on bit strings of length n. Due to the increased interest in mutation operators that do not produce offspring identical to the parent [CD18], and to mutation operators different to standard bit mutation, such as the *fast* mutation operators sampling from heavy-tailed distributions [DLMN17], we consider them in a generalized form, which samples the number of bits to flip from some distribution $\mathcal{M}$. We also use a distribution over search points $\mathcal{D}$ during initialization. A default choice for $\mathcal{D}$ is to sample every bit string with equal probability, however, we consider also initialization with the search point having smallest possible fitness value. These algorithms are presented in the most general form in Algorithm 1 as a $(\mu+\lambda)$ EA parameterized by $\mathcal{M}$ and $\mathcal{D}$.

We consider the following distributions of $\mathcal{M}$ for the $(1+1)$ EA:

- randomized local search, or RLS: $\mathcal{M} = 1$;

- the $(1+1)$ EA with standard bit mutation: $\mathcal{M} = B(n,p)$, where $B(n,p)$ is the binomial distribution;
- the $(1+1)$ $\text{EA}_{0\to1}$ using the *shift mutation strategy*: $\mathcal{M} = \max\{1, B(n,p)\}$;
- the $(1+1)$ $\text{EA}_{>0}$ using the *resampling mutation strategy*: $\mathcal{M} = [x \sim B(n,p) \mid x > 0]$;
- the *fast* $(1+1)$ EA with parameter $\beta > 1$: $\mathcal{M} = [x \sim B(n, a/n) \mid a \sim \mathcal{H}_\beta]$, and $\mathcal{H}_\beta$ is a distribution defined as follows:

$$\Pr[x = a \mid x \sim \mathcal{H}_\beta] = a^{-\beta} \cdot (C^\beta_{n/2})^{-1},$$

  where $C^\beta_{n/2} = \sum_{i=1}^{n/2} i^{-\beta}$ is the normalization constant.

Note that, in fact, we could have also considered the *fast* $(1+1)$ EA with the shift or replacement mutation strategy, as well as the version of the *fast* $(1+1)$ EA that directly samples the number of bits to flip from $\mathcal{H}_\beta$ without using the binomial distribution as a proxy. However, since this paper would not benefit from repetitive analyses of very similar algorithms, we restrict ourselves only to the canonical *fast* $(1+1)$ EA.

Sometimes we are only interested in the probability q of flipping a particular bit while not flipping any other bit. For problem size n and mutation strength p, the values of q for the algorithms above are

- RLS: $q = 1/n$;
- $(1+1)$ EA: $q = p(1-p)^{n-1}$;
- $(1+1)$ $\text{EA}_{0\to1}$: $q = p(1-p)^{n-1} + \frac{(1-p)^n}{n}$;
- $(1+1)$ $\text{EA}_{>0}$: $q = \frac{p(1-p)^{n-1}}{1-(1-p)^n}$;
- *fast* $(1+1)$ EA_β: $q = \frac{1}{n} \cdot (C^\beta_{n/2})^{-1} \cdot \gamma(n,\beta) = \frac{1}{n} \cdot (C^\beta_{n/2})^{-1} \cdot \Theta(1)$, as the factor $\gamma(n,\beta) = \sum_{i=1}^{n/2} i^{1-\beta}(1-\frac{i}{n})^{n-1}$ is between e^{-1} and $\frac{2\pi^2}{3}$ by [DLMN17, Lemma 5].

We consider the following classical problems on bit strings:

$$\begin{aligned}
\textsc{OneMax}(x) &\mapsto \sum\nolimits_{i=1}^{n} x_i \\
\textsc{LeadingOnes}(x) &\mapsto \sum\nolimits_{i=1}^{n} \prod\nolimits_{j=1}^{i} x_i \\
\textsc{BinVal}(x) &\mapsto \sum\nolimits_{i=1}^{n} 2^{i-1} x_i.
\end{aligned}$$

We also consider the minimum spanning tree (MST) problem. Given a connected undirected graph G with positive weights on each edge, the MST problem asks to find a minimum spanning tree of it, that is, a subgraph that connects all vertices of G and that has the minimum possible weight. This problem was adapted to bit strings as in [NW07] as follows: each bit corresponds to one edge of G, and the bit value of 1 means that the edge is included in the subgraph.

4 Overview of Known Fixed-Target Results

In this section, we comment on the known fixed-target results available in recent papers. We cover both the results and the techniques which have been used and possibly modified to achieve these results. Where necessary, we estimate the precision of these results by comparing them with the actual expected hitting times of the corresponding algorithms. In the whole section, n is the problem size and k is the target fitness.

4.1 LeadingOnes

LeadingOnes was the first problem for which certain fixed-target results were derived. The paper [Wit06] studied upper and lower bounds on the runtime of the $(\mu+1)$ EA on several benchmark problems. For LeadingOnes, the runtime was proven in [Wit06, Theorem 1] using a technique similar to fitness levels, where the state space also involved the number of the best individuals in the population. Then, [Wit06, Corollary 1] bounded the expectation of the time needed to reach a state of at least k leading ones from above by $\mu + 3ek \cdot \max\{\mu \ln en, n\}$, which is a fixed-target result. In the framework of that paper, this result appeared to be useful in a subsequent part, where the necessity of having population size $\mu > 1$ was discussed by constructing and analyzing an artificial problem involving both OneMax and LeadingOnes.

Böttcher et al. [BDN10] proved exact expected runtimes for the $(1+1)$ EA on LeadingOnes, which was an important cornerstone in the studies of LeadingOnes. In this paper it is proved for the $(1+1)$ EA that the expected time to leave a state with the fitness of i and the mutation probability p is $E(A_i) = 1/((1-p)^i p)$. However, the next fitness value may be greater than $i+1$. The problem-dependent observation, which simplifies the analysis, is that the bits following the i-th bit have a probability of exactly $1/2$ to equal 1. This yields the expected optimization time to be exactly $((1-p)^{1-n} - (1-p))/(2p^2)$ for a fixed p.

This result was reused in [DJWZ13, Section 4] to prove that the expected time that the $(1+1)$ EA with mutation probability $p \in (0,1)$ needs to hit target $k \in [n]$ equals

$$\frac{(1-p)^{1-k} - (1-p)}{2p^2}. \tag{1}$$

Technically, the result in [DJWZ13] is given only for $p = 1/n$, partially because they used this mutation probability throughout the whole paper. However, their proof does not depend on the value of p and hence extends to the general case.

Theorem 11 in [CD18] extends this result to similar algorithms: the $(1+1)$ $\text{EA}_{>0}$, the $(1+1)$ $\text{EA}_{0\to1}$, and RLS. While only upper bounds are claimed in that work, it is clear from the above that these bounds are exact. As the results of our techniques are identical, we direct the reader to our Theorem 4.

Finally, in [Doe19a, Corollary 4] an even stronger result is proven about the exact *distribution* of the time the $(1+1)$ EA needs to hit a certain target. The result is expressed in terms of random variables X_i for initial bit values and the probabilities q_i to generate a point with a strictly better fitness from a point with fitness i, and reads as $\sum_{i=0}^{k-1} X_i \cdot \text{Geom}(q_i)$, where Geom is the geometric distribution.

The summary of the results for LeadingOnes is given in Table 1.

Table 1: Summary of the fixed-target results available in literature for LeadingOnes

Algorithm	Result	Constraints	Reference
$(\mu+1)$ EA	$\leq \mu + 3ek \cdot \max\{\mu \ln(en), n\}$	$p = 1/n$, $\mu = \text{poly}(n)$	[Wit06, Cor. 1]
$(1+1)$ EA	$= \frac{n^2-n}{2} \cdot \left(\left(1+\frac{1}{n-1}\right)^k - 1\right)$	$p = 1/n$	[DJWZ13, Sec. 4]
RLS	$=^{(1)}\ kn/2$	–	[CD18, Th. 11]
$(1+1)$ EA	$=^{(1)}\ \frac{(1-p)^{1-k}-(1-p)}{2p^2}$	–	[CD18, Th. 11]
$(1+1)$ $\text{EA}_{0\to1}$	$=^{(1)}\ \frac{1}{2}\sum_{i=n-k+1}^{n} \frac{1}{p(1-p)^{n-i}+\frac{1}{n}(1-p)^n}$	–	[CD18, Th. 11]
$(1+1)$ $\text{EA}_{>0}$	$=^{(1)}\ \frac{1-(1-p)^n}{2p^2}((1-p)^{1-k}-(1-p))$	–	[CD18, Th. 11]

(1)Note: [CD18, Th. 11] states only the upper bound. However, using the same arguments as in [DJWZ13, Sec. 4], it is not difficult to verify that these bounds are exact.

4.2 OneMax

To the best of our knowledge, explicitly formulated fixed-target results regarding OneMax exist so far only for the $(1+1)$ EA and similar algorithms. What is more, due to the fact that the $(1+1)$ EA shows different expected progress for different fitness values on the same problem, for which it is hard to find sharp bounds, the available upper and lower fixed-target bounds for OneMax are generally less precise than those for LeadingOnes. For this reason, we augment the existing theoretical results with the actual runtime profiles of the $(1+1)$ EA, which are given in Figure 1 for three problem sizes $n = 10^3, 10^4, 10^5$.

The first result for OneMax appeared in [LS15, Corollary 3], again in a fixed-budget setting. It is a lower bound, which was proven using fitness levels for lower bounds [Sud13]. This result says that, for the $(1+1)$ EA with mutation probability $p = 1/n$ optimizing OneMax, the expected time until hitting a target k is at least

$$en \ln(n/(n-k)) - 2n \ln \ln n - 16n, \tag{2}$$

which was used in [LS15] to prove fixed-budget results.

This bound is aimed at estimating the time needed to reach targets which are very close to the optimum. Since the $(1+1)$ EA spends most of its time in these regions, and because [LS15] is dedicated to fixed-budget estimations, this choice seems reasonable. However, from the fixed-target perspective, (2) results in a satisfactory lower bound only for a small fraction of possible conditions. In particular, it turns negative when the target is smaller than $n \cdot (1 - (\ln n)^{-2/e})$, so, for instance, it cannot be used to estimate the time needed to reach targets of the form cn, where $0 < c < 1$ is a constant factor. The combination of the constant factors also renders it usable only for quite large values of n: for instance, at $n = 10^4$ only five values $k \in [9996..10^4]$ produce nonzero values. For this reason, Figure 1 contains special subfigures which highlight this particular bound at targets close to the optimum.

In [CD18, Theorem 10], the first fixed-target upper bounds were given for the $(1+1)$ EA, the $(1+1)$ $\text{EA}_{>0}$, the $(1+1)$ $\text{EA}_{0\to1}$, and for RLS on OneMax. They cover the whole range of targets, as well as arbitrary mutation probabilities. They are

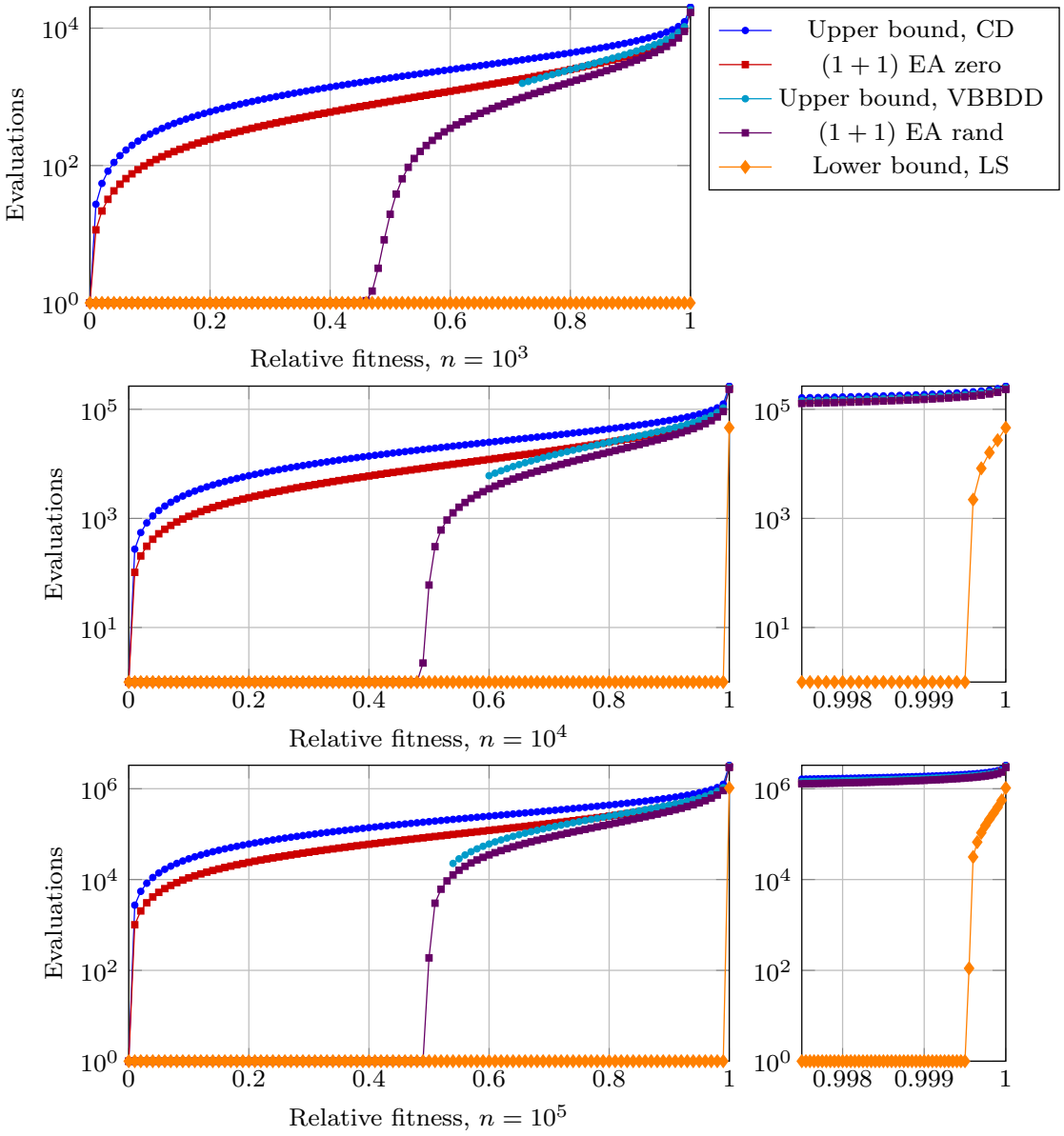


Figure 1: The runtime profile of the $(1+1)$ EA on OneMax and the applicable bounds. Plots in the left part of the figure show the general picture for three problem sizes $n \in \{10^3, 10^4, 10^5\}$, whereas plots in the right part highlight the regions close to the optimum. The plotted quantity is the expected number of fitness evaluations required to reach the target fitness value against that (relative) fitness value. The runtime profiles for both cases regarding $(1+1)$ EA were computed exactly using dynamic programming following the ideas of [BD19]. For the upper and lower bounds, the respective expressions were evaluated and plotted, disregarding the $(1 \pm o(1))$ factors. The notation in the legend is as follows: "$(1+1)$ EA zero" is the $(1+1)$ EA starting at the all-zero bit string, "$(1+1)$ EA rand" is the $(1+1)$ EA starting at a randomly generated bit string, "Upper bound, CD" is the bound from [CD18, Theorem 10], "Upper bound, VBBDD" is the bound from [VBB$^+$19, Theorem 3.1] applicable to "$(1+1)$ EA rand", and finally "Lower bound, LS" is the bound from [LS15, Corollary 3].

Table 2: Insights about the ratio of the upper bound on the fixed-target runtime of the $(1+1)$ EA on OneMax from [CD18, Th. 10] and the actual expected hitting times computed using dynamic programming following the ideas of [BD19]. We present the maximum of the ratio across all targets $k \in [n]$, as well as the ranges of targets where the ratio is at least some fixed value (values of 1.5 and 2.5 are used). Note how the left endpoints of these ranges remain the same as the problem size n grows, whereas the right endpoints scale linearly with remarkable precision. The maximum ratio slowly grows and apparently approaches a constant, which resembles $e = 2.71\ldots$ quite closely.

n	Maximum	Ratio ≥ 2.5		Ratio ≥ 1.5	
	ratio	Min target	Max target	Min target	Max target
10^3	2.606391	19	221	2	920
10^4	2.681519	18	2320	2	9205
10^5	2.706512	18	23306	2	92057

based on simple fitness level arguments, and for simplicity they assume the worst case regarding initialization, that is, that the algorithm starts at the all-zero bit string. As a result, all these bounds have a form of of $\frac{1}{q}(H_n - H_{n-k})$, where H_i is the i-th harmonic number and q is the lower bound on the probability of flipping exactly one bit. This bound is exact for RLS and also captures the behavior of other $(1+1)$ EA flavors quite well (see Figure 1 for the example of the standard $(1+1)$ EA).

To quantify how much the upper bound for the $(1+1)$ EA on OneMax from [CD18, Theorem 10] differs from the actual times needed for the $(1+1)$ EA to hit various targets when starting at the all-zeros bit string, we reuse the data from Figure 1 to compute the ratios of the upper bound to the corresponding average target-hitting time. We present these insights in Table 2, from which one can infer, for instance, that the upper bound from [CD18, Theorem 10] overestimates the true hitting times at most by a factor that tends to some constant value.

While being moderately precise for estimating the fixed-target runtimes of the $(1+1)$ EA, and similar algorithms, on OneMax starting from the all-zeros bit string, these bounds overestimate the fixed-target results much more when these algorithms start from a randomly generated bit string, which is a more realistic condition. For random initialization, [VBB+19, Theorem 3.1] proves a similar upper bound of $\frac{1}{q}(H_{n/2} - H_{n-k})(1 - o(1))$ for $k > n/2 + \sqrt{n}\ln n$ using the same technique and additional arguments to use $H_{n/2}$. Technically, this result was proven only for $(1+1)$ EA$_{>0}$, however, the proof holds also for arbitrary probabilities q of flipping exactly one bit. Such a bound gives a moderately good approximation, as illustrated in Figure 1 for the classic $(1+1)$ EA. The evaluation of the approximation quality, similar to the one shown in Table 2, shows that the constant factors in the approximation are better: the maximum ratio for $n = 10^5$ is less than 1.8, and the ratio gets smaller than 1.5 at smaller $k \approx 0.815n$.

In [Wit13, Theorem 6.5], lower bounds for the runtimes of the $(1+1)$ EA on OneMax are proven, assuming arbitrary mutation probabilities p satisfying $p = O(n^{-2/3-\varepsilon})$. For this, an interval $S = [n\tilde{p}\ln^2 n; 1/(2\tilde{p}^2 n\ln n)]$ of distances to the optimum is considered, where $\tilde{p} = \max\{1/n, p\}$. Multiplicative drift for lower bounds is applied to this interval

Table 3: Summary of the fixed-target results available in literature for OneMax

Algorithm	Result	Constraints	Reference
$(1+1)$ EA	$\geq en\ln(n/(n-k)) - 2n\ln\ln n - 16n$	$p = 1/n$	[LS15, Cor. 3]
RLS	$\leq n \cdot (H_n - H_{n-k})$	–	[CD18, Th. 10]
$(1+1)$ EA	$\leq \frac{H_n - H_{n-k}}{p(1-p)^{n-1}}$	–	[CD18, Th. 10]
$(1+1)$ $\text{EA}_{0\to1}$	$\leq \frac{H_n - H_{n-k}}{(1-p)^{n-1}(p+1/n-p/n)}$	–	[CD18, Th. 10]
$(1+1)$ $\text{EA}_{>0}$	$\leq \frac{1-(1-p)^n}{p(1-p)^{n-1}}(H_n - H_{n-k})$	–	[CD18, Th. 10]
$(1+1)$ $\text{EA}_{>0}$	$\leq \frac{1-(1-p)^n}{p(1-p)^{n-1}}(H_{n/2} - H_{n-k}) \cdot (1-o(1))$	$k > n/2 + \sqrt{n}\ln n$	[VBB+19, Th. 3.1]
$(1+1)$ $\text{EA}_{>0}$	$\geq (1-o(1))\frac{1-(1-p)^n}{p(1-p)^{n-1}} \ln \frac{1}{4\tilde{p}^3 n^2 \ln^3 n}$	$p = O(n^{-2/3-\varepsilon})$,	[VBB+19, Th. 3.4]
	where $\tilde{p} = \max\{1/n, p\}$	$k \geq n - n\tilde{p}\ln^2 n$	
$(1+1)$ $\text{EA}_{>0}$	$\geq (1-o(1))\frac{1-(1-p)^n}{p(1-p)^{n-1}} \ln \frac{1}{4\tilde{p}^2 n(n-k)\ln n}$	$p = O(n^{-2/3-\varepsilon})$,	[VBB+19, Th. 3.4]
	where $\tilde{p} = \max\{1/n, p\}$	$k \leq n - n\tilde{p}\ln^2 n$,	
		$n - k = o(1/(2\tilde{p}^2 n\ln n))$	

to yield the lower bound of

$$(1-o(1))(1-p)^{-n}(1/p)\min\{\ln n, \ln(1/(p^3n^2))\}.$$

This result was used in [VBB+19] to obtain the fixed-target results for the $(1+1)$ $\text{EA}_{>0}$ on OneMax. The small difference between the $(1+1)$ EA and the $(1+1)$ $\text{EA}_{>0}$, from the point of view of the above-mentioned proof, is only in one of the factors, which depends on p in a slightly different way. The obtained fixed-target results are piecewise. Indeed, [Wit13, Theorem 6.5] uses the time that the $(1+1)$ EA needs to go through an interval of fitness values S. For this reason, for all targets k that do not belong to this interval, $k \geq n - n\tilde{p}\ln^2 n$, the fixed-target runtime bound does not depend on k. What is more, the bounds contain a factor $1-o(1)$ which is hard to estimate well, and whose value appears to be large enough for practical values of n. For this reason we do not show the plots for these bounds in Figure 1.

The summary of the results for OneMax is given in Table 3.

4.3 BinVal

The fixed-target bounds for BinVal were proven in [VBB+19] for the $(1+1)$ $\text{EA}_{>0}$. The methods for proving optimization times for linear functions, such as the ones in [Wit13], were found to be insufficient, so a problem-dependent observation was used. To achieve a target value k such that $2^n - 2^t \leq k < 2^n - 2^{t+1}$, one requires to optimize the t heaviest bits, and it is enough to optimize the $t+1$ heaviest bits. As a result, reaching the target k is equivalent to solving BinVal of size $t + O(1)$ to optimality using a n/t times smaller mutation rate. The quite complicated bounds from [Wit13] were adapted to the case of BinVal in [VBB+19, Theorem 4.1]. The summary of these results is given in Table 4.

5 Fitness Levels

In this section we consider the fitness level theorems in the fixed-target context. A key take-away, implicitly known in the community for years, is that the most important

Table 4: Summary of the fixed-target results for BinVal

Algorithm	Result	Constraints	Reference
$(1+1)$ EA$_{>0}$	$\geq (1-o(1))\frac{1-(1-p)^n}{p(1-p)^{n^-}} \min\left\{\ln n^-, \ln \frac{1}{p^3(n^-)^2}\right\}$, $n^- = n - \lceil \log_2(2^n-k)\rceil$	$p = O(n^{-2/3-\varepsilon})$	[VBB+19, Th. 4.1]
$(1+1)$ EA$_{>0}$	$\leq \frac{pn^+\alpha^2(1-p)^{1-n^+}+\alpha\left(\ln\frac{1}{p}+(n^+-1)\ln(1-p)+1\right)}{(1-p)^{n^+-1}\cdot p(\alpha-1)\cdot(1-(1-p)^n)^{-1}}$, $n^+ = n - \lfloor \log_2(2^n-k)\rfloor$, α arbitrary e.g. $\ln\ln n^+$	$p = O(n^{-2/3-\varepsilon})$	[VBB+19, Th. 4.1]

theorems of this sort are already suitable to produce fixed-target results.

5.1 Fitness Level Theorems

In the fitness level method (also known as *artificial fitness levels* and *the method of f-based partitions* [Weg02]), the state of the algorithm is typically described by the best fitness of an individual in the population. It is transformed into a *fitness level*, which may aggregate one or more consecutive fitness values. For the fitness function f (to be maximized) and two search space subsets A and B one writes $A <_f B$ if $f(a) < f(b)$ for all $a \in A$ and $b \in B$. A partition $\{A_1 \dots A_m\}$ of the search space, such that $A_i <_f A_{i+1}$ for all $1 \leq i < m$ and A_m contains only the optima of the problem, is called an *f-based partition*. If the best individual in a population of an algorithm $\mathcal{A}$ has a fitness $f \in A_i$, then the algorithm is said to be in A_i.

Fitness level theorems work best for the algorithms that typically visit the fitness levels one by one in the increasing order. We summarize below several popular fitness level theorems. We note that a simpler version of the fitness level theorems in [Sud13] was recently given in [DK21]. Since this work appeared significantly after the original preparation of this manuscript, we do not discuss it any further, but note nevertheless that for future fixed-target analyses it might be easier to use than the theorems from [Sud13].

Theorem 1 (Fitness levels for upper bounds; Lemma 1 from [Weg02])**.** *Let $\{A_i\}_{1\leq i\leq m}$ be an f-based partition, and let p_i be a lower bound for the probability that the elitist algorithm $\mathcal{A}$ samples a search point belonging to $A_{i+1} \cup \dots \cup A_m$ provided it currently is in A_i. Then the expected hitting time of A_m is at most*

$$\sum_{i=1}^{m-1} P[\mathcal{A} \text{ starts in } A_i] \cdot \sum_{j=i}^{m-1} \frac{1}{p_i} \leq \sum_{j=1}^{m-1} \frac{1}{p_i}. \tag{3}$$

Theorem 2 (Fitness levels for lower bounds; Theorem 3 from [Sud13])**.** *Let $\{A_i\}_{1\leq i\leq m}$ be a partition of the search space. Let the probability for the elitist algorithm $\mathcal{A}$ to transfer in one step from A_i to A_j, $i < j$, to be at most $u_i \cdot \gamma_{i,j}$, and let $\sum_{j=i+1}^m \gamma_{i,j} = 1$. Assume there is some $\chi \in [0,1]$ such that $\gamma_{i,j} \geq \chi \sum_{k=j}^m \gamma_{i,k}$ for all $1 \leq i < j \leq m$. Then the expected hitting time of A_m is at least*

$$\sum_{i=1}^{m-1} P[\mathcal{A} \text{ starts in } A_i] \cdot \left(\frac{1}{u_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{u_i}\right). \tag{4}$$

We also use an upper-bound theorem similar to Theorem 2.

Theorem 3 (Refined fitness levels for upper bounds; Theorem 4 from [Sud13])**.** *Let $\{A_i\}_{1\leq i\leq m}$ be a partition of the search space. Let the probability for the elitist algorithm $\mathcal{A}$ to transfer in one step from A_i to A_j, $i<j$, to be at least $s_i\cdot\gamma_{i,j}$, and let $\sum_{j=i+1}^{m}\gamma_{i,j}=1$. Assume there is some $\chi\in[0,1]$ such that $\gamma_{i,j}\leq\chi\sum_{k=j}^{m}\gamma_{i,k}$ for all $1\leq i<j<m$. Further, assume that $(1-\chi)s_i\leq s_{i+1}$ for all $1\leq i\leq m-2$. Then the expected hitting time of A_m is at most*

$$\sum_{i=1}^{m-1}P[\mathcal{A}\text{ starts in }A_i]\cdot\left(\frac{1}{s_i}+\chi\sum_{j=i+1}^{m-1}\frac{1}{s_i}\right). \tag{5}$$

Theorems 1–3 are applicable only to elitist algorithms. However, a fitness level theorem was proposed in [CDEL18] that can be applied to non-elitist algorithms as well (see [DK19] for a sharper version of this result).

Observation 1. *If one of Theorems 1–3 is proven for a certain algorithm on a certain problem, it also holds if a new target fitness level m' is chosen, such that $1<m'<m$, and all fitness levels $A_{m'},A_{m'+1},\ldots,A_m$ are merged.*

Proof. This is essentially the same argument as in [LS15].

This modification does not alter the estimations of probabilities to leave fitness levels preceding m': p_i, u_i and s_i for Theorems 1, 2, and 3, respectively. The only affected locations are the constraints on $\gamma_{i,j}$. Their affected occurrences on the right-hand sides effectively merge by summing up, e.g., $\gamma'_{i,m'}\leftarrow\sum_{k=m'}^{m}\gamma_{i,k}$. Note that only those right-hand sides, which contain the complete sums from m' to m, survive after the transformation, and not just their parts. For the left-hand sides, only those $\gamma_{i,j}$ survive where $j=m'$, as all others are either unchanged or cease to exist. However, these occurrences are trivial, since they are limited only by identity inequalities $\gamma_{i,m'}\leq\chi\gamma_{i,m'}$ in Theorem 2 and are not limited by anything in Theorem 3 as their limits are conditioned on $j<m'$. □

It follows from Observation 1 that it is very easy to obtain fixed-target results from the existing optimization time results whose proofs use the above-mentioned fitness level theorems.

Note that, technically, Theorems 2, 3 and the theorem from [CDEL18] do not require the employed partition of the search space to be an f-based partition. Formally speaking, this enables using them in the fixed-target context as is, contrary to the original formulation of fitness levels for upper bounds, where the last partition must contain optima and nothing else. However, this alone does not yet allow to reuse the existing optimization time results in order to obtain the fixed-target ones. Observation 1 fills this gap by specifying the sufficient conditions for this to be possible.

We also note that Theorems 2 and 3 only require the transition graph over the employed partition to be acyclic with regard to the analyzed algorithm, but does not require that a fitness level A_m is reachable from another fitness level A_n for every $n<m$. In fact, Observation 1 may be extended without much effort to allow merging of fitness levels whose indices do not form a contiguous sequence, provided that one cannot reach a non-merged fitness level from any of the merged levels. This feature might have applications in analysis of certain local search algorithms.

5.2 Applications

5.2.1 Hill Climbers on LeadingOnes

We re-prove here the statements about the fixed-target performance of the algorithms from the $(1+1)$ EA family, which were proven in [DJWZ13, Section 4] and stated, but not formally proven, in [CD18, Theorem 11]. For this we use Theorems 2 and 3, similarly to their use in [Sud13] to prove the results from [BDN10] with fitness levels alone.

Lemma 1. *In the context of Theorems 2 and 3 for* LeadingOnes*, assuming the target fitness is k, the values $\gamma_{i,j} = 2^{i-j}$ when $j < k$, $\gamma_{i,k} = 2^{i-k+1}$, and $\chi = 1/2$ satisfy their requirements.*

Proof. Recall that for LeadingOnes, if i is the fitness of an individual, its first $i+1$ bits are known, and the bits at indices $\{i+2, \dots, n\}$ are uniformly distributed. Thus, conditioned on mutation being applied and the fitness being improved, the probability of transferring to fitness level j is $\gamma_{i,j} = 2^{-j+i+[j=n]}$. Due to [Sud13, Theorem 5], the constant χ to be used in the theorems above is equal to $1/2$.

In the fixed-target context, the transition probabilities $\gamma_{i,j}$ where $j < k$ remain the same, as the underlying process is not changed, whereas since fitness levels k, $k+1$, …, n merge, the probability $\gamma_{i,k} = \gamma^{\text{old}}_{i,k} + \gamma^{\text{old}}_{i,k+1} + \dots$ results in telescoping the inverse powers of two. Since fitness level merging effectively reduces the problem size from n to k without affecting anything else, the constant value $\chi = 1/2$ also remains valid. □

Lemma 2. *Assume that q_i is the probability for algorithm $\mathcal{A}$ to flip a given bit while not flipping any of some other i given bits. The expected time for $\mathcal{A}$ to reach a target of at least k on* LeadingOnes *of size n is:*

$$\sum_{i=0}^{k-1} P[\mathcal{A} \text{ starts with fitness } i] \cdot \left(\frac{1}{q_i} + \frac{1}{2} \sum_{j=i+1}^{k-1} \frac{1}{q_j} \right) = \frac{1}{2} \sum_{i=0}^{k-1} \frac{1}{q_i}. \tag{6}$$

Proof. The left-hand side of the lemma statement follows from Lemma 1, Theorems 2 and 3, and Observation 1. The right-hand side follows by recalling that in LeadingOnes, it holds for all considered algorithms that $P[\mathcal{A} \text{ starts with fitness } i] = 2^{-i-1}$, and by reordering the sums as in [BDN10].

We can also derive this result from [Doe19a, Corollary 4]. □

Theorem 4. *The expected fixed-target time to reach a target of at least k when optimizing* LeadingOnes *of size n is exactly*

- $\frac{kn}{2}$ *for RLS;*
- $\frac{(1-p)^{1-k}-(1-p)}{2p^2}$ *for the $(1+1)$ EA with mutation probability p;*
- $\frac{(1-p)^{1-k}-(1-p)}{2p^2}(1-(1-p)^n)$ *for the $(1+1)$ $EA_{>0}$ with mutation probability p;*
- $\frac{1}{2} \cdot \sum_{i=0}^{k-1} \frac{1}{p(1-p)^i + \frac{1}{n}(1-p)^n}$ *for the $(1+1)$ $EA_{0\to1}$ with mutation probability p;*
- $\frac{1}{2} C_n^\beta \cdot \sum_{i=0}^{k-1} \frac{1}{\sum_{j=1}^{n/2} j^{-\beta} \frac{j}{n}(1-\frac{j}{n})^i}$ *for the* fast *$(1+1)$ EA_β.*

Proof. We use Lemma 2 and note that, for a fitness of i:

- for RLS, $q_i = 1/n$;
- for the $(1+1)$ EA, $q_i = (1-p)^i \cdot p$;
- for the $(1+1)$ $\text{EA}_{>0}$, $q_i = (1-p)^i \cdot p \cdot (1-(1-p)^n)$;
- for the $(1+1)$ $\text{EA}_{0\to1}$, $q_i = (1-p)^i \cdot p + \frac{1}{n}(1-p)^n$;
- for the *fast* $(1+1)$ EA_β, $q_i = (C_n^\beta)^{-1} \cdot \sum_{j=1}^{n/2} j^{-\beta} \frac{j}{n}(1-\frac{j}{n})^i$.

The sums for the $(1+1)$ EA and the $(1+1)$ $\text{EA}_{>0}$ are simplified as in [BDN10]. □

Note that the presented expression for the *fast* $(1+1)$ EA_β is complicated, and little can be understood from it about its behavior, and in particular about the dependency on β. The simplified upper bound can be obtained by considering only $j = 1$, in which case it is greater by a factor of C_n^β than the corresponding fixed-target time for the $(1+1)$ EA, and this factor is a constant for $1 < \beta < 2$. Lower bounds, however, require much more work, as, for instance, q_i for $i = \Theta(1)$ is $\Theta(n^{1-\beta})$, so the *fast* $(1+1)$ EA_β performs the first improvements much faster than more conventional versions of the $(1+1)$ EA. It is hence quite hard to obtain closed-form sharp bounds, which we leave for possible future work.

5.2.2 Hill Climbers on OneMax, Upper Bounds

We now re-prove the existing results for the $(1+1)$ EA variants on OneMax. Our results for the case of random initialization of an algorithm are sharper than in [VBB+19], because we use the following argument about the weighed sums of harmonic numbers.

Lemma 3. *The following equality holds:*

$$\sum_{i=0}^{n} \frac{\binom{n}{i} H_i}{2^n} = H_n - \sum_{k=1}^{n} \frac{1}{k2^k} = H_n - \ln 2 + O(2^{-n}) = H_{n/2} - \frac{1-o(1)}{2n}.$$

Proof. Proven in [DD16, Sec. 2.5] with [Spi07, Identity 14]. □

Our results are formalized as follows.

Theorem 5. *The expected fixed-target time for Algorithm 1 with $\mu = \lambda = 1$, whose mutation distribution $\mathcal{M}$ selects a single bit to flip with probability q, to reach a target of at least k on* OneMax *of size n is at most:*

- $\frac{1}{q}(H_n - H_{n-k})$ *when initializing with the worst solution;*
- $\frac{1}{q}(H_{n/2} - H_{n-k} - \frac{1-o(1)}{2n})$ *when initializing randomly, assuming $k \geq n/2 + 2\sqrt{n \ln n}$.*

Proof. Let s_i be the probability for the algorithm to be initialized at fitness i. Assuming pessimistically that the fitness does not improve when two and more bits are flipped, we apply Theorem 1 to get the following upper bound:

$$\sum_{i=0}^{k-1} s_i \cdot \sum_{j=i}^{k-1} \frac{1}{q(n-j)} = \frac{1}{q} \sum_{i=0}^{k-1} s_i \cdot (H_{n-i} - H_{n-k}) \leq \frac{H_n - H_{n-k}}{q}.$$

The pessimistic bound above proves the theorem for the algorithms initialized with the worst solution. For the random initialization, we note that the initial search point has a fitness i with the probability of $\binom{n}{i}/2^n$. From the equality above we derive:

$$\begin{aligned}
&\sum_{i=0}^{k-1} \frac{\binom{n}{i}(H_{n-i} - H_{n-k})}{q2^n} \\
&\leq \sum_{i=0}^{n} \frac{\binom{n}{i}(H_{n-i} - H_{n-k})}{q2^n} + \sum_{i=k}^{n} \frac{\binom{n}{i}(H_{n-k} - H_{n-i})}{q2^n} \\
&= \sum_{i=0}^{n} \frac{\binom{n}{i} H_{n-i}}{q2^n} - \sum_{i=0}^{n} \frac{\binom{n}{i} H_{n-k}}{q2^n} + \frac{1}{q} \sum_{i=k}^{n} \frac{\binom{n}{i}(H_{n-k} - H_{n-i})}{2^n} \\
&= \frac{1}{q}\left(H_{n/2} - H_{n-k} - \frac{1-o(1)}{2n}\right) + \frac{1}{q} \sum_{i=k}^{n} \frac{\binom{n}{i}(H_{n-k} - H_{n-i})}{2^n},
\end{aligned}$$

where the last transformation uses Lemma 3. The second addend is $o(1/(qn))$, because $H_{n-k} - H_{n-i} = O(\ln n)$ when $i \geq k$, which is further multiplied by $O(n^{-8/3})$, since for $k \geq n/2 + 2\sqrt{n \ln n}$ it holds that

$$\frac{1}{2^n} \sum_{i=k}^{n} \binom{n}{i} \leq \exp\left(-\frac{16 \frac{\ln n}{n} \frac{n}{2}}{3}\right) = n^{-8/3}$$

by the Chernoff bound. As a result, the fixed-target upper bound for the random initialization is $\frac{1}{q}(H_{n/2} - H_{n-k} - \frac{1-o(1)}{2n})$. □

5.2.3 Hill Climbers on OneMax, Lower Bounds

We also apply fitness-levels to the lower bounds to improve the result from [LS15].

Theorem 6 (Lower fixed-target bounds on $(1+1)$ EA for OneMax)**.** *For mutation probability $p \leq 1/(\sqrt{n} \log n)$, and assuming that $\ell = \lceil n - \min\{n/\log n, 1/(p^2 n \log n)\} \rceil$, the expected time to find a solution with fitness at least $k \geq \ell$ is at least*

$$\left(1 - \frac{18/5}{(1-p)^2 \log n}\right) \frac{H_{n-\ell} - H_{n-k}}{p(1-p)^{n-1}}.$$

Proof. We use the proof of [Sud13, Theorem 9] with Observation 1. The source of this formula is located at [Sud13, journal page 428, bottom of left column]. We use this particular formula, as it gives a good enough precision for a wide range of possible targets and matches the shape of Theorem 5 quite well. □

We conjecture that a similar result can be derived for other versions of the $(1+1)$ EA as well, however, finding the exact constant factors would require additional investigation into [Sud13, Theorem 9].

5.2.4 The $(\mu+1)$ EA on OneMax, Upper Bounds

We now introduce the fixed-target bounds for the $(\mu+1)$ EA using fitness levels. For this we adapt [Wit06, Theorem 2] to support fixed targets different from the optimum, which makes it no longer possible to use certain transitions from the original proof.

Theorem 7. *Let $\mu = poly(n)$, and assume $b = \lfloor n(1-1/\mu) \rfloor$. The expected time to reach an individual with fitness at least $k > 0$ on* OneMax *is:*

$$T_k \leq \mu + \frac{\mu}{(1-p)^n}\left(2k - 1 - (n-k)\ln\frac{n}{n-k+1}\right) + \frac{\mu}{p(1-p)^{n-1}}\begin{cases}\frac{k}{n}, & k \leq b+1,\\ \frac{b+1}{n} + \frac{1}{\mu}(H_{n-b-1} - H_{n-k}) & \text{otherwise.}\end{cases}$$

Proof. Similarly to [Wit06, Theorem 2], we pessimistically assume that on every fitness level L the $(\mu+1)$ EA creates $R = \min\{\mu, n/(n-L)\}$ replicas of the best individual, and then it waits for the fitness improvement. We also assume that the $(\mu+1)$ EA never improves the fitness by more than one.

If there are $i < R$ best individuals, the probability of creating a replica is $(1-p)^n i/\mu$, so the expected time until creating R replicas is at most

$$\frac{\mu}{(1-p)^n}\sum_{i=1}^{R-1}\frac{1}{i} \leq \frac{\mu}{(1-p)^n}\sum_{i=1}^{\frac{L}{n-L}}\frac{1}{i} \leq \frac{\mu}{(1-p)^n}\ln\frac{en}{n-L}$$

and the total time the $(\mu+1)$ EA spends in creating replicas is

$$\begin{aligned} T_r &\leq \sum_{L=0}^{k-1}\frac{\mu}{(1-p)^n}\ln\frac{en}{n-L} = \frac{\mu}{(1-p)^n}\left(k\ln en + \sum_{x=n-k+1}^{n}\ln\frac{1}{x}\right)\\ &\leq \frac{\mu}{(1-p)^n}(k\ln en + k - 1 + (n-k)\ln(n-k+1) - n\ln n)\\ &= \frac{\mu}{(1-p)^n}\left(2k - 1 - (n-k)\ln\frac{n}{n-k+1}\right), \end{aligned}$$

where we use the condition $k > 0$ and that, for all $1 \leq a \leq b$,

$$\sum_{x=a}^{b}\ln\frac{1}{x} \leq \ln\frac{1}{a} + \int_a^b \ln\frac{1}{x}dx = \ln\frac{1}{a} + \left[x + x\ln\frac{1}{x}\right]_a^b.$$

Concerning the fitness gain, if there are R replicas of the best individual with fitness L, the probability of creating new offspring with fitness $L+1$ is at least

$$\frac{R}{\mu}\cdot(n-L)\cdot p(1-p)^{n-1} \geq \frac{\min\{\mu(n-L), n\}}{\mu}\cdot p(1-p)^{n-1},$$

therefore we can apply Theorem 1 to estimate this part of the fixed-target time:

$$T_f \leq \sum_{i=0}^{k-1}\frac{1}{p_i} = \frac{\mu}{p(1-p)^{n-1}}\sum_{i=0}^{k-1}\frac{1}{\min\{\mu(n-i), n\}}.$$

Unlike [Wit06], we consider what the min clauses can be. Depending on how the target k relates to the boundary $b = \lfloor n(1-1/\mu) \rfloor$, we write

$$T_f \leq \begin{cases} \frac{\mu}{np(1-p)^{n-1}} \cdot k, & k \leq b+1 \\ \frac{\mu}{np(1-p)^{n-1}} \cdot (b+1) + \frac{H_{n-b-1}-H_{n-k}}{p(1-p)^{n-1}}, & k > b+1 \end{cases}$$

which completes the proof, noting that the fixed-target time is $\mu + T_r + T_f$. □

6 Drift Analysis

In this section we consider the drift theorems in the fixed-target context. These theorems are generally more powerful, but it appears that one should use them for proving fixed-target results with slightly more care than in the case of fitness levels. We also propose variations of variable and multiplicative drift theorems that are specifically aimed at gaining precision in the fixed-target context.

6.1 Drift Theorems

Drift theorems translate bounds on step-wise expected progress into bounds on expected first-hitting runtimes. They are usually formulated in terms of a random process that needs to hit a certain minimum value. To this end, the search process of the algorithm is mapped to real numbers via so-called potential functions. Some of these theorems prohibit the process from falling below the target, or from visiting an interval between a target and the next greater value. For this reason, some optimization time results cannot be converted into the fixed-target results without additional work, as targets different from the optimum violate the requirements above.

The paper [KK19] contains a discussion of processes which may fall below the target, and the implications for drift theorems. For instance, [KK19, Example 6] gives an example of a process with the target 0 and the expected progress of 1 at $X_t = 1$, which is given by $X_{t+1} = -n+1$ with probability of $1/n$ and $X_{t+1} = 1$ otherwise. By mistakenly applying a well-known additive drift theorem from [HY01] to this process, one can get an overly optimistic upper bound of 1 on the expected runtime, which is, in fact, n.

We start the discussion with the additive drift theorems. We provide their versions from [KK19] which appear to be preadapted to the fixed-target conditions. For the first of these theorems we explicitly note that its upper bound is not $(X_0-k)/\delta$, but a (generally) larger value. Indeed, if we define X_T to be the value of the process at the hitting time T, it is only known that $E[X_T \mid X_0] \leq k$, and the latter may be far from being an equality. Proving the bounds for $E[X_T \mid X_0]$ seems to be the essential additional work in order to prove fixed-target results.

Theorem 8 (Additive drift, fixed-target upper bounds; Theorem 7 from [KK19], original version in [HY01])**.** *Let k be the target value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t \leq k\}$. Suppose that there is some value $\delta > 0$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq \delta$. Then $E[T \mid X_0] \leq (X_0 - E[X_T \mid X_0])/\delta$.*

Theorem 9 (Additive drift, fixed-target lower bounds; Theorem 8 from [KK19])**.** *Let k be the target value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t \leq k\}$. Suppose that:*

- *there is some value $\delta > 0$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \dots, X_t] \leq \delta$;*
- *there is some value $c \geq 0$ such that, for all $t < T$, it holds that $E[|X_{t+1} - X_t| \mid X_0, \dots, X_t] \leq c$.*

Then $E[T \mid X_0] \geq (X_0 - E[X_T \mid X_0])/\delta \geq (X_0 - k)/\delta$.

For the lower bound, the simplification to $(X_0 - k)/\delta$ is possible (contrary to the upper bound), but having a better bound may be desirable. By recalling again [KK19, Example 6], we can see that taking $E[X_T \mid X_0]$ into account can improve the bound asymptotically. In fixed-target runtime analysis, such an improvement may occur with very easy targets.

More advanced drift theorems such as the multiplicative drift theorems [DJW12] and the variable drift theorems [MRC09, Joh10, DFW11, LW21] make it easier to prove sharp bounds on hitting times for processes with drift that depends on the current value, a situation that occurs rather frequently in evolutionary computation. Most of the popular drift theorems of this sort can be classified using the following properties: they estimate the time for a process to either reach a certain target value k or to surpass a certain threshold value k', and they also may or may not require the process to never fall below the target or to never visit a region between the threshold and the ultimate termination state (usually zero).

The case analysis of four variants of drift theorems presented in [KK19] revealed that only two of these four variants are suitable to be used for fixed-target research: theorems for upper bounds which require to surpass a threshold k', and theorems for lower bounds which require to reach a target k. The former contain an extra addend in their statement (such as "1+" or "$x_{\min}/h(x_{\min})$"), while the latter do not. This seems to be closely related to the $E[X_T \mid X_0]$ issue in additive drift theorems, which the "good" theorems pessimize to the right direction.

We now present or reformulate certain drift theorems for upper bounds, which can be used out of the box for proving the fixed-target results.

Theorem 10 (Multiplicative drift, upper bounds [DJW12] adapted to fixed-target settings)**.** *Let k' be the threshold value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t < k'\}$. Furthermore, suppose that:*

- *$X_0 \geq k'$, and for all $t \leq T$, it holds that $X_t \geq 0$;*
- *there is some value $\delta > 0$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \dots, X_t] \geq \delta X_t$.*

Then $E[T \mid X_0] \leq (1 + \ln(X_0/k'))/\delta$.

Proof. Apply [DJW12] to the new potential function $X'_t := X_t/k'$. □

Theorem 11 (Variable drift, fixed-target upper bounds; Theorem 10 from [KK19])**.** *Let k' be the threshold value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t < k'\}$. Furthermore, suppose that:*

- $X_0 \geq k'$*;*
- *for all $t \leq T$, it holds that $X_t \geq 0$;*
- *there is a monotonically increasing function $h : [k'; \infty) \to \mathbb{R}^+$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq h(X_t)$.*

Then $E[T \mid X_0] \leq \frac{k'}{h(k')} + \int_{k'}^{X_0} \frac{1}{h(z)} d(z)$.

In all these theorems, the conditions are the same as for their corresponding optimization-time variants. We now present a drop-in replacement version of [DFW11, Theorem 7] that enables obtaining lower bounds on fixed-target hitting times by using the same functions and conditions that were used for proving the lower bounds on the optimization time using that theorem. Note that, for technical reasons, this version does not support setting the target to the optimum value.

Theorem 12 (Variable drift, fixed-target lower bounds; adapted from Theorem 7 from [DFW11])**.** *Let $k > 0$ be the target value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of monotonically decreasing random variables over $\mathbb{R}_0^+$, and let $T = \inf\{t \mid X_t \leq k\}$. Suppose that there are two continuous monotonically increasing functions $c, h : \mathbb{R}_0^+ \to \mathbb{R}^+$, and that for all $t < T$ it holds that*

- $X_{t+1} \geq c(X_t)$;
- $E[X_t - X_{t+1} \mid X_t] \leq h(c(X_t))$.

Then $E[T \mid X_0] \geq \int_k^{X_0} \frac{1}{h(z)} d(z)$.

Proof. We proceed similarly to how it was done in the proof of [DFW11, Theorem 7], however, with a different additive drift theorem in mind.

Let $g : \mathbb{R} \to \mathbb{R}$ be the function defined by

$$g(z) = \begin{cases} \frac{z-k}{h(k)} & \text{if } z < k; \\ \int_k^z \frac{1}{h(x)} dx & \text{if } z \geq k. \end{cases}$$

This function is strictly monotone increasing and continuous on $\mathbb{R}$. Moreover, it is differentiable on $\mathbb{R}$ with

$$g'(z) = \begin{cases} \frac{1}{h(k)} & \text{if } z < k; \\ \frac{1}{h(z)} & \text{if } z \geq k. \end{cases}$$

Using $x := X_t$ and $y := X_{t+1}$, the monotonicity of the sequence $(X_t)_{t\in\mathbb{N}}$ and the condition $X_{t+1} \geq c(X_t)$, we get that $c(x) \leq y \leq x$. According to the mean-value theorem, for all $x \geq k$ there exists $\xi \in (y, x)$ such that

$$g'(\xi) = \frac{g(x) - g(y)}{x - y}$$

which implies

$$\frac{g(x)-g(y)}{x-y} \leq g'(y) \leq g'(c(x)) = \frac{1}{h(c(x))}$$

since g' decreases monotonically.

In terms of the function g, the hitting time T describes the smallest t with $g(X_t) \leq 0$. As g is monotone and invertible, it holds for all $t < T$ that

$$E[g(X_t) - g(X_{t+1}) \mid g(X_t)] \leq E\left[\frac{X_t - X_{t-1}}{h(c(X_t))} \mid X_t\right] \leq 1,$$

where the last inequality comes from the theorem condition $E[X_t - X_{t+1} \mid X_t] \leq h(c(X_t))$. We apply Theorem 9 for $g(X_t)$ and the zero target to complete the proof. □

6.2 Overshoot-Aware Drift Theorems

We have already shown, using additive drift as an example, that in order to obtain good fixed-target bounds, an expected overshoot value $E[X_T \mid X_0]$ needs to be considered. However, more advanced theorems cannot easily make an advantage out of this idea, although it may be desired (especially when analyzing easy targets, or in the case when the hardest part of the optimization process is not near the optimum, as it may happen for the $(\mu+1)$ genetic algorithm on certain monotone functions [LZ19]). Here, we present variations of the variable drift and multiplicative drift theorems for upper bounds which explicitly contain the overshoot term $E[X_T \mid X_0]$ in the expression for the hitting time, for which reason we call them *overshoot-aware*.

Theorem 13 (Overshoot-aware variable drift, upper bounds)**.** *Let k' be the threshold value, let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t < k'\}$. Furthermore, suppose that*

- *$X_0 \geq k'$;*
- *there is a non-decreasing function $h : [k';\infty) \to \mathbb{R}^+$ such that for all $t < T$ it holds that $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq h(X_t)$.*

Then $E[T \mid X_0] \leq \frac{k'-E[X_T|X_0]}{h(k')} + \int_{k'}^{X_0} \frac{dz}{h(z)}$.

Proof. Let $D = [k';\infty)$. We define the potential function $g : D \to \mathbb{R}$ by setting

$$g(x) = \int_{k'}^{x} \frac{dz}{h(\max\{z, k'\})}.$$

For any two points $x \geq y$, such that $x, y \in D$ and $x \geq k'$, the following holds:

$$g(x) - g(y) = \begin{cases} \int_y^x \frac{dz}{h(z)} \geq \frac{x-y}{h(x)}, & y \geq k' \\ \frac{k'-y}{h(k')} + \int_{k'}^x \frac{dz}{h(z)} \geq \frac{k'-y}{h(x)} + \frac{x-k'}{h(x)} = \frac{x-y}{h(x)}, & y \leq k'. \end{cases}$$

As a result, the expected potential decrease is:

$$E[g(X_t) - g(X_{t+1}) \mid X_0, \ldots, X_t] \geq \frac{E[X_t - X_{t+1} \mid X_0, \ldots, X_t]}{h(X_t)} \geq 1.$$

We also note that $g(x)$ is a linear function for $x \in [0; k']$, hence for any random variable Z taking values from $[0, k']$ it holds that $E[g(Z)] = g(E[Z])$.

Now we apply Theorem 8 and prove this theorem:

$$\begin{aligned} E[T \mid g(X_0)] &\leq g(X_0) - E[g(X_T) \mid g(X_0)] \\ &= g(X_0) - g(E[X_T \mid X_0]) \\ &= g(X_0) - g(k') + g(k') - g(E[X_T \mid X_0]) \\ &= \int_{k'}^{X_0} \frac{dz}{h(z)} + \frac{k' - E[X_T \mid X_0]}{h(k')}, \end{aligned}$$

where we use the fact that the random variable X_T takes values from $[0; k']$ together with the observation just above, and g is a bijection, so conditioning on $g(X_0)$ is equivalent to conditioning on X_0. □

Corollary 1 (Overshoot-aware multiplicative drift, upper bounds)**.** *Let $D \subseteq \{0\} \cup \mathbb{R}^+$ be the set of possible values of the random process that is defined by the sequence of random variables $(X_t)_{t \in \mathbb{N}}$. Let $k \in D$ be the target value of the process, let $T = \inf\{t \mid X_t \leq k\}$ be the hitting time, and let $k' = \inf\{x \mid x \in D, x > k\}$ be the lower bound on the values from D that are larger than k. Suppose that*

- *$X_0 \geq k'$;*
- *there is some constant value $\delta > 0$ such that, for all $t < T$, it holds that $X_t - E[X_{t+1} \mid X_0, \ldots, X_t] \geq \delta X_t$.*

Then $E[T \mid X_0] \leq \frac{1}{\delta} \cdot \left(1 - \frac{E[X_T \mid X_0]}{k'} + \ln \frac{X_0}{k'}\right)$.

Proof. Follows directly from Theorem 13 by choosing $h(x) = \delta x$. □

Note that a proper estimation of the expected overshoot $E[X_T \mid X_0]$ may in fact be tricky, especially given that it is a conditional expectation. Contrary to Markovian processes, where $E[X_T \mid X_0] = E[X_T \mid X_{T-1}]$ and understanding X_{T-1} may be rather easy, for more involved processes, such as optimization of multimodal functions or algorithms with self-adjusting parameters, this value may depend on the whole sequence of values.

6.3 Applications

6.3.1 Minimum Spanning Trees

We begin with fixed-target bounds for minimum spanning trees solved by the $(1 + 1)$ EA and its variations. In the context of evolutionary computation, the function to optimize can be defined in different ways. We follow [NW07] and use a function which consists of two parts: the number of connected components with a large weight (to facilitate connecting all the graph vertices), and the weight of the chosen edges. This function is to be minimized. It is known [NW07, DJW12] that the $(1 + 1)$ EA optimizes this function in expected time $O(m^2(\log n w_{\max}))$, where m is the number of edges, n is the number of vertices, and $w_{\max}$ is the maximum edge weight.

Theorem 14. *Starting from a randomly initialized graph, the expected time for Algorithm 1 with $\mu = \lambda = 1$, whose mutation distribution $\mathcal{M}$ selects a single bit to flip with probability q, to find a graph with at most k connected components is at most $\frac{1}{q}(1+\ln\frac{m-1}{k})$.*

Proof. Consider the potential function $g(x) = s - 1$, where s is the number of connected components in the subgraph which consists of the edges included in the genotype x. If there are s such components, there are at least $s-1$ edges, which can be added to decrease the number of components. To do that, it is enough to flip at least one bit corresponding to these edges. To apply Theorem 10, we estimate the drift as $E[g(X_t) - g(X_{t+1}) \mid g(X_t) = c] \geq cq$.

The target of k connected components maps to the target potential of $k-1$ and hence to the threshold value k. By applying Theorem 10 we get the desired bound. □

Theorem 15. *Starting from some spanning tree, the expected time for Algorithm 1 with $\mu = \lambda = 1$, whose mutation distribution $\mathcal{M}$ selects exactly two bits to flip with probability q, to find spanning tree with the weight at most k larger than the minimum possible weight is at most $\frac{1}{q}(1 + \ln\frac{(n-1)w_{\max}}{k+1})$.*

Proof. We again reuse the corresponding result from [DJW12]. The process is defined as $X_t = w(x) - w_{\text{opt}}$, and [DJW12] gives the drift bound of $E[X_t - X_{t+1} \mid X_t = x] \geq X_t \cdot q$. The application of Theorem 10 yields the desired upper bound on the fixed-target runtime, as $X_0 \leq (m-1)w_{\max}$. □

We give the two-bit probabilities for the common algorithms.

- RLS which flips pairs of bits ("2-opt mutation operator"): $q = \frac{2}{m(m-1)}$;
- $(1+1)$ EA and $(1+1)$ $\text{EA}_{0\to1}$: $q = p^2(1-p)^{m-2}$;
- $(1+1)$ $\text{EA}_{>0}$: $q = \frac{p^2(1-p)^{m-2}}{1-(1-p)^m}$;
- $(1+1)$ $\text{EA}_{0\to2}$: $q = p^2(1-p)^{m-2} + (1-p)^m\frac{2}{m(m-1)}$;
- *fast* $(1+1)$ EA_β: $q = \frac{2}{m(m-1)} \cdot (C_n^\beta)^{-1} \cdot 2^{-\beta} \cdot \Theta(1)$.

Note that RLS which flips pairs of bits is a rather a mind experiment than an algorithm to use, however, one may use RLS that tosses a coin and flips either single bits or pairs of bits, which just halves the probability above.

6.3.2 The $(1+1)$ EA on OneMax, Lower Bounds

We prove the lower fixed-target bounds using variable drift.

Theorem 16 (The lower fixed-target bound on $(1+1)$ EA with $p = 1/n$ for OneMax)**.** *The expected time to find an individual with fitness at least k, $2n/3 < k < n$, when n is large enough, is at least*

$$\left(1 - O\left(\frac{n\log n}{n^3}\right)\right) en\left(2\ln\frac{\sqrt{\frac{n}{3}}+2}{\sqrt{n-k}+2} + \ln\frac{n+16(n-k)+32\sqrt{n-k}}{n+16\frac{n}{3}+32\sqrt{\frac{n}{3}}}\right).$$

Proof. The basis of this proof is [DFW11, Theorem 5]. Our aim is to apply Theorem 12, which allows jumps below the target. This allows us to use the original the potential function $X_t = n - f(x)$ and the existing bound on the expected drift [DFW11, Lemma 6]: $E[X_t - X_{t+1} \mid X_t = s] \leq \frac{s}{en}(1 + \frac{16s}{n})$.

Following [DFW11], we bound the step size with $c(x) = x - \sqrt{x}$. We denote as a bad step the event of increasing the fitness by more than $\sqrt{x}$. To condition on that, we estimate the probability of making a bad step for $2 \leq x \leq n$, which was shown in [DFW11] to be $O(n^{-3})$ for $x \geq 9$. For a good fixed-target result, we need to cover the rest. For $5 \leq x \leq 8$, the probability of a bad step is at most $\binom{8}{3}p^3 = \frac{56}{n^3} = O(n^{-3})$. For $2 \leq x \leq 4$, a similar calculation yields the probability of $O(n^{-2})$. Since the latter bound corresponds to only $\Theta(1)$ fitness values, in which the algorithm spends at most $O(n)$ iterations in expectation, the union bound over all bad steps during $en \ln n$ iterations is at most $O((n \log n)/n^3)$. This is reflected by having the $1 - O((n \log n)/n^3)$ quotient in the result.

We also reuse the function $h(x)$ from [DFW11, Theorem 5], which is $h(x) \leq \frac{x+2\sqrt{x}}{en}(1 + \frac{16+32\sqrt{x}}{n})$, and apply Theorem 12 to get

$$\begin{aligned} E[T \mid X_0] &\geq \int_{k'}^{X_0} \frac{en}{(x + 2\sqrt{x})(1 + \frac{16+32\sqrt{x}}{n})} dx \\ &\geq en \cdot \left[2 \ln \left(\sqrt{x} + 2 \right) - \ln \left(n + 16x + 32\sqrt{x} \right) + \frac{8 \arctan \frac{4\sqrt{x}+4}{\sqrt{n-16}}}{\sqrt{n-16}} \right]_{k'.}^{X_0} \end{aligned}$$

We simplify the expression above by dropping the addend containing the arctangent, since it increases with x and its value is asymptotically smaller than other addends. Finally we choose $k' = n - k$ and bound X_0 from below by $n/3$. The latter choice is due to Chernoff bounds, which show that the algorithm starts with an individual with a Hamming distance to the optimum of at least $n/3$ with probability $1 - e^{-\Omega(n)}$, which hides in the leading factor of $1 - O((n \log n)/n^3)$. □

Figure 2 illustrates that the bound proven in Theorem 16 is significantly better than the one from [LS15] and captures the essense of the algorithm's behaviour.

7 Summary of Difficulties of Fixed-Target Analysis

In this work, we have often seen that fixed-target analyses are not much more difficult than classic runtime analyses. However, from the section dedicated to drift theorems we also learned that, in order to obtain fixed-target results from existing optimization time results, more powerful drift theorems sometimes require additional statements to yield bounds that are good enough. More extreme examples are known, such as our earlier attempt with the BinVal function in [VBB+19], for which a radically different approach had to be developed. What is more, sometimes even the easy proofs employing fitness levels required us to dig deeper into the original optimization time result and refer not to the main theorem formulation, but to the details of its proof. Below we present an attempt to classify the possible difficulties that may arise when one wants to prove fixed-target results based on the existing results about optimization times.

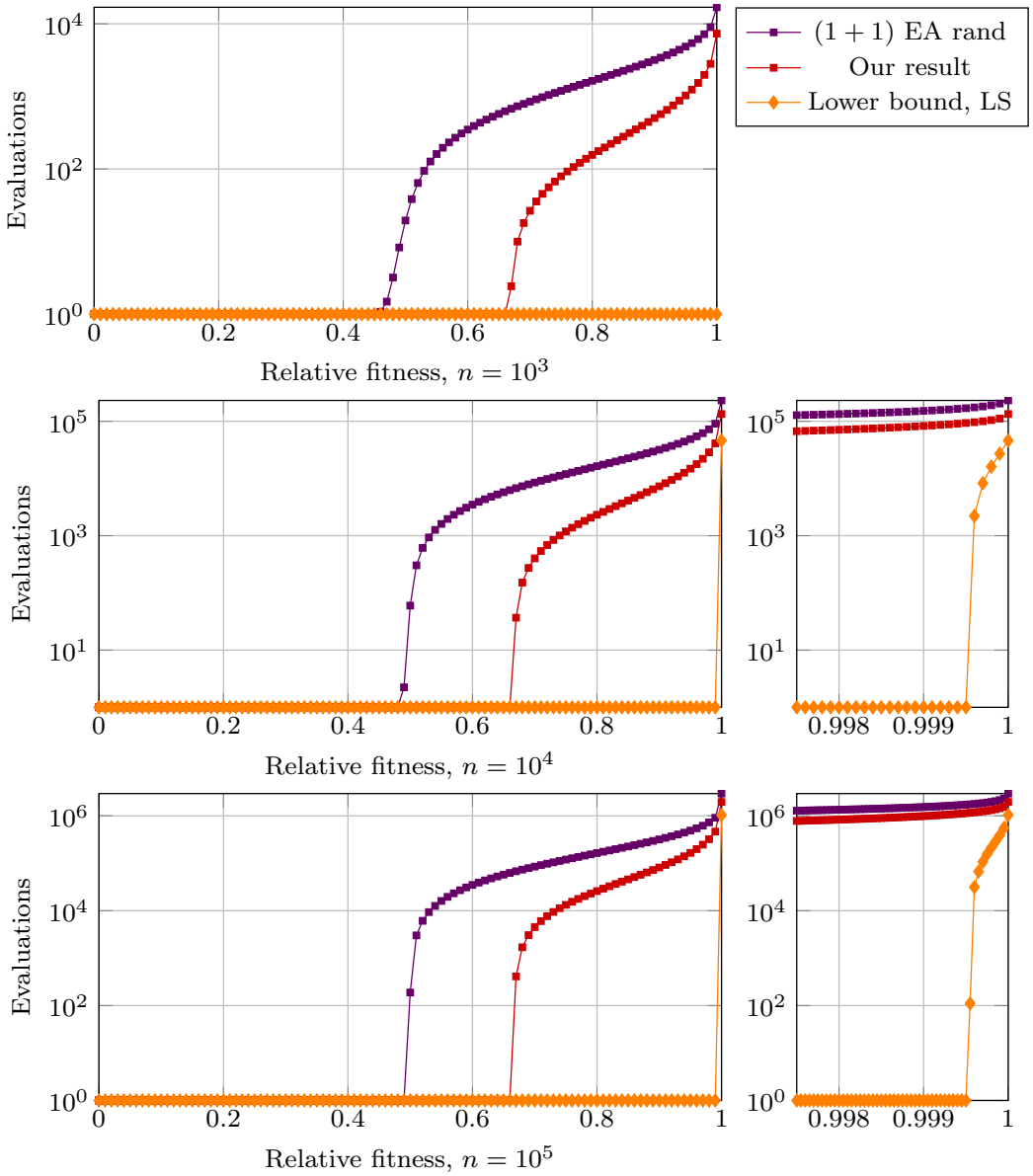


Figure 2: The runtime profile of the $(1+1)$ EA on OneMax and the lower bounds, the existing and the new ones. Plots in the left part of the figure show the general picture for three problem sizes $n \in \{10^3, 10^4, 10^5\}$, whereas plots on the right zoom into the regions close to the optimum. The plotted quantity is the expected number of fitness evaluations required to reach the target fitness value against that (relative) fitness value. The runtime profiles of the $(1+1)$ EA were computed exactly using dynamic programming following the ideas of [BD19]. For the lower bounds, the respective expressions were evaluated and plotted, disregarding the $(1 \pm o(1))$ factors. The notation in the legend is as follows: "$(1+1)$ EA rand" is the $(1+1)$ EA starting at a randomly generated bit string, "Our result" is the one proven in Theorem 16, and "Lower bound, LS" is the bound from [LS15, Corollary 3].

(1) *Intermediate results can directly be reused, but final theorems are not enough.* Since many of the existing papers aim at proving optimization times, they may not expose the

intermediate results, which can be used in fixed-target proofs, as separate citable units, such as theorems or lemmas. A simple example is the proof of Theorem 6, where we could not even point to the equation we used due to it being unnumbered.

It may even happen that the proof nearly gives the fixed-target result, but it cannot be derived from the theorem statement. The typical reason is that the result gets simplified in a way that does not change the optimization time much but affects the possible fixed-target result. Our proof of Theorem 7 diverged from its source at the point of simplification of one of the $\min\{\ldots\}$ clauses: if we followed this simplification too, the final result would be much less precise.

Presenting the results as a set of smaller lemmas and theorems may make it easier to build fixed-target results (and further extensions) atop of them. We admit, however, that it requires more work, and some proofs may be too difficult to split into multiple lemmas, which would then have large and cumbersome statements.

(2) *Results can be directly reused, but they are not applicable or sharp enough for the full range of targets.* For various reasons the existing optimization time results may be quite sharp on their own, but appear to be too loose in fixed-target contexts. A good example of such a situation is a bound which is sharp for hard regions of the search space, but is loose for the remaining part. This is the case for all simple upper bounds for OneMax, which assume that only one-bit flips are beneficial. A similar example for lower bounds is [LS15, Corollary 3], which implicitly assumes that optimization in easy parts is performed instantly. Such existing results need to be augmented by more work that refines them in parts that are not too meaningful for finding optimization times, but essential for good fixed-target results.

It follows that, surprisingly at first sight, more complicated proofs of optimization times that yield sharper bounds should be easier to adapt to the fixed-target context than easier proofs. We admit, however, that such proofs are harder to produce and verify, and the main messages important for understanding optimization times may get obscured.

(3) *Some results can be directly reused, but additional statements need to be proven.* One of such cases observed in this paper are drift theorems, which can borrow all the conditions from the existing optimization time results, but require to prove the bounds on the overshoot term $E[X_T \mid X_0]$ in order to yield the best possible precision. This is not actually required at times, however, on some occasions not doing that results in a ridiculously imprecise bound, such as an upper bound that does not take the target into account.

This can be seen as a fair price to pay, since drift theorems are applicable for a wider range of processes than fitness level theorems, and harder problems may demand more complicated solutions. Based on some of our preliminary work, however, we conjecture that if one has both upper and lower bounds that are based on drift theorems and are good enough, the matching fixed-target results may be obtained with much less effort.

(4) *Existing results require more work in order to be usable.* In the most complicated cases, the existing optimization time proofs may be too overfitted to the optimum being the target. For instance, it may be the case that the distance between the optimum and the target, which used to be zero, appears in so many locations of the proof, that the whole proof needs to be significantly reworked and extended to produce the fixed-target results. This is what seems be the case for linear functions: some of our work

in [VBB+19] for BinVal, especially the failed attempts that were not included into that paper, suggests that the principles of designing the potential function in [Wit13] shall be significantly changed to work well in the fixed-target context.

While two latter points seem to be fundamental, two former appear to follow from the current habits of performing and presenting the research, for which reason they may be called the *social* ones: the use of best practices, possibly assisted by automated theorem provers, has the potential of resolving a large fraction of these difficulties just as a side effect.

8 Conclusion

In this first work focussed on fixed-target runtime analysis, most of our results indicate that deriving fixed-target results for the whole set of reasonable targets is in several cases not more complicated than just analyzing the classical optimization time (which is the special case where the target is set to the fitness of an optimal solution). Since fixed-target results are much more informative than the optimization time alone, in such situations we can only advocate to conduct runtime analyses in the more general fixed-target perspective. As discussed, this often does not need different proofs, all that is required is to formulate the information present in the proof also in the result. That said, there are also problems which require additional facts to be proven in order to obtain sharp enough fixed-target results. Extending our current toolbox to analyze these situations should be a fruitful direction for further research.

Together with a complementary fine-grained notion that appeared recently, optimization times starting from a good solution [ABD20], which might also be called *fixed-start runtime analysis* to unify terms, fixed-target runtime analysis can help to prove runtimes for algorithms to cross fitness intervals. This has direct applications in designing and understanding hyper-heuristics, as well as in parameter tuning and parameter control. Since these research areas tend to require especially sharp bounds, it may be the case that the difficulties in applying even the hardest tools that take care of fine effects, such as target overshooting, may actually pay off well.

Acknowledgments

This was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, and by RFBR and CNRS, project number 20-51-15009.

References

[ABD20] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. First steps towards a runtime analysis when starting with a good solution. In *Parallel Problem Solving from Nature, PPSN 2020*, pages 560–573. Springer, 2020.

[BD19] Nathan Buskulic and Carola Doerr. Maximizing drift is not optimal for solving OneMax. In *Proceedings of the Genetic and Evolutionary Computation*

Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019, pages 425–426, 2019.

[BDDV20] Maxim Buzdalov, Benjamin Doerr, Carola Doerr, and Dmitry Vinokurov. Fixed-target runtime analysis. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1295–1303. ACM, 2020.

[BDN10] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 1–10. Springer, 2010.

[CD18] Eduardo Carvalho Pinto and Carola Doerr. Towards a more practice-aware runtime analysis of evolutionary algorithms. `https://arxiv.org/abs/1812.00493`, 2018.

[CDEL18] Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Level-based analysis of genetic algorithms and other search processes. *IEEE Transactions on Evolutionary Computation*, 22:707–719, 2018.

[DD16] Benjamin Doerr and Carola Doerr. The impact of random initialization on the runtime of randomized search heuristics. *Algorithmica*, 75:529–553, 2016.

[DDE15] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[DDY16a] Benjamin Doerr, Carola Doerr, and Jing Yang. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Parallel Problem Solving from Nature, PPSN 2016*, pages 824–834. Springer, 2016.

[DDY16b] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 1123–1130. ACM, 2016.

[DDY20] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science*, 801:1–34, 2020.

[DFW11] Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Sharp bounds by probability-generating functions and variable drift. In *Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 2083–2090. ACM, 2011.

[DG10] Benjamin Doerr and L.A. Goldberg. Drift analysis with tail bounds. In *Parallel Problem Solving from Nature, PPSN 2010*, volume 6238 of *Lecture Notes in Computer Science*, pages 174–183. Springer, 2010.

[DJW12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.

[DJWZ13] Benjamin Doerr, Thomas Jansen, Carsten Witt, and Christine Zarges. A method to derive fixed budget results from expected optimisation times. In *Genetic and Evolutionary Computation Conference, GECCO 2013*, pages 1581–1588. ACM, 2013.

[DK19] Benjamin Doerr and Timo Kötzing. Multiplicative up-drift. In *Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 1470–1478. ACM, 2019.

[DK21] Benjamin Doerr and Timo Kötzing. Lower bounds from fitness levels made easy. In *Genetic and Evolutionary Computation Conference, GECCO 2021*, pages 1142–1150. ACM, 2021.

[DL17] Carola Doerr and Johannes Lengler. OneMax in black-box models with several restrictions. *Algorithmica*, 78:610–640, 2017.

[DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 777–784. ACM, 2017.

[DN20] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020. Also available at `https://cs.adelaide.edu.au/~frank/papers/TheoryBook2019-selfarchived.pdf`.

[Doe19a] Benjamin Doerr. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science*, 773:115–137, 2019.

[Doe19b] Benjamin Doerr. A tight runtime analysis for the cGA on jump functions: EDAs can cross fitness valleys at no extra cost. In *Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 1488–1496. ACM, 2019.

[DWY+18] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. `https://arxiv.org/abs/1810.05281`, 2018. IOHprofiler is available at `https://github.com/IOHprofiler`.

[DYvR+18] Carola Doerr, Furong Ye, Sander van Rijn, Hao Wang, and Thomas Bäck. Towards a theory-guided benchmarking suite for discrete black-box optimization heuristics: profiling $(1 + \lambda)$ EA variants on OneMax and LeadingOnes. In *Genetic and Evolutionary Computation Conference*, pages 951–958. ACM, 2018.

[HAR+20] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: a platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, pages 1–31, 2020.

[HJZ19] Jun He, Thomas Jansen, and Christine Zarges. Unlimited budget analysis. In *Genetic and Evolutionary Computation Conference Companion, GECCO 2019*, pages 427–428, 2019.

[HY01] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:51–81, 2001.

[Jan13] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer, 2013.

[Joh10] Daniel Johannsen. *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, 2010.

[JW06] Thomas Jansen and Ingo Wegener. On the analysis of a dynamic evolutionary algorithm. *Journal of Discrete Algorithms*, 4:181–199, 2006.

[JZ14a] Thomas Jansen and Christine Zarges. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, 545:39–58, 2014.

[JZ14b] Thomas Jansen and Christine Zarges. Reevaluating immune-inspired hypermutations using the fixed budget perspective. *IEEE Transactions on Evolutionary Computation*, 18:674–688, 2014.

[KK19] Timo Kötzing and Martin Krejca. First-hitting times under drift. *Theoretical Computer Science*, 796:51–69, 2019.

[Köt16] Timo Kötzing. Concentration of first hitting times under additive drift. *Algorithmica*, 75:490–506, 2016.

[KW20] Timo Kötzing and Carsten Witt. Improved fixed-budget results via drift analysis. In *Parallel Problem Solving from Nature, PPSN 2020*, pages 648–660. Springer, 2020.

[Len20] Johannes Lengler. Drift analysis. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 89–131. Springer, 2020. Also available at `https://arxiv.org/abs/1712.00964`.

[LS15] Johannes Lengler and Nicholas Spooner. Fixed budget performance of the (1+1) EA on linear functions. In *Foundations of Genetic Algorithms, FOGA 2015*, pages 52–61. ACM, 2015.

[LS20] Per Kristian Lehre and Dirk Sudholt. Parallel black-box complexity with tail bounds. *IEEE Transactions on Evolutionary Computation*, 24:1010–1024, 2020.

[LW21] P. K. Lehre and C. Witt. Tail bounds on hitting times of randomized search heuristics using variable drift analysis. *Combinatorics, Probability and Computing*, 30(4):550–569, 2021.

[LZ19] Johannes Lengler and Xun Zou. Exponential slowdown for larger populations: the $(\mu + 1)$-EA on monotone functions. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 87–101. ACM, 2019.

[MRC09] Boris Mitavskiy, Jonathan E. Rowe, and Chris Cannings. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *International Journal on Intelligent Computing and Cybernetics*, 2:243–284, 2009.

[NNS17] Samadhi Nallaperuma, Frank Neumann, and Dirk Sudholt. Expected fitness gains of randomized search heuristics for the traveling salesperson problem. *Evolutionary Computation*, 25(4):673–705, 2017.

[NW07] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378:32–40, 2007.

[Spi07] Michael Z. Spivey. Combinatorial sums and finite differences. *Discrete Mathematics*, 307:3130–3146, 2007.

[Sud13] Dirk Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435, 2013.

[VBB+19] Dmitry Vinokurov, Maxim Buzdalov, Arina Buzdalova, Benjamin Doerr, and Carola Doerr. Fixed-target runtime analysis of the (1+1) EA with resampling. In *Genetic and Evolutionary Computation Conference Companion, GECCO 2019*, pages 2068–2071, 2019.

[Weg02] Ingo Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In Ruhul Sarker, Masoud Mohammadian, and Xin Yao, editors, *Evolutionary Optimization*, pages 349–369. Kluwer, 2002.

[Wit06] Carsten Witt. Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14:65–86, 2006.

[Wit13] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22:294–318, 2013.

[Wit14] Carsten Witt. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters*, 114:38–41, 2014.