Fixed-Target Runtime Analysis

Maxim Buzdalov, Benjamin Doerr, Carola Doerr, Dmitry Vinokurov

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

April 22, 2020

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 a theoretical viewpoint, more fine-grained performance measures are needed. Two complementary approaches have been suggested: fixed-budget analysis and fixed-target analysis.

In this work, we conduct an in-depth study on the advantages and 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 the existing runtime result to a fixed-target result is highly non-trivial.

1 Introduction

The classic performance measure in the theory of evolutionary computation [Jan13, DN20] is the 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 drift analysis [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

[*]This is a slightly extended version of the paper accepted to GECCO 2020.

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 then is more powerful. 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 [HAM$^+$16, DWY$^+$18]. These fine-grained performance analyses have helped to advance our understanding of evolutionary computation methods, and have contributed significantly to the 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 used to find a solution of at least this quality.

The main goal of this work is a systematic study of 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 often fixed-target results come almost for free when one can prove a result on the optimization time, so it is a waste to not use this opportunity. 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.

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

The notion to first become the object of explicit scientific attention is *fixed-budget analysis* [JZ14a]. Fixed-budget analysis asks for, given a computational budget $b \in \mathbb{N}$, what is 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 ($(1+1)$ 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, which shows again that fine-grained runtime results can lead to insights not available from optimization times.

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. For the $(1+1)$ EA, which can gain more than one fitness level in one iteration, the analyses were even more complicated 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.

The first fixed budget analysis for a combinatorial optimization problem was conducted in [NNS14]. 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, and this even more when 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 $a \in \mathbb{R}$, the fixed-target runtime $T_{A,f}(a)$ was defined as the number of the first fitness evaluation which gives a fitness of a or larger. Since this notion was merely used as tool in a proof, the name fixed-target runtime was not used yet. Apparently unaware of the fact that the fixed-target notion was used already in [DJWZ13, LS15], [PD18] re-invented it under the name *runtime profiles* and it was advocated to use this notion as it gives 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.

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 result 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, BLS14, DDE15, DDY16b] 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 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 $\{a, a+1, \ldots, b-1, b\}$, and we denote the set $[1..n]$ as $[n]$. We write H_n for the n-th harmonic number, that is, $H_n = \sum_{i=1}^{n} \frac{1}{i}$, and $H_0 = 0$ for convenience.

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

```
Require: mutation 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 $[X]$, $\ell \leftarrow$ sample from $\mathcal{M}$
          $y_i \leftarrow$ flip $\ell$ 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
```

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 [PD18], 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 worst individual. 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 $(1 + 1)$ EA$_{0\to1}$ using the *shift mutation strategy*:
  $\mathcal{M} = \max\{1, B(n, p)\}$;
- the $(1 + 1)$ EA$_{>0}$ using the *resampling mutation strategy*:
  $\mathcal{M} = [x \sim B(n, p) \mid x > 0]$.

Sometimes we are only interested in the probability q of flipping exactly one given bit. For a problem size n and mutation strength p, the values of q for the algorithms above are

- RLS: $q = \frac{1}{n}$;
- $(1 + 1)$ EA: $q = p(1 - p)^{n-1}$;
- $(1 + 1)$ EA$_{0\to1}$: $q = p(1 - p)^{n-1} + \frac{(1-p)^n}{n}$;
- $(1 + 1)$ EA$_{>0}$: $q = \frac{p(1-p)^{n-1}}{1-(1-p)^n}$.

We consider the following classical problems on bit strings:

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

We also consider the minimum spanning tree (MST) problem. The problem is, given a connected undirected graph G with positive weights on each edge, to find a minimum spanning tree of it, that is, a subgraph that contains all vertices of G and 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 list the known fixed-target results. In the whole section, n is the problem size and k is the target fitness.

4.1 LeadingOnes

The paper [Wit06] studied the upper and lower bounds on the running time 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. Then, [Wit06, Corollary 1] bounded the expectation of the time needed to reach a state of at least k leading ones from above by $\mu + 3ek \cdot \max\{\mu \ln(en), n\}$.

In [BDN10] exact expected running times for the $(1+1)$ EA on LeadingOnes were proven, which was an important cornerstone in the studies of LeadingOnes. The expected time to leave a state with the fitness of i is $E(A_i) = 1/((1-p)^i p)$ with mutation probability p. The next fitness value may be greater than $i+1$, but for LeadingOnes the bits following the i-th bit equal 1 with probability $1/2$. This yields the expected optimization time to be $((1-p)^{1-n} - (1-p))/(2p^2)$.

Doerr et al. reused this result in [DJWZ13, Section 4] to prove that the expected time to hit a target k is $((1-p)^{1-k} - (1-p))/(2p^2)$ for any p and k. They gave it only for $p = 1/n$, however, their proof does not depend on the value of p.

Pinto and Doerr [PD18, Theorem 11] extended this result to the similar algorithms: $(1+1)$ $\text{EA}_{>0}$, $(1+1)$ $\text{EA}_{0\to 1}$, and RLS. Note that Theorem 11 claims only the upper bound, however, it is clear from the above that this bound is 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.

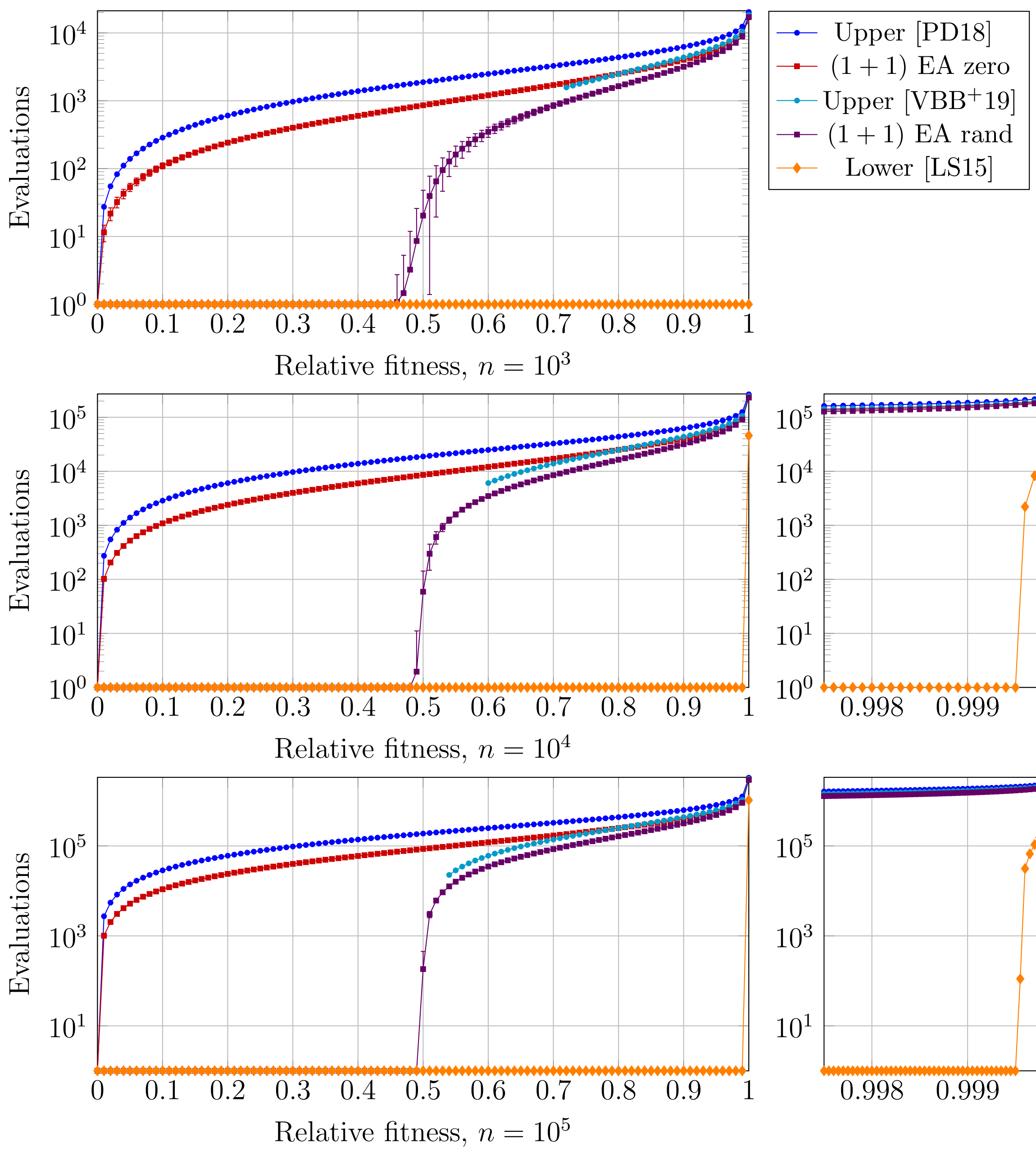

Figure 1: The runtime profile of the $(1 + 1)$ EA on OneMax and the applicable bounds; “zero” stands for starting at the all-zero point and “rand” for starting at a random point

4.2 OneMax

As the upper and lower bounds for OneMax are generally less precise than for LeadingOnes, we augment the theoretical results with the actual runtime profiles of the (1+1) EA in Figure 1.

In [LS15, Corollary 3], a lower bound for the $(1 + 1)$ EA on OneMax 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 $en \ln(n/(n-k)) - 2n \ln \ln n - 16n$. This bound is used in [LS15] to prove fixed-target results. From the fixed-target perspective, this results in a satisfactory lower bound only for a small fraction of possible conditions, see Figure 1 for visual comparison.

In [PD18, Theorem 10], the fixed-target upper bound was given for various flavors of $(1+1)$ EA which start from the worst possible solution consisting of only zeros. Assuming q to be the probability that one given bit, and only it, is flipped, the upper bound is $\frac{1}{q}(H_n - H_{n-k})$. This bound is exact for RLS and also captures the behavior of other $(1+1)$ EA flavors quite well (see Figure 1).

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 an additional argument to justify the use of $H_{n/2}$. Technically, it was proven only for the $(1+1)$ EA$_{>0}$, but the proof holds also for arbitrary q. This bound also gives a moderately good approximation, see again Figure 1.

In [Wit13, Theorem 6.5], lower bounds for the running times of the $(1+1)$ EA on OneMax are proven for mutation probabilities $p = O(n^{-2/3-\varepsilon})$. 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 to yield the lower bound of $(1 - o(1))(1-p)^{-n}(1/p)\min\{\ln n, \ln(1/(p^3 n^2))\}$.

This result was used in [VBB+19] to obtain the fixed-target lower bounds for the $(1+1)$ EA$_{>0}$ on OneMax. These bounds are piecewise because the interval S in [Wit13, Theorem 6.5] does not cover the entire range of k. Namely, for too small k it is zero, for too large k it does not depend on k, and only for $k \in S$ it can be used to bound the algorithm dynamics. As omitting $o(1)$ from the $1 - o(1)$ multiple gives overly optimistic results in this case, we do not provide the plots for these bounds.

4.3 BinVal

The fixed-target bounds for BinVal were proven in [VBB+19] for the $(1+1)$ 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 $2^n - 2^t \le k < 2^n - 2^{t+1}$, 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].

5 Fitness Levels

In this section we consider the fitness level theorems in the fixed-target context. Our observation is that the most important theorems of this sort are already suitable to produce fixed-target results.

5.1 Method

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 $1 \le i < m$ 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 it 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 now give several most popular fitness level theorems.

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

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

Theorem 2 (Fitness levels for lower bounds; Theorem 3 from [Sud13]). *Let $\{A_i\}_{1 \le i \le m}$ 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 $\sum_{j=i+1}^{m} \gamma_{i,j} = 1$. Assume there is some $\chi, 0 \le \chi \le 1$, such that $\gamma_{i,j} \ge \chi \sum_{k=j}^{m} \gamma_{i,k}$ for all $1 \le i < j \le m$. Then the expected hitting time of A_m is at least*

$$\sum_{i=1}^{m-1} P[\mathcal{A} \textit{ starts in } A_i] \cdot \left(\frac{1}{u_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{u_i} \right). \tag{2}$$

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

Theorem 3 (Refined fitness levels for upper bounds; Theorem 4 from [Sud13]). *Let $\{A_i\}_{1 \le i \le m}$ 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 $\sum_{j=i+1}^{m} \gamma_{i,j} = 1$. Assume there is some $\chi, 0 \le \chi \le 1$, such that $\gamma_{i,j} \le \chi \sum_{k=j}^{m} \gamma_{i,k}$ for all $1 \le i < j < m$. Further, assume that $(1 - \chi)s_i \le s_{i+1}$ for all $1 \le i \le m - 2$. Then the expected hitting time of A_m is at most*

$$\sum_{i=1}^{m-1} P[\mathcal{A} \textit{ starts in } A_i] \cdot \left(\frac{1}{s_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{s_i} \right). \tag{3}$$

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. We are sure that our observations can also be extended to that theorem.

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 $A_{m'}, A_{m'+1}, \dots, A_m$ 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 resp. Theorems 1, 2 and 3. 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 occurences are trivial, since they are limited only by identity inequalities $\gamma_{i,m'} \le \chi\gamma_{i,m'}$ 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. For this reason, they can be freely used in the fixed-target context. Nevertheless, Observation 1 is more than that, as it says that, whenever the bounds on the probabilities are known for the optimum-hitting process, they remain the same for a different target.

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 [PD18, 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. Follows the proof of [Sud13, Theorem 5]. □

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

$$\sum_{i=0}^{k-1} P[\mathcal{A} \textit{ starts with fitness } i] \cdot \left(\frac{1}{q_i} + \frac{1}{2}\sum_{j=i+1}^{k-1}\frac{1}{q_j}\right) = \frac{1}{2}\sum_{i=0}^{k-1}\frac{1}{q_i}. \tag{4}$$

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 $P[\mathcal{A}$ 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;*
- $\frac{(1-p)^{1-k}-(1-p)}{2p^2}(1-(1-p)^n)$ *for the* $(1+1)$ *EA*$_{>0}$*;*
- $\frac{1}{2} \cdot \sum_{i=0}^{k-1} \frac{1}{p(1-p)^i+\frac{1}{n}(1-p)^n}$ *for the* $(1+1)$ *EA*$_{0\to1}$*.*

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 $(1+1)$ EA$_{>0}$, $q_i = (1-p)^i \cdot p \cdot (1-(1-p)^n)$;
- for the $(1+1)$ EA$_{0\to1}$, $q_i = (1-p)^i \cdot p + \frac{1}{n}(1-p)^n$.

The simplification of the sum for the $(1+1)$ EA and the $(1+1)$ EA$_{>0}$ is done as in [BDN10]. $\square$

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]. $\square$

Our results are formalized as follows.

Theorem 5. *The expected fixed-target time for a* $(1+1)$ *EA that flips a single bit with probability q to reach a target of at least k on* OneMax *of size n, assuming $k \geq n/2 + \sqrt{n}\ln n$, 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.*

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:

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

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:

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

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 $i \ge k$, which is further multiplied by an exponentially small value, since for $k \ge n/2 + \sqrt{n}\ln n$ it holds that $\frac{1}{2^n}\sum_{i=k}^{n}\binom{n}{i} = n^{-\Omega(\ln n)}$ by the Chernoff bound. As a result, the fixed-target upper bound for the random initialization is $\frac{1}{q}(H_{n/2} - H_{n-k} - \frac{1-o(1)}{2n})$. $\square$

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 the mutation probability $p \le 1/(\sqrt{n}\log n)$, and assuming $\ell = \lceil n - \min\{n/\log n, 1/(p^2 n \log n)\}\rceil$, the expected time to find a solution with fitness at least $k \ge \ell$ 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 particular formula is taken from [Sud13, journal page 428, bottom of left column], so that we do not lose precision when k is close to n, and also to match the shape of Theorem 5. $\square$

We conjecture that a similar result can be derived for the generalized $(1+1)$ EAs as well, however, finding the exact leading multiple 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 the fixed-target manner.

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:*

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

6.1 Method

Drift theorems translate the bounds on the expected progress into the bounds on the expected first-hitting running times. They are usually formulated in terms of a random process that needs to hit a certain minimum value, which corresponds to minimization. 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 running time, 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 $E[X_T \mid X_0] \le k$, 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 random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t \le k\}$. Suppose that:*

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

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

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 random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t \le k\}$. Suppose that:*

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

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

More advanced drift theorems, such as the multiplicative drift theorems [DJW12] and variable drift theorems [DFW11, LW14], make it easier to prove rather sharp bounds on hitting times for processes with drift that depends on the current value. This is more common in evolutionary algorithms even on simple problems.

Most of the popular drift theorems of this sort can be classified using the following properties: they estimate the time for a process either to 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, motivated by having all variants in one paper [KK19], revealed that only two of the 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 the theorems which we use in this paper.

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 random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t < k'\}$. Furthermore, suppose that:*

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

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

Theorem 11 (Variable drift, fixed-target lower bounds; adapted from Theorem 7 from [DFW11])**.** *Let k be the target value, let $(X_t)_{t \in \mathbb{N}}$ be random variables over $\mathbb{R}$, and let $T = \inf\{t \mid X_t \le k\}$. Suppose that there are two continuous monotonically increasing functions $c, h : \mathbb{R}_0^+ \to \mathbb{R}^+$, and that for all $t < T$ holds:*

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

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

6.2 Applications

6.2.1 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 connectivity components with a large weight to faciliate 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 time $O(m^2(\log nw_{\max}))$, where m is the number of edges, n is the number of vertices, and $w_{\max}$ is the maximum edge weight.

Theorem 12. *Starting from a randomly initialized graph, the expected time for a* $(1+1)$ *EA that flips a single bit 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 $g(x) = s-1$, where s is the number of connectivity 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 13. *Starting from some spanning tree, the expected time for a* $(1+1)$ *EA that flips exactly two bits with probability* q *to find spanning tree with the weight at most* k *larger than the minimum possible weight is at most* $\frac{1}{q}(1+\ln\frac{(m-1)w_{\max}}{k+1})$.

Proof. We again re-use 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: $q = \frac{2}{m(m-1)}$;
- $(1+1)$ EA and $(1+1)$ EA$_{0\to 1}$: $q = p^2(1-p)^{m-2}$;
- $(1+1)$ EA$_{>0}$: $q = \frac{p^2(1-p)^{m-2}}{1-(1-p)^m}$;
- $(1+1)$ EA$_{0\to 2}$: $q = p^2(1-p)^{m-2} + (1-p)^m\frac{2}{m(m-1)}$.

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.

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

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

Theorem 14 (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 $2n/3 < k < n$, when n is large enough, is at least*

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

Proof. The basis of this proof is [DFW11, Theorem 5]. Our aim is to apply the Theorem 11, which allows jumps below the target. This allows to use the original the potential function $X_t = n - f(x)$ and the existing bound on the expected drift [DFW11, Lemma 6]: $E[X_t - X_{t+1} \mid X_t = s] \le \frac{s}{en}(1 + \frac{16s}{n})$.

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

We also reuse the function $h(x)$ from [DFW11, Theorem 5], which is $h(x) \le \frac{x+2\sqrt{x}}{en}(1+\frac{16+32\sqrt{x}}{n})$, and apply Theorem 11 to get

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

We choose the target value to encounter $k' = n - k$, and note that the arctangent is strictly increasing so we can prune it away.

We finish the bound with defining X_0 to be at least $n/3$ on the initialization with probability $1 - e^{-\Omega(n)}$ using a Chernoff bound, so the leading constant remains $1 - o(1/n)$. □

Figure 2 illustrates that the just-proven bound 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

From the section dedicated to drift theorems, we already got the message that, in order to obtain fixed-target results from the existing optimization time results, more powerful drift theorems either require to prove additional statements, such as the expected fitness

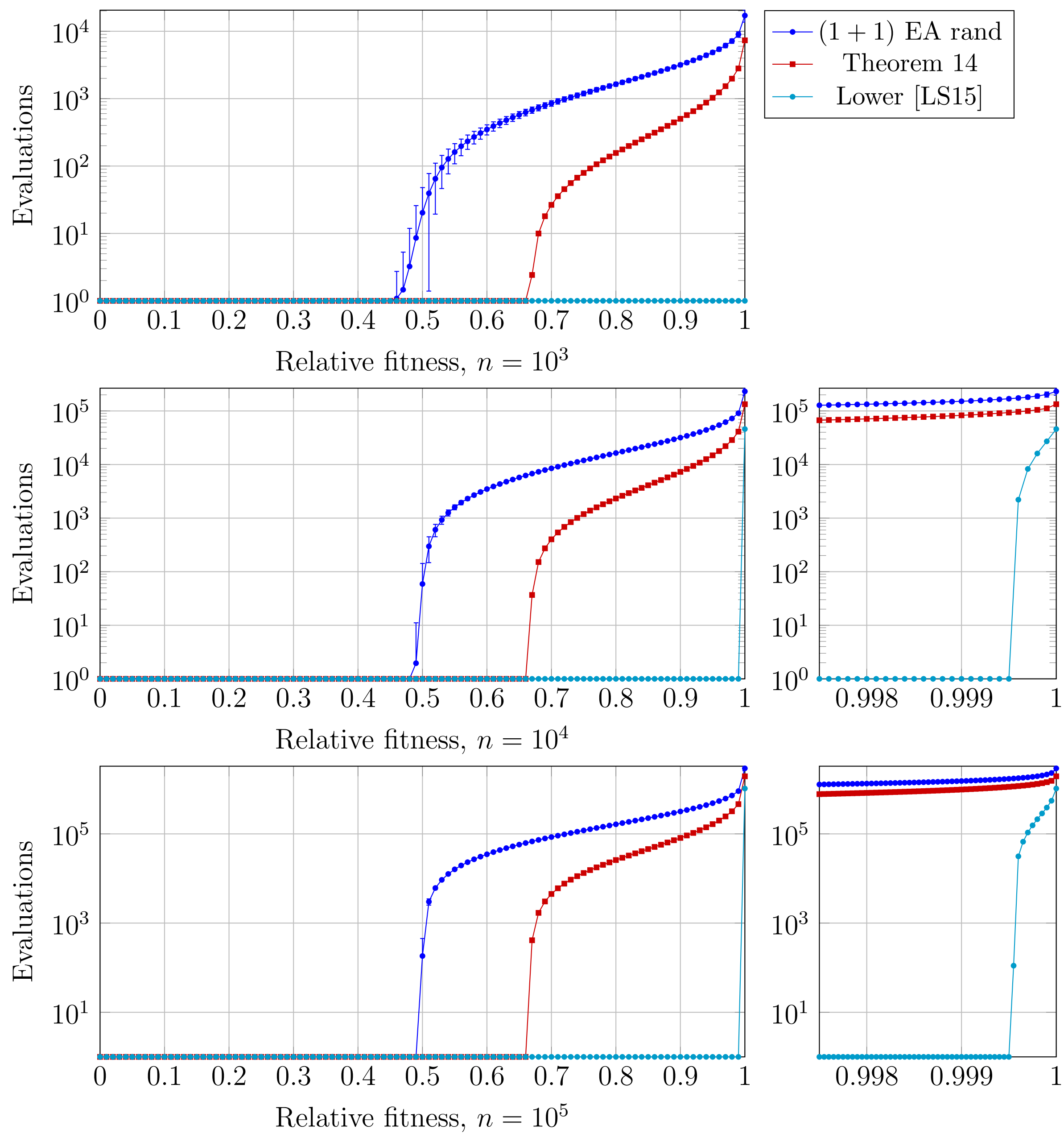

Figure 2: The comparison of the fixed-target lower bounds for the (1+1) EA on OneMax

when the algorithm reaches the target, or produce too pessimistic results when they do not care about that.

Some of the existing optimization time results, which could be a source for the fixed-target results, apply very different effort to different fitness values. This is often a consequence of the desire to get a good result with less effort. For instance, not very much effort is paid to fitness values of the form cn, $c < 1$, in simple applications to $(1 + 1)$ EA on OneMax, which results in an overestimation of the running time in these regions, and, consequently, in an overestimation of the fixed-target results. For the lower bounds, a common pattern is to concentrate only on fitness values that are difficult enough for the algorithm and convenient enough to apply a theorem, which results in piecewise fixed-target lower bounds that do not give all the information about the behavior of the algorithm.

However, most of these difficulties come from the scientific patterns of the theoretic community and hence can be described as the *social* ones. The difficulties that arise from the structure of the problems can lead to the differences between lower and upper bounds that are much more difficult to close than for the optimization times of the same problems. For instance, any problem which can be optimized by an algorithm in more than one possible way immediately introduces a large gap between the possible upper and lower fixed-target bounds. This can be seen already for hill climbers on linear functions, for which the optimization times are known for more than a decade [Wit06], but attempts to apply the same methods in the same fashion failed so far: the extreme example, BinVal, required a radically different approach in [VBB$^+$19]. Combinatorial problems with independent parts, where each part can be easy or hard, will be a similarly hard challenge.

8 Conclusion

In this first work solely devoted to fixed-target results in discrete runtime analysis, most of our results indicate that deriving fixed target results for the whole set of reasonable targets is 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 such a fixed-target spectrum of results is much more informative than the optimization time alone, we can only advocate to conduct future runtime analyses in this more general perspective. As discussed, this often needs no different proofs, all that is required is to formulate the information present in the proof also in the result. That said, for some problems fixed-target results seem to be much harder to obtain, and this might be a fruitful direction for further research.

Acknowledgments

This was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences, as well by the Government of the Russian Federation (Grant 08-08).

References

[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.

[BLS14] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *Parallel Problem Solving from Nature, PPSN 2014*, pages 892–901. Springer, 2014.

[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.

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

[DN20] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization.* Springer, 2020.

[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`.

[HAM$^+$16] Niklaus Hansen, Anne Auger, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. `https://arxiv.org/abs/1603.08785`, 2016.

[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.

[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.

[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.

[LW14] Per Kristian Lehre and Carsten Witt. Concentrated hitting times of randomized search heuristics with variable drift. In *International Symposium on Algorithms and Computation, ISAAC 2014*, pages 686–697. Springer, 2014.

[NNS14] Samadhi Nallaperuma, Frank Neumann, and Dirk Sudholt. A fixed budget analysis of randomized search heuristics for the traveling salesperson problem. In *Genetic and Evolutionary Computation Conference, GECCO 2014*, pages 807–814, 2014.

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

[PD18] Eduardo Carvalho Pinto and Carola Doerr. Towards a more practice-aware runtime analysis of evolutionary algorithms. `https://arxiv.org/abs/1812.00493`, 2018.

[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 R. Sarker, X. Yao, and M. Mohammadian, 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.