



HAL
open science

Lower bounds from fitness levels made easy

Benjamin Doerr, Timo Kötzing

► **To cite this version:**

Benjamin Doerr, Timo Kötzing. Lower bounds from fitness levels made easy. GECCO '21: Genetic and Evolutionary Computation Conference, 2021, Lille, France. pp.1142-1150, 10.1145/3449639.3459352 . hal-04486470

HAL Id: hal-04486470

<https://hal.science/hal-04486470>

Submitted on 4 Apr 2024

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.

Lower Bounds from Fitness Levels Made Easy^{*†}

Benjamin Doerr^{**} Timo Kötzing^{††}

April 29, 2021

Abstract

One of the first and easy to use techniques for proving run time bounds for evolutionary algorithms is the so-called method of fitness levels by Wegener. It uses a partition of the search space into a sequence of levels which are traversed by the algorithm in increasing order, possibly skipping levels. An easy, but often strong upper bound for the run time can then be derived by adding the reciprocals of the probabilities to leave the levels (or upper bounds for these). Unfortunately, a similarly effective method for proving lower bounds has not yet been established. The strongest such method, proposed by Sudholt (2013), requires a careful choice of the viscosity parameters $\gamma_{i,j}$, $0 \leq i < j \leq n$.

In this paper we present two new variants of the method, one for upper and one for lower bounds. Besides the level leaving probabilities, they only rely on the probabilities that levels are visited at all. We show that these can be computed or estimated without greater difficulties and apply our method to reprove the following known results in an easy and natural way. (i) The precise run time of the (1+1) EA on LEADINGONES. (ii) A lower bound for the run time of the (1+1) EA on ONEMAX, tight apart from an $O(n)$ term. (iii) A lower bound for the run time of the (1+1) EA on long k -paths. We also prove a tighter lower bound for the run time of the (1+1) EA on jump functions by showing that, regardless of the jump size, only with probability $O(2^{-n})$ the algorithm can avoid to jump over the valley of low fitness.

^{*}This work was supported by a public grant as part of the Investissements d’avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, and by the Deutsche Forschungsgemeinschaft (DFG), grant FR 2988/17-1.

[†]Extended version of the conference paper [DK21] accepted for publication in the proceedings of *GECCO 2021*.

^{**}Laboratoire d’Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

^{††}Hasso Plattner Institute, Potsdam, Germany

1 Introduction

The theory of evolutionary computation aims at explaining the behavior of evolutionary algorithms, for example by giving detailed run time analyses of such algorithms on certain test functions, defined on some search space (for this paper we will focus on $\{0, 1\}^n$). The first general method for conducting such analyzes is the *fitness level method (FLM)* [Weg01, Weg02]. The idea of this method is as follows. We partition the search space into a number m of sections (“levels”) in a linear fashion, so that all elements of later levels have better fitness than all elements of earlier levels. For the algorithm to be analyzed we regard the best-so-far individual and the level it is in. Since the best-so-far individual can never move to lower levels, it will visit each level at most once (possibly staying there for some time). Suppose we can show that, for any level $i < m$ which the algorithm is currently in, the probability to leave this level is at least p_i . Then, bounding the expected waiting for leaving a level i by $1/p_i$, we can derive an upper bound for the run time of $\sum_{i=1}^{m-1} 1/p_i$ by pessimistically assuming that we visit (and thus have to leave) each level $i < m$ before reaching the target level m . The fitness level method allows for simple and intuitive proofs and has therefore frequently been applied. Variations of it come with tail bounds [Wit14], work for parallel EAs [LS14], or admit non-elitist EAs [Leh11, DL16, CDEL18, DK19].

While very effective for proving upper bounds, it seems much harder to use fitness level arguments to prove lower bounds (see Theorem 5 for an early attempt). The first (and so far only) to devise a fitness level-based lower bound method that gives competitive bounds was Sudholt [Sud13]. His approach uses viscosity parameters $\gamma_{i,j}$, $0 \leq i < j \leq n$, which control the probability of the algorithm to jump from one level i to a higher level j (see Section 3.3 for details). While this allows for deriving strong results, the application is rather technical due to the many parameters and the restrictions they have to fulfill.

In this paper, we propose a new variant of the FLM for lower bounds, which is easier to use and which appears more intuitive. For each level i , we regard the *visit probability* v_i , that is, the probability that level i is visited at all during a run of the algorithm. This way we can directly characterize the run time of the algorithm as $\sum_{i=1}^{m-1} v_i/p_i$ when p_i is the precise probability to leave level i independent of where on level i the algorithm is. When only estimates for these quantities are known, e.g., because the level leaving probability is not independent from the current state, then we obtain the corresponding upper or lower bounds on the expected run time (see Section 3.4 for details).

We first use this method to give the precise expected run time of the $(1 + 1)$ EA on LEADINGONES in Section 4. While this run time was already well-understood before, it serves as a simple demonstration of the ease with which our method can be applied.

Next, in Section 5, we give a bound on the expected run time of the $(1 + 1)$ EA on ONEMAX, precise apart from terms of order $\Theta(n)$. Such bounds have also been known before, but needed much deeper methods (see Section 5.3 for a detailed discussion). Sudholt’s lower bound method has also been applied to this problem, but gave a slightly weaker bound deviating from the truth by an $O(n \log \log n)$ term. In addition to the precise result, we feel that our FLM with visit probabilities gives a clearer structure of the proof than the previous works.

In Section 6, we prove tighter lower bounds for the run time of the $(1 + 1)$ EA on jump functions. We do so by determining (asymptotically precise) the probability that in a run of the $(1 + 1)$ EA on a jump function the algorithm does not reach a non-optimal search point outside the fitness valley (and thus does not have to jump over this valley). Interestingly, this probability is only $O(2^{-n})$ regardless of the jump size (width of the valley).

Finally, in Section 7, we consider the $(1 + 1)$ EA on so-called long k -paths. We show how the FLM with visit probabilities can give results comparable to those of the FLM with viscosities while again being much simpler to apply.

2 The $(1 + 1)$ EA

In this paper we consider exactly one randomized search heuristic, the $(1 + 1)$ EA. It maintains a single individual, the best it has seen so far. Each iteration it uses standard bit mutation with mutation rate $p \in (0, 1)$ (flipping each bit of the bit string independently with probability p) and keeps the result if and only if it is at least as good as the current individual under a given fitness function f . We give a more formal definition in Algorithm 1.

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

```

1 Let  $x$  be a uniformly random bit string from  $\{0, 1\}^n$ ;
2 while optimum not reached do
3    $y \leftarrow \text{mutate}_p(x)$ ;
4   if  $f(y) \geq f(x)$  then  $x \leftarrow y$ ;
```

3 The Fitness Level Methods

The fitness level method is typically phrased in terms of a *fitness-based partition*, that is, a partition of the search space into sets A_1, \dots, A_m such that elements of later sets have higher fitness. We first introduce this concept and abstract away from it to ease the notation. After this, in Section 3.2, we state the original FLM. In Section 3.3 we describe the lower bound based on the FLM from Sudholt [Sud13], before presenting our own variant, the FLM with visit probabilities, in Section 3.4.

3.1 Level Processes

Definition 1 (Fitness-Based Partition [Weg02]). *Let $f : \{0, 1\}^n \rightarrow \mathbb{R}$ be a fitness function. A partition A_1, \dots, A_m of $\{0, 1\}^n$ is called a fitness-based partition if for all $i, j \leq m$ with $i < j$ and $x \in A_i, y \in A_j$, we have $f(x) < f(y)$.*

We will use the shorthands $A_{\geq i} = \bigcup_{j=i}^m A_j$ and $A_{\leq i} = \bigcup_{j=1}^i A_j$.

In order to simplify our notation, we focus on processes on $[1..m]$ (the levels) with underlying Markov chain as follows.

Definition 2 (Non-decreasing Level Process). *A stochastic process $(X_t)_t$ on $[1..m]$ is called a non-decreasing level process if and only if (i) there exists a Markov process $(Y_t)_t$ over a state space S such that there is an $\ell : S \rightarrow [1..m]$ with $\ell(Y_t) = X_t$ for all t , and (ii) the process $(X_t)_t$ is non-decreasing, that is, we have $X_{t+1} \geq X_t$ with probability one for all t .*

We later want to analyze algorithms in terms of non-decreasing level processes, making the transition as follows. Suppose we have an algorithm with state space $\{0, 1\}^n$. Denoting by Y_t the best among the first t search points generated by the algorithm, this defines a Markov Chain $(Y_t)_t$ in the state space $S = \{0, 1\}^n$, the run of the algorithm. Further, suppose the algorithm optimizes a fitness function f such that the state of the algorithm is non-decreasing in terms of fitness. In order to get a non-decreasing level process, we can now define any fitness-based partition and get a corresponding *level function* $\ell : S \rightarrow [1..m]$ by mapping any $x \in S$ to the unique i with $x \in A_i$. Then the process $(\ell(Y_t))_t$ is a non-decreasing level process.

The main reason for us to use the formal notion of a level process is the property formalized in the following lemma. Essentially, if a level process makes progress with probability at least p in each iteration (regardless of the precise current state), then the expected number of iterations until the process progresses is at most $1/p$. This situation resembles a geometric

distribution, but does not assume independence of the different iterations (one could show that the time to progress is stochastically dominated by a geometric distribution with success rate p , but we do not need this level of detail).

Lemma 3. *Let $(X_t)_t$ be a non-decreasing level process with underlying Markov chain $(Y_t)_t$ and level function ℓ . Assume X_t starts on some particular level. Let $p \in (0, 1]$ be a lower bound on the probability for level process to leave this level regardless of the state of the underlying Markov chain. Then the expected first time t such that X_t changes is at most $1/p$.*

Analogously, if p is an upper bound, the expected time t such that X_t changes is at least $1/p$.

Proof. We let $(Z_t)_t$ be the stochastic process on $\{0, 1\}$ such that Z_t is 1 if and only if $X_t > X_0$. According to our assumptions, we have, for all t before the first time that $Z_t = 1$, that $E[Z_{t+1} - Z_t \mid Z_t] \geq p$. From the additive drift theorem [HY01, Len20] we obtain that the expected first time such that $Z_t = 1$ is bounded by $1/p$ as desired. The “analogously” clause follows analogously. \square

3.2 Original Fitness Level Method

The following theorem contains the original Fitness Level Method and makes the basic principle formal.

Theorem 4 (Fitness Level Method, upper bound [Weg02]). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2).*

For all $i \in [1..m - 1]$, let p_i be a lower bound on the probability of a state change of $(X_t)_t$, conditional on being in state i . Then the expected time for $(X_t)_t$ to reach the state m is

$$E[T] \leq \sum_{i=1}^{m-1} \frac{1}{p_i}.$$

This bound is very simple, yet strong. It is based on the idea that, in the worst case, all levels have to be visited sequentially. Note that one can improve this bound (slightly) by considering only those levels which come after the (random) start level X_0 (by changing the start of the sum to X_0 instead of 1). Intuitively, low levels that are never visited do not need to be left.

There is a lower bound based on the observation that at least the initial level has to be left (if it was not the last level).

Theorem 5 (Fitness Level Method, lower bound [Weg02]). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2).*

For all $i \in [1..m-1]$, let p_i be an upper bound on the probability of a state change, conditional on being in state i . Then the expected time for $(X_t)_t$ to reach the state m is

$$E[T] \geq \sum_{i=0}^{m-1} \Pr[X_0 = i] \frac{1}{p_i}.$$

This bound is very weak since it assumes that the first improvement on the initial search point already finds the optimum.

We note, very briefly, that a second main analysis method, *drift analysis*, also has additional difficulties with lower bounds. Additive drift [HY01], multiplicative drift [DJW12], and variable drift [MRC09, Joh10] all easily give upper bounds for run times, however, only the additive drift theorem yields lower bounds with the same ease. The existing multiplicative [Wit13, DDK18, DKLL20] and variable [DFW11, FK13, GW18, DDY20] drift theorems for lower bounds all need significantly stronger assumptions than their counterparts for upper bounds.

3.3 Fitness Level Method with Viscosity

While the upper bound above is strong and useful, the lower bound is typically not strong enough to give more than a trivial bound. Sudholt [Sud13] gave a refinement of the method by considering bounds on the transition probabilities from one level to another.

Theorem 6 (Fitness Level Method with Viscosity, lower bound [Sud13]). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2). Let $\chi, \gamma_{i,j} \in [0, 1]$ and $p_i \in (0, 1]$ be such that*

- *for all t , if $X_t = i$, the probability that $X_{t+1} = j$ is at most $p_i \cdot \gamma_{i,j}$;*
- *$\sum_{j=i+1}^m \gamma_{i,j} = 1$; and*
- *for all $j > i$, we have $\gamma_{i,j} \geq \chi \sum_{k=j}^m \gamma_{i,k}$.*

Then the expected time for $(X_t)_t$ to reach the state m is

$$E[T] \geq \sum_{i=1}^{m-1} \Pr[X_0 = i] \chi \sum_{j=i}^{m-1} \frac{1}{p_j}.$$

This result is much stronger than the original lower bound from Fitness Level Method, since now the leaving probabilities of all segments are part of the bound, at least with a fractional impact prescribed by χ . The weakness of the method is that χ has to be defined globally, the same for all segments i .

There is also a corresponding upper bound as follows.

Theorem 7 (Fitness Level Method with Viscosity, upper bound [Sud13]). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2). Let $\chi, \gamma_{i,j} \in [0, 1]$ and $p_i \in (0, 1]$ be such that*

- for all t , if $X_t = i$, the probability that $X_{t+1} = j$ is at least $p_i \cdot \gamma_{i,j}$;
- $\sum_{j=i+1}^m \gamma_{i,j} = 1$;
- for all $j > i$, we have $\gamma_{i,j} \leq \chi \sum_{k=j}^m \gamma_{i,k}$; and
- for all $j \leq m - 2$, we have $(1 - \chi)p_j \leq p_{j+1}$.

Then the expected time for $(X_t)_t$ to reach the state m is

$$E[T] \leq \sum_{i=1}^{m-1} \Pr[X_0 = i] \left(\frac{1}{p_j} + \chi \sum_{j=i+1}^{m-1} \frac{1}{p_j} \right).$$

3.4 Fitness Level Method with Visit Probabilities

In this paper, we give a new FLM theorem for proving lower bounds. The idea is that exactly all those levels that have ever been visited need to be left; thus, we can use the expected waiting time for leaving a specific level multiplied with the probability of visiting that level at all. The following theorem makes this idea precise for lower bounds; Theorem 9 gives the corresponding upper bound. We note that for the particular case of the optimization of the LEADINGONES problem via $(1 + 1)$ -type elitist algorithms, our bounds are special cases of [DJWZ13, Lemma 5] and [Doe19, Theorem 3].

Theorem 8 (Fitness Level Method with visit probabilities, lower bound). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2). For all $i \in [1..m - 1]$, let p_i be an upper bound on the probability of a state change of $(X_t)_t$, conditional on being in state i . Furthermore, let v_i be a lower bound on the probability of there being a t such that $X_t = i$. Then the expected time for $(X_t)_t$ to reach the state m is*

$$E[T] \geq \sum_{i=1}^{m-1} \frac{v_i}{p_i}.$$

Proof. For each $i < m$, let T_i be the (random) time spent in level i . Thus,

$$T = \sum_{i=1}^{m-1} T_i.$$

Let now $i < m$. We want to show that $E[T_i] \geq v_i/p_i$. We let E be the event that the process ever visits level i and compute

$$E[T_i] = E[T_i | E] \Pr[E] + E[T_i | \bar{E}] \Pr[\bar{E}] \geq E[T_i | E]v_i.$$

For all t with $X_t = i$, with probability at most p_i , we have $X_{t+1} > i$. Thus, using Lemma 3, the expected time until a search point with $X_k > i$ is found is at least $1/p_i$, giving $E[T_i | E] \geq 1/p_i$ as desired. \square

A strength of this formulation is that skipping levels due to a higher initialization does not need to be taken into account separately (as in the two previous lower bounds), it is part of the visit probabilities. A corresponding upper bound follows with analogous arguments.

Theorem 9 (Fitness Level Method with visit probabilities, upper bound). *Let $(X_t)_t$ be a non-decreasing level process (as detailed in Definition 2).*

For all $i \in [1..m-1]$, let p_i be a lower bound on the probability of a state change of $(X_t)_t$, conditional on being in state i . Furthermore, let v_i be an upper bound on the probability there being a t such that $X_t = i$. Then the expected time for $(X_t)_t$ to reach the state m is

$$E[T] \leq \sum_{i=1}^{m-1} \frac{v_i}{p_i}.$$

In a typical application of the method of the FLM, finding good estimates for the leaving probabilities is easy. It is more complicated to estimate the visit probabilities accurately, so we propose one possible approach in the following lemma.

Lemma 10. *Let $(Y_t)_t$ be a Markov-process over state space S and $\ell : S \rightarrow [1..m]$ a level function. For all t , let $X_t = \ell(Y_t)$ and suppose that $(X_t)_t$ is non-decreasing. Further, suppose that $(X_t)_t$ reaches state m after a finite time with probability 1.*

Let $i < m$ be given. For any $x \in S$ and any set $M \subseteq S$, let $x \rightarrow M$ denote the event that the Markov chain with current state x transitions to a state in M . For all j let $A_j = \{s \in S \mid \ell(s) = j\}$. Suppose there is v_i such that, for all $x \in A_{\leq i-1}$ with $\Pr[x \rightarrow A_{\geq i}] > 0$,

$$\Pr[x \rightarrow A_i \mid x \rightarrow A_{\geq i}] \geq v_i,$$

and

$$\Pr[Y_0 \in A_i \mid Y_0 \in A_{\geq i}] \geq v_i.$$

Then v_i is a lower bound for visiting level i as required by Theorem 8.

Proof. Let T be minimal such that $Y_T \in A_{\geq i}$. Then the probability that level i is being visited is $\Pr[Y_T \in A_i]$, since $(X_t)_t$ is non-decreasing.

By the law of total probability we can show the claim by showing it first conditional on $T = 0$ and then conditional on $T \neq 0$.

We have that $T = 0$ is equivalent to $Y_0 \in A_{\geq i}$, thus we have $\Pr[Y_T \in A_i \mid T = 0] \geq v_i$ from the second condition in the statement of the lemma.

Otherwise, let $x = Y_{T-1}$. Since $Y_T \in A_{\geq i}$,

$$\begin{aligned} \Pr[Y_T \in A_i \mid T \neq 0] &= \Pr[Y_T \in A_i \mid Y_T \in A_{\geq i}, T \neq 0] \\ &= \Pr[x \rightarrow A_i \mid x \rightarrow A_{\geq i}, T \neq 0] \\ &= \Pr[x \rightarrow A_i \mid x \rightarrow A_{\geq i}]. \end{aligned}$$

As T was chosen minimally, we have $x \notin A_{\geq i}$ and thus get the desired bound from the first condition in the statement of the lemma. \square

Implicitly, the lemma suggests to take the minimum of all these conditional probabilities over the different choices for x . Note that this estimate might be somewhat imprecise since worst-case x might not be encountered frequently. Also note that a corresponding upper bound for Theorem 9 follows analogously.

4 The Precise Run Time for LeadingOnes

One of the classic fitness functions used for analyzing the optimization behavior of randomized search heuristics is the LEADINGONES function. Given a bit string x of length n , the LEADINGONES value of x is defined as the number of 1s in the bit string before the first 0 (if any). In parallel independent work, the precise expected run time of the $(1+1)$ EA on the LEADINGONES benchmark function was determined in [BDN10, Sud13]. Even more, the distribution of the run time was determined with variants of the FLM in [DJWZ13, Doe19]. As a first simple application of our methods, we now determine the precise run time of the $(1+1)$ EA on LEADINGONES via Theorems 8 and 9.

Theorem 11. *Consider the $(1+1)$ EA optimizing LEADINGONES with mutation rate p . Let T be the (random) time for the $(1+1)$ EA to find the*

optimum. Then

$$E[T] = \frac{1}{2} \sum_{i=0}^{n-1} \frac{1}{(1-p)^i p}.$$

Proof. We want to apply Theorems 8 and 9 simultaneously. We partition the search space in the canonical way such that, for all $i \leq n$, A_i contains the set of all search points with fitness i . Now we need a precise result for the probability to leave a level and for the probability to visit a level.

First, we consider the probability p_i to leave a given level $i < n$. Suppose the algorithm has a current search point in A_i , so it has i leading 1s and then a 0. The algorithm leaves level A_i now if and only if it flips the first 0 of the bit string (probability of p) and no previous bits (probability $(1-p)^i$). Hence, $p_i = p(1-p)^i$.

Next we consider the probability v_i to visit a level i . We claim that it is exactly $1/2$, following reasoning given in several places before [DJW02, Sud13]. We want to use Lemma 10 and its analog for upper bounds. Let i be given. For the initial search point, if it is at least on level i (the condition considered by the lemma), the individual is on level i if and only if the $i+1$ st bit is a 0, so exactly with probability $1/2$ as desired for both bounds. Before an individual with at least i leading 1s is created, the bit at position $i+1$ remains uniformly random (this can be seen by induction: it is uniform at the beginning and does not experience any bias in any iteration while no individual with at least i leading 1 is created). Once such an individual is created, if the bit at position $i+1$ is 1, the level i is skipped, otherwise it is visited. Thus, the algorithm skips level i with probability exactly $1/2$, giving $v_i = 1/2$. With these exact values for the p_i and v_i , Theorems 8 and 9 immediately yield the claim. \square

By computing the geometric series in Theorem 11, we obtain as a (well-known) corollary that the $(1+1)$ EA with the classic mutation rate $p = 1/n$ optimizes LEADINGONES in an expected run time of $n^2 \frac{\epsilon-1}{2} (1 \pm o(1))$.

5 A Tight Lower Bound for OneMax

In this section, as a first real example of the usefulness of our general method, we prove a lower bound for the run time of the $(1+1)$ EA with standard mutation rate $p = \frac{1}{n}$ on ONEMAX, which is only by an additive term of order $O(n)$ below the upper bound following from the classic fitness level method. This is tighter than the best gap of order $O(n \log \log n)$ proven previously with fitness level arguments. Moreover, our lower bound is the tightest lower

bound apart from the significantly more complicated works that determine the run time precise apart from $o(n)$ terms. We defer a detailed account of the literature together with a comparison of the methods to Section 5.3.

We recall that the fitness levels of the ONEMAX function are given by

$$A_i := \{x \in \{0, 1\}^n \mid \text{OM}(x) = i\}, i \in [0..n].$$

We use the notation $A_{\geq i} := \bigcup_{j=i}^n A_j$ and $A_{\leq i} := \bigcup_{j=0}^i A_j$ for all $i \in [0..n]$ as defined above for fitness-based partitions, but with the appropriate bounds 0 and n instead of 1 and m .

We denote by $T_{k,\ell}$ the expected number of iterations the $(1+1)$ EA, started with a search point in A_k , takes to generate a search point in $A_{\geq \ell}$. We further denote by $T_{\text{rand},\ell}$ the expected number of iterations the $(1+1)$ EA started with a random search point takes to generate a solution in $A_{\geq \ell}$. These notions extend previously proposed fine-grained run time notions: $T_{\text{rand},\ell}$ is the fixed target run time first proposed in [DJWZ13] as a technical tool and advocated more broadly in [BDDV20]. The time $T_{k,n}$ until the optimum is found when starting with fitness k was investigated in [ABD20] when $k > n/2$, that is, when starting with a better-than-average solution. We spare the details and only note that such fine-grained complexity notions (which also include the fixed-budget complexity proposed in [JZ14]) have given a much better picture on how to use EAs effectively than the classic run time $T_{\text{rand},n}$ alone. In particular, it was observed that different parameters or algorithms are preferable when not optimizing until the optimum or when starting with a good solution.

For all $k, \ell \in [0..n]$, we denote by $p_{k,\ell}$ the probability that standard bit mutation with mutation rate $p = \frac{1}{n}$ creates an offspring in A_ℓ from a parent in A_k . We also write $p_{k,\geq \ell} := \sum_{j=\ell}^n p_{k,j}$ to denote the probability to generate an individual in $A_{\geq \ell}$ from a parent in A_k . Then $p_i := p_{i,\geq i+1}$ is the probability that the $(1+1)$ EA optimizing ONEMAX leaves the i -th fitness level.

5.1 Upper and Lower Bounds Via Fitness Levels

Using the notation just introduced, the classic fitness level method (see Theorem 4 and note that the fitness of the parent individuals describes a non-decreasing level process with state change probabilities p_i) shows that

$$T_{k,\ell} \leq \sum_{i=k}^{\ell-1} \frac{1}{p_i} =: \tilde{T}_{k,\ell}.$$

To prove a nearly matching lower bound employing our new methods, we first analyze the probability that the $(1+1)$ EA optimizing ONEMAX skips

a particular fitness level. Note that if q_i is the probability to skip the i -th fitness level, then $v_i := 1 - q_i$ is the probability to visit the i -th level as used in Theorem 8.

Lemma 12. *Let $i \in [0..n]$. Consider a run of the $(1 + 1)$ EA with mutation rate $p = \frac{1}{n}$ on the ONEMAX function started with a (possibly random) individual x with $\text{ONEMAX}(x) < i$. Then the probability q_i that during the run the parent individual never has fitness i satisfies*

$$q_i \leq \frac{n - i}{n(1 - \frac{1}{n})^{i-1}}.$$

Proof. Since we assume that we start below fitness level i , by Lemma 10 (and using the notation from that lemma for a moment) we have

$$\begin{aligned} q_i &\leq \max\{\Pr[x \rightarrow A_{\geq i+1} \mid x \rightarrow A_{\geq i}] \mid \text{ONEMAX}(x) < i\} \\ &\leq \max_{k \in [0..i-1]} \frac{p_{k, \geq i+1}}{p_{k, \geq i}}. \end{aligned}$$

Hence it suffices to show that $\frac{p_{k, \geq i+1}}{p_{k, \geq i}} \leq \frac{n-i}{n(1-\frac{1}{n})^{i-1}}$ for all $k \in [0..i-1]$, and this is what we will do in the remainder of this proof.

Let us, slightly abusing the common notation, write $\text{Bin}(m, p)$ to denote a random variable following a binomial law with parameters m and p . Let $k, \ell \in \mathbb{N}$ with $k \leq \ell$. Noting that the only way to generate a search point in A_ℓ from some $x \in A_k$ is to flip, for some $j \in [\ell - k.. \min\{n - k, \ell\}]$, exactly j of the $n - k$ zero-bits of x and exactly $j - (\ell - k)$ of the k one-bits, we easily obtain the well-known fact that

$$\begin{aligned} p_{k, \ell} &= \sum_{j=\ell-k}^{\min\{n-k, \ell\}} \Pr[\text{Bin}(n-k, p) = j] \Pr[\text{Bin}(k, p) = j - (\ell - k)] \\ &= \sum_{j=\ell-k}^{\min\{n-k, \ell\}} \binom{n-k}{j} \binom{k}{j - (\ell - k)} p^{2j - \ell + k} (1-p)^{n-2j + \ell - k}. \end{aligned}$$

Since $p = \frac{1}{n}$, the mode of $\text{Bin}(n-k, p)$ is at most 1. Since the binomial distribution is unimodal, we conclude that $\Pr[\text{Bin}(n-k, p) = j] \leq \Pr[\text{Bin}(n-k, p) = \ell - k]$ for all $j \geq \ell - k$. Consequently, the first line of the above set of equations gives

$$\begin{aligned} p_{k, \ell} &\leq \Pr[\text{Bin}(n-k, p) = \ell - k] \Pr[\text{Bin}(k, p) \in [0.. \min\{n - \ell, k\}]] \\ &\leq \Pr[\text{Bin}(n-k, p) = \ell - k] \end{aligned}$$

and thus

$$p_{k,\geq\ell} \leq \Pr[\text{Bin}(n-k, p) \geq \ell - k]. \quad (1)$$

We recall that our target is to estimate $\frac{p_{k,\geq i+1}}{p_{k,\geq i}}$ for all $k \in [0..i-1]$. By (1), we have

$$\begin{aligned} p_{k,\geq i+1} &\leq \Pr[\text{Bin}(n-k, p) \geq i+1-k] \\ &\leq \frac{(i+1-k)(1-p)}{i+1-k-(n-k)p} \Pr[\text{Bin}(n-k, p) = i+1-k], \end{aligned}$$

where the last estimate is [Doe20b, equation following Lemma 1.10.38]. We also have $p_{k,\geq i} \geq p_{k,i} \geq (1-p)^k \Pr[\text{Bin}(n-k, p) = i-k]$. Hence from

$$\begin{aligned} \frac{\Pr[\text{Bin}(n-k, p) = i+1-k]}{\Pr[\text{Bin}(n-k, p) = i-k]} &= \frac{\binom{n-k}{i+1-k} p^{i+1-k} (1-p)^{n-k-(i+1-k)}}{\binom{n-k}{i-k} p^{i-k} (1-p)^{n-k-(i-k)}} \\ &= \frac{(n-i)p}{(i+1-k)(1-p)} \end{aligned}$$

we conclude

$$\begin{aligned} \frac{p_{k,\geq i+1}}{p_{k,\geq i}} &\leq \frac{(i+1-k)(1-p)}{i+1-k-(n-k)p} \frac{(n-i)p}{(i+1-k)(1-p)^{k+1}} \\ &\leq \frac{n-i}{n(i-k)(1-\frac{1}{n})^k}, \end{aligned}$$

using again that $p = \frac{1}{n}$. For $k \in [0..i-1]$, this expression is maximal for $k = i-1$, giving that $q_i \leq \frac{n-i}{n(1-\frac{1}{n})^{i-1}}$ as claimed. \square

With this estimate, we can now easily give a very tight lower bound on the run time of the $(1+1)$ EA on ONEMAX.

Theorem 13. *Let $k, \ell \in [0..n]$ with $k < \ell$. Then the expected number $T_{k,\ell}$ of iterations the $(1+1)$ EA optimizing ONEMAX and initialized with any search point x with $\text{ONEMAX}(x) = k$ takes to generate a search point z with fitness $\text{ONEMAX}(z) \geq \ell$ is at least*

$$T_{k,\ell} \geq \tilde{T}_{k,\ell} - (\ell - k - 1)e(e-1) \exp\left(\frac{k}{n-1}\right),$$

where $\tilde{T}_{k,\ell}$ is the upper bound stemming from the fitness level method as defined at the beginning of this section. This lower bound holds also for $T_{k',\ell}$ with $k' \leq k$, that is, when starting with a search point x with $\text{ONEMAX}(x) \leq k$.

Proof. We use our main result Theorem 8. We note first that when assuming that the level process regarded in Theorem 8 starts on level k' , then the expected time for it to reach level ℓ or higher is at least $\sum_{i=k'}^{\ell-1} \frac{v_i}{p_i}$. This follows immediately from the proof of the theorem or by applying the theorem to the level process (X'_t) defined by $X'_t = \min\{\ell, X_t\} - k'$ for all t .

Consider now a run of the $(1+1)$ EA on the ONEMAX function started with an initial search point x_0 such that $k' = \text{ONEMAX}(x_0) \leq k$. Denote by x_t the individual selected in iteration t as future parent. Then $X_t = \text{ONEMAX}(x_t)$ defines a level process. As before, we denote the probabilities to visit level i by v_i , to not visit it by $q_i = 1 - v_i$, and to leave it to a higher level by p_i . Using our main result and the elementary argument above, we obtain an expected run time of

$$E[T_{k',\ell}] \geq \sum_{i=k'}^{\ell-1} \frac{v_i}{p_i} \geq \sum_{i=k}^{\ell-1} \frac{v_i}{p_i} \geq \sum_{i=k}^{\ell-1} \frac{1}{p_i} - \sum_{i=k+1}^{\ell-1} \frac{q_i}{p_i}.$$

We note that the first expression is exactly the upper bound $\tilde{T}_{k,\ell}$ stemming from the classic fitness level method. We estimate the second expression. We have

$$p_i = p_{i,\geq i+1} \geq p_{i,i+1} \geq \left(1 - \frac{1}{n}\right)^{n-1} \frac{n-i}{n}, \quad (2)$$

where the last estimate stems from regarding only the event that exactly one missing bit is flipped. Together with the estimate $q_i \leq \frac{n-i}{n(1-\frac{1}{n})^{i-1}}$ from Lemma 12, we compute

$$\begin{aligned} \sum_{i=k+1}^{\ell-1} \frac{q_i}{p_i} &\leq \sum_{i=k+1}^{\ell-1} \frac{n-i}{n(1-\frac{1}{n})^{i-1}} \frac{n}{(n-i)(1-\frac{1}{n})^{n-1}} \\ &= \sum_{i=k+1}^{\ell-1} \left(1 + \frac{1}{n-1}\right)^{n+i-2} \\ &= \left(1 + \frac{1}{n-1}\right)^{n+k-1} \sum_{j=0}^{\ell-k-2} \left(1 + \frac{1}{n-1}\right)^j \\ &= \left(1 + \frac{1}{n-1}\right)^{n+k-1} \frac{\left(1 + \frac{1}{n-1}\right)^{\ell-k-1} - 1}{\left(1 + \frac{1}{n-1}\right) - 1} \\ &= \left(1 + \frac{1}{n-1}\right)^{n+k-1} (n-1) \left(\left(1 + \frac{1}{n-1}\right)^{\ell-k-1} - 1 \right) \\ &\leq (n-1) \exp\left(\frac{n+k-1}{n-1}\right) \left(\exp\left(\frac{\ell-k-1}{n-1}\right) - 1 \right) \end{aligned} \quad (3)$$

$$\begin{aligned}
&= (n-1)e \exp\left(\frac{k}{n-1}\right) \left(\exp\left(\frac{\ell-k-1}{n-1}\right) - 1\right) \\
&\leq (n-1)(e-1)e \exp\left(\frac{k}{n-1}\right) \frac{\ell-k-1}{n-1},
\end{aligned}$$

where the estimate in (3) uses the well-known inequality $1+r \leq e^r$ valid for all $r \in \mathbb{R}$ and the last estimate exploits the convexity of the exponential function in the interval $[0, 1]$, that is, that $\exp(\alpha) \leq 1 + \alpha(\exp(1) - \exp(0))$ for all $\alpha \in [0, 1]$. \square

The result above shows that the classic fitness level method and our new lower bound method can give very tight run time results. We note that the difference $\delta_{k,\ell} = (\ell-k-1)e(e-1)\exp\left(\frac{k}{n-1}\right)$ between the two fitness level estimates is only of order $O(\ell-k)$, in particular, only of order $O(n)$ for the classic run time $T_{\text{rand},n}$, which itself is of order $\Theta(n \log n)$. Hence here the gap is only a term of lower order.

5.2 Estimating the Fitness Level Estimate $\tilde{T}_{k,\ell}$

To make our results above meaningful, it remains to analyze the quantity $\tilde{T}_{k,\ell} = \sum_{i=k}^{\ell-1} 1/p_i$, which is the estimate from the classic fitness level method.

Here, again, it turns out that upper bounds tend to be easier to obtain since they require a lower bound for the p_i , for which the estimate $p_i \geq (1 - \frac{1}{n})^{n-1} \frac{n-i}{n}$ from (2) usually is sufficient. To ease the presentation, let us use the notation $e_n = (1 - \frac{1}{n})^{-(n-1)}$ and note that $e(1 - \frac{1}{n}) \leq e_n \leq e$, see, e.g., [Doe20b, Corollary 1.4.6]. With this notation, the lower bound (2) gives the upper bound

$$\tilde{T}_{k,\ell} \leq e_n n \sum_{i=k}^{\ell-1} \frac{1}{n-i} =: \tilde{T}_{k,\ell}^+. \quad (4)$$

To prove a lower bound, we observe that

$$p_i = \sum_{d=1}^{n-i} \Pr[\text{Bin}(n-i, p) = d] \Pr[\text{Bin}(i, p) < d].$$

We can thus estimate

$$\begin{aligned}
p_i &\leq \Pr[\text{Bin}(n-i, p) = 1] \Pr[\text{Bin}(i, p) = 0] + \Pr[\text{Bin}(n-i, p) \geq 2] \\
&\leq \left(1 - \frac{1}{n}\right)^{n-1} \frac{n-i}{n} + \frac{(n-i)(n-i-1)}{2n^2},
\end{aligned} \quad (5)$$

where the last inequality follows from the estimate $\Pr[\text{Bin}(n, p) \geq k] \leq \binom{n}{k} p^k$, see, e.g., [GW17, Lemma 3] or [Doe20b, Lemma 1.10.37]. We note that the first summand in (5) is exactly our lower bound (2) for p_i , so it is the second term that determines the slack of our estimates. We estimate coarsely

$$\begin{aligned} \frac{1}{p_i} &\geq \left(\left(1 - \frac{1}{n}\right)^{n-1} \frac{n-i}{n} + \frac{(n-i)(n-i-1)}{2n^2} \right)^{-1} \\ &= \frac{2e_n n^2}{2n(n-i) + e_n(n-i)(n-i-1)} \\ &= \frac{e_n n}{n-i} - \frac{e_n^2 n(n-i-1)}{(n-i)(2n + e_n(n-i-1))} \geq \frac{e_n n}{n-i} - \frac{1}{2} e_n^2. \end{aligned}$$

Summing over the fitness levels, we obtain

$$\begin{aligned} \tilde{T}_{k,\ell} &= \sum_{i=k}^{\ell-1} \frac{1}{p_i} \\ &\geq \sum_{i=k}^{\ell-1} \left(\frac{e_n n}{n-i} - \frac{1}{2} e_n^2 \right) \\ &= \tilde{T}_{k,\ell}^+ - \frac{1}{2} e_n^2 (\ell - k) =: \tilde{T}_{k,\ell}^-. \end{aligned} \tag{6}$$

We note that our upper and lower bounds on $\tilde{T}_{k,\ell}$ deviate only by $\tilde{T}_{k,\ell}^+ - \tilde{T}_{k,\ell}^- = \frac{1}{2} e_n^2 (\ell - k)$. Together with Theorem 13, we have proven the following estimates for $T_{k,\ell}$, which are tight apart from a term of order $O(\ell - k)$.

Theorem 14. *The expected number of iterations the (1 + 1) EA optimizing ONEMAX, started with a search point of fitness k , takes to find a search point with fitness ℓ or larger, satisfies*

$$\begin{aligned} e_n n \sum_{i=n-\ell+1}^{n-k} \frac{1}{i} - (\ell - k - 1) e(e-1) \exp\left(\frac{k}{n-1}\right) - \frac{1}{2} e_n^2 (\ell - k) \\ \leq T_{k,\ell} \leq \\ e_n n \sum_{i=n-\ell+1}^{n-k} \frac{1}{i}, \end{aligned}$$

where $e_n := \left(1 - \frac{1}{n}\right)^{-(n-1)}$.

We recall from above that $e(1 - \frac{1}{n}) \leq e_n \leq e$. We add that for $\ell < n$, the sum $\sum_{i=n-\ell+1}^{n-k} \frac{1}{i}$ is well-approximated by $\ln(\frac{n-k}{n-\ell})$, e.g., $\ln(\frac{n-k}{n-\ell}) - 1 <$

$\sum_{i=n-\ell+1}^{n-k} \frac{1}{i} < \ln\left(\frac{n-k}{n-\ell}\right)$ or $\sum_{i=n-\ell+1}^{n-k} \frac{1}{i} = \ln\left(\frac{n-k}{n-\ell}\right) - O\left(\frac{1}{n-\ell}\right)$, see, e.g., [Doe20b, Section 1.4.2]. For $\ell = n$, we have $\ln(n-k) < \sum_{i=n-\ell+1}^{n-k} \frac{1}{i} \leq \ln(n-k) + 1$ and $\sum_{i=n-\ell+1}^{n-k} \frac{1}{i} = \ln(n-k) + O\left(\frac{1}{n-k}\right)$.

When starting the $(1+1)$ EA with a random initial search point, the following bounds apply.

Theorem 15. *There is an absolute constant K such that the expected run time $T = T_{\text{rand},n}$ of the $(1+1)$ EA with random initialization on ONEMAX satisfies*

$$e_n n \sum_{i=1}^{\lceil n/2 \rceil} \frac{1}{i} - 4.755n - K \leq T \leq e_n n \sum_{i=1}^{\lceil n/2 \rceil} \frac{1}{i} + K.$$

In particular,

$$en \ln(n) - 4.871n - O(\log n) \leq T \leq en \ln(n) - 0.115n + O(1).$$

Proof. By [DD16, Theorem 2], the expected run time of the $(1+1)$ EA with random initialization on ONEMAX differs from the run time when starting with a search point on level A_M , $M := \lfloor n/2 \rfloor$, by at most a constant. Hence we have $T \leq T_{M,n} + O(1) \leq \tilde{T}_{M,n}^+ + O(1) = e_n n \sum_{i=1}^{\lceil n/2 \rceil} \frac{1}{i} + O(1)$ by Theorem 14.

For the lower bound, we use Equation (3) in the proof of Theorem 13, which is slightly tighter than the result stated in the theorem itself. Together with (6), we estimate

$$\begin{aligned} T &\geq T_{M,n} - O(1) \\ &\geq \tilde{T}_{M,n} - (n-1) \exp\left(\frac{n+M-1}{n-1}\right) (\exp\left(\frac{n-M-1}{n-1}\right) - 1) - O(1) \\ &\geq \tilde{T}_{M,n}^+ - \frac{1}{2} e_n^2 (n-M) - n e^{1.5} (e^{0.5} - 1) - O(1) \\ &= \tilde{T}_{M,n}^+ - \frac{1}{4} e^2 n - n(e^2 - e^{1.5}) - O(1) \\ &= \tilde{T}_{M,n}^+ - n\left(\frac{5}{4} e^2 - e^{1.5}\right) - O(1) \geq \tilde{T}_{M,n}^+ - 4.755n - O(1). \end{aligned}$$

The second set of estimates stems from noting that $\tilde{T}_{M,n}^+ = e_n n \sum_{i=1}^{\lceil n/2 \rceil} \frac{1}{i} = e_n n (\ln(\lceil n/2 \rceil) + \gamma \pm O(\frac{1}{n})) = e(1 - O(\frac{1}{n})) n (\ln n - \ln 2 + \gamma \pm O(\frac{1}{n}))$, where $\gamma = 0.5772156649 \dots$ is the Euler-Mascheroni constant. \square

Let us comment a little on the tightness of our result. Due to the symmetries in the ONEMAX process, the probability to leave the i -th fitness level is independent of the particular search point $x \in A_i$ the current parent is equal to. Consequently, in principle, Theorems 9 and 8 give the exact bound

$$E[T] = \sum_{k=0}^{n-1} 2^{-n} \binom{n}{k} \sum_{i=k}^{n-1} \frac{v_i |k}{p_i},$$

where $v_{i|k}$ denotes the probability that the process started on level k visits level i .

The reason why we cannot avoid a gap of order $\Theta(n)$ in our bounds is that computing the $v_{i|k}$ and p_i precisely is very difficult. Let us regard the $v_{i|k}$ first. It is easy to see that states i with $k < i \leq (1 - \varepsilon)n$, ε a positive constant, have a positive chance of not being visited: By Lemma 12, with probability $\Omega(1)$ level $i - 1$ is visited and from there, again with probability $\Omega(1)$, a two-bit flip occurs that leads to level $i + 1$. Since with constant probability the last level visited below level i is not $i - 1$, and since skipping level i conditional on the last level below i being at most $i - 2$ is, by a positive constant, less likely that skipping level i when on level $i - 1$ before (that is, $\frac{p_{i-2, \geq i+1}}{p_{i-2, \geq i}} \leq \frac{p_{i-1, \geq i+1}}{p_{i-1, \geq i}} - \Omega(1)$, we omit a formal proof of this statement), our estimate $q_{i|k} \leq \max_{j \in [k..i-1]} \frac{p_{j, \geq i+1}}{p_{j, \geq i}}$ already leads to a constant factor loss in the estimate of the q_i , which translates into a $\Theta(n)$ contribution to the gap of our lower bound from the truth. To overcome this, one would need to compute $q_{i|k} = \sum_{j=k}^{i-1} Q_{j|k} \frac{p_{j, \geq i+1}}{p_{j, \geq i}}$ precisely, where $Q_{j|k}$ is the probability that level j is the highest level visited below i in a process started on level k . This appears very complicated.

The second contribution to our $\Theta(n)$ gap is the estimate of p_i . We need a lower bound on p_i both in the estimate of the run time advantage due to not visiting all levels (see Equation (3)) and in the estimate of the run time estimate stemming from the fitness level method (4). Since the q_i are $\Omega(1)$ when $i \leq (1 - \varepsilon)n$, a constant-factor misestimation of the p_i leads to a $\Theta(n)$ contribution to the gap. Unfortunately, it is hard to avoid a constant-factor misestimation of the p_i , $i \leq (1 - \varepsilon)n$. Our estimate $p_i \geq (1 - \frac{1}{n})^{n-1} \frac{n-i}{n}$ only regards the event that the i -th level is left (to level $i + 1$) by flipping exactly one zero-bit into a one-bit. However, for each constant j the event that level $i + 1$ is reached by flipping $j + 1$ zero-bits and j one-bits has a constant probability of appearing. Moreover, for each constant j the event that level i is left to level $i + j$ also has a constant probability. For these reasons, a precise estimate of the p_i appears rather tedious.

In summary, we feel that our method quite easily gave a run time estimate precise apart from terms of order $O(n)$, but for more precise results drift analysis [Len20] might be the better tool (though still the relatively precise estimate of the expected progress from a level $i \leq (1 - \varepsilon)n$, which will necessarily be required for such an analysis, will be difficult to obtain).

5.3 Comparison with the Literature

We end this section by giving an overview on the previous works analyzing the run time of the $(1+1)$ EA on ONEMAX and comparing them to our result. Some of the results described in the following, in particular, Sudholt’s lower bound [Sud13], were also proven for general mutation rates p instead of only $p = \frac{1}{n}$. To ease the comparison with our result, we only state the results for the case that $p = \frac{1}{n}$. We note that with our method we could also have analysed broader ranges of mutation rates. The resulting computations, however, would have been more complicated and would have obscured the basic application of our method.

To the best of our knowledge, the first to state and rigorously prove a run time bound for this problem was Rudolph in his dissertation [Rud97, p. 95], who showed that $T = T_{\text{rand},n}$ satisfies $E[T] \leq (1 - \frac{1}{n})^{n-1} n \sum_{i=1}^n \frac{1}{i}$, which is exactly the upper bound $\tilde{T}_{0,n}^+$ from the fitness level method and from only regarding the events that levels are left via one-bit flips. A lower bound of $n \ln(n) - O(n \log \log n)$ was shown in [DJW98] for the optimization of a general separable function with positive weights when starting in the search point $(0, \dots, 0)$. From the proof of this result, it is clear that it holds for any pseudo-Boolean function with unique global optimum $(1, \dots, 1)$. This lower bound builds on the argument that each bit needs to be flipped at least once in some mutation step. It is not difficult to see that the expected time until this event happens is indeed $(1 \pm o(1))n \ln n$, so this argument is too weak to make the leading constant of $E[T]$ precise.

Only a very short time after these results and thus quite early in the young history of run time analysis of evolutionary algorithms, Garnier, Kallel, and Schoenauer [GKS99] showed that $E[T] = en \ln(n) + c_1 n + o(n)$ for a constant $c_1 \approx -1.9$, however, the completeness of their proof has been doubted in [HPR⁺18]. Since at that early time precise run time analyses were not very popular, it took a while until Doerr, Fouz, and Witt [DFW10] revisited this problem and showed with $E[T] \geq (1 - o(1))en \ln(n)$ the first lower bound that made the leading constant precise. Their proof used a variant of additive drift from [Jäg07] together with the potential function $\ln(Z_t)$, where Z_t denotes the number of zeroes in the parent individual at time t . Shortly later, Sudholt [Sud10] (journal version [Sud13]) used his fitness level method for lower bounds to show $E[T] \geq en \ln(n) - 2n \log \log n - 16n$. That the run time was $E[T] = en \ln(n) - \Theta(n)$ was proven first in [DFW11], where an upper bound of $en \ln(n) - 0.1369n + O(1)$ ¹ was shown via variable drift for upper bounds [MRC09, Joh10] and a lower bound of $E[T] \geq en \ln(n) - O(n)$

¹The constant 0.1369 was wrongly stated as 0.369 as pointed out in [LW14]

was shown via a new variable drift theorem for lower bounds on hitting times. An explicit version of the lower bound of $en \ln(n) - 7.81791n - O(\log n)$ and an alternative proof of the upper bound $en \ln(n) - 0.1369n + O(1)$ was given in [LW14] via a very general drift theorem.

The final answer to this problem was given in an incredibly difficult work by Hwang, Panholzer, Rolin, Tsai, and Chen [HPR⁺18] (see [HW19] for a simplified version), who showed $E[T] = en \ln(n) + c_1n + \frac{1}{2}e \ln(n) + c_2 + O(n^{-1} \log n)$ with explicit constants $c_1 \approx -1.9$ and $c_2 \approx 0.6$.

In the light of these results, we feel that our proof of an $en \ln(n) \pm O(n)$ bound is the first simple proof a run time estimate of this precision for this problem. Interestingly, our explicit lower bound $en \ln(n) - 4.871n - O(\log n)$ is even a little stronger than the bound $en \ln(n) - 7.81791n - O(\log n)$ proven with drift methods in [LW14].

6 Jump Functions

In this section, we regard jump functions, which comprise the most intensively studied benchmark in the theory of randomized search heuristics that is not unimodal and which has greatly aided our understanding how different heuristics cope with local optima [DJW02, JW02, Leh10, DLMN17, COY17, COY18, DFK⁺16, DFK⁺18, WVHM18, LOW19, RA19, Doe20a, ADK20, AD20, ABD21, RW20, RW21b, RW21a, DZ21, BBD21].

For all representation lengths n and all $k \in [1..n]$, the *jump function* with jump size k is defined by

$$\text{JUMP}_{n,k}(x) = \begin{cases} \|x\|_1 + k & \text{if } \|x\|_1 \in [0..n - k] \cup \{n\}, \\ n - \|x\|_1 & \text{if } \|x\|_1 \in [n - k + 1..n - 1], \end{cases}$$

for all $x \in \{0, 1\}^n$. Jump functions have a fitness landscape isomorphic to ONEMAX, except on the fitness valley or gap

$$G_{n,k} := \{x \in \{0, 1\}^n \mid n - k < \|x\|_1 < n\},$$

where the fitness is low and deceptive (pointing away from the optimum).

For simple elitist heuristics, not surprisingly, the time to find the optimum is strongly related to the time to cross the valley of low fitness. For the $(1 + 1)$ EA with mutation rate $\frac{1}{n}$, the probability to generate the optimum from a search point on the local optimum $L = \{x \in \{0, 1\}^n \mid \|x\|_1 = n - k\}$ is $p_k = (1 - \frac{1}{n})^{n-k} n^{-k}$, and hence the expected time to cross the valley of low fitness is $\frac{1}{p_k}$.

The true expected run time deviates slightly from this value, both because some time is spent to reach the local optimum and because the algorithm may

be lucky and not need to cross the valley or not in its full width. The first aspect, making additive terms of order at most $O(n \log n)$ more precise, can be treated with arguments very similar to the ones of the previous section, so we do not discuss this here. More interestingly appears the second aspect. In particular for larger values of k , the algorithm has a decent chance to start in the fitness valley. It is clear that even when starting in the valley, the deceptive nature of the valley will lead the algorithm rather towards the local optimum. We show now how our argumentation via omitted fitness levels allows to prove very precise bounds with elementary arguments. In principle, we could also use the our fitness level theorem, but since we shall regard only the single level $N_{n,k} = \{x \in \{0, 1\}^n \mid \|x\|_1 \in [0..n - k]\}$, we shall not make this explicit and simply use the classic typical-run argument (that except with some probability q , a state is reached from which the expected run time is at least some t , and that this gives a lower bound of $(1 - q)t$ for the expected run time).

The two previous analyses of the run time of the $(1 + 1)$ EA on jump functions deal with the problem of starting in the valley in a different manner. In [DJW02], it is argued that with probability at least $\frac{1}{2}$, the initial search point has at most $\frac{n}{2}$ ones. In case the initial search point is nevertheless in the gap region (because $k > \frac{n}{2}$), then with high probability a ONEMAX-style optimization process will reach the local optimum with high probability in time $O(n^2)$ except when in this period the optimum is generated. Since all parent individuals in this period have Hamming distance at least $\frac{n}{2}$ from the optimum, the probability for this exceptional event is exponentially small. This argument proves an $\Omega(\frac{1}{p_k})$ bound for the expected run time, and this for all values of $k \geq 2$. In [DLMN17], only the case $k \leq \frac{n}{2}$ was regarded and it was exploited that in this case, the probability for the initial search point to be in the gap (or the optimum) is only $2^{-n} \binom{n}{\leq k-1}$. This gives a lower bound of $(1 - 2^{-n} \binom{n}{\leq k-1}) \frac{1}{p_k}$, which is tight including the leading constant for $k \in [2.. \frac{n}{2} - \omega(\sqrt{n})]$.

We now show that estimating the probability of never reaching a search point x with $\|x\|_1 \leq n - k$ is not difficult with arguments similar to the ones used in the previous section. We need a slightly different approach since now the probability to skip a fitness level is not maximal when closest to this fitness level (the probability to skip $N_{n,k}$ is maximal when in the lowest fitness level, which is in Hamming distance $k - 1$ from $N_{n,k}$). Interestingly, we obtain very tight bounds which could be of some general interest, namely that the probability to never reach a point x with $\|x\|_1 \leq n - k$ is $O(\frac{1}{n})$, when allowing an arbitrary initialization (different from the global optimum), and is only $O(2^{-n})$ when using the usual random initialization.

Theorem 16. *Let $n \in \mathbb{N}$ and $k \in [2..n]$. Consider a run of the $(1+1)$ EA with mutation rate $p = \frac{1}{n}$ on the jump function $\text{JUMP}_{n,k}$. Denote by $N := N_{n,k} = \{x \in \{0,1\}^n \mid \|x\|_1 \in [0..n-k]\}$ the set of non-optimal solutions that lie not in the gap region of the jump function and by $p_k = (1 - \frac{1}{n})^{n-k} n^{-k}$ the probability to generate the optimum from a solution on the local optimum.*

- (i) *Assume that the $(1+1)$ EA starts with an arbitrary solution different from the global optimum. Then with probability $1 - O(\frac{1}{n})$, the algorithm reaches a search point in N . Consequently, the expected run time is at least $(1 - O(\frac{1}{n}))p_k^{-1}$.*
- (ii) *Assume that the $(1+1)$ EA starts with a random initial solution. Then with probability $1 - O(2^{-n})$, the algorithm reaches a search point in N . Consequently, the expected run time is at least $(1 - O(2^{-n}))p_k^{-1}$.*

Proof. Denote by f the jump function $\text{JUMP}_{n,k}$. We consider the partition of the search space into the fitness levels of the gap as well as N and the optimum. Hence let

$$\begin{aligned} A_j &:= \{x \in \{0,1\}^n \mid f(x) = j\} \text{ for } j \in [1..k-1], \\ A_k &:= \{x \in \{0,1\}^n \mid f(x) \in [k..n]\} = N, \\ A_{k+1} &:= \{(1, \dots, 1)\}. \end{aligned}$$

Our first claim is that, regardless of the initialization as long as different from the optimum, the probability q_k that the algorithm never has the parent individual in A_k is $O(\frac{1}{n})$. Since we start the algorithm with a non-optimal search point, the only way the algorithm can avoid A_k is by generating from a parent in A_j , $j \in [1..k-1]$, the global optimum. Denote by r_j the probability that the algorithm, if the current search point is in A_j , in the remaining run generates the optimum from a search point in A_j . Then, as just discussed, by the union bound,

$$q_k \leq \sum_{j=1}^{k-1} r_j. \tag{7}$$

The probability r_j is exactly the probability that in the iteration in which from a search point in A_j a better individual is generated, this is actually the global optimum. Hence $r_j = \Pr[y = (1, \dots, 1) \mid f(y) > j]$, where y is a mutation offspring generated from a search point in A_j . We compute

$$\begin{aligned} r_j &= \Pr[y = (1, \dots, 1) \mid f(y) > j] = \frac{\Pr[y = (1, \dots, 1)]}{\Pr[f(y) > j]} \\ &\leq \frac{n^{-j}}{(1 - \frac{1}{n})^{n-1} \frac{n-j}{n}} \leq \frac{e}{n^{j-1}(n-j)}, \end{aligned}$$

where we estimated the probability to generate a search point with fitness better than j by the probability of the event that a single one is flipped into a zero. Consequently, $q_k \leq \sum_{j=1}^{k-1} r_j \leq \sum_{j=1}^{n-1} \frac{e}{n^j(n-j)} = O(\frac{1}{n})$.

Once a search point in A_k is reached, the remaining run time dominates a geometric distribution with success probability $p_k = (1 - \frac{1}{n})^{n-k} n^k$, simply because each of the following iterations (before the optimum is found) has at most this probability of generating the optimum; hence the expected remaining run time is at least $\frac{1}{p_k}$. This shows that the expected run time of the $(1 + 1)$ EA started with any non-optimal search point is at least $(1 - q_k) \frac{1}{p_k}$.

For the case of a random initialization, we proceed in a similar manner, but also use the trivial observation that to skip the fitness range N by jumping from A_j , $j \in [1..k-1]$, right into the optimum, it is necessary that the algorithm visits A_j . To visit A_j , it is necessary that the initial search point lies in $A_1 \cup \dots \cup A_j$, which happens with probability $2^{-n} \sum_{i=1}^{j-1} \binom{n}{i}$ only. This, together with the observation that the only other way to avoid A_k is that the initial individual is already the optimum, gives

$$q_k \leq \sum_{j=1}^{k-1} \left(2^{-n} \sum_{i=1}^j \binom{n}{i} \right) r_j + 2^{-n}.$$

Using a tail estimate for binomial distributions (equation (VI.3.4) in [Fel68], also to be found as (1.10.62) in [Doe20b]), we bound $\sum_{i=1}^j \binom{n}{i} \leq 1.5 \binom{n}{j}$ for all $j \leq \frac{1}{4}n$. We also note from (7) that $r_j \leq 4n^{-j}$ in this case. For $j \geq \frac{1}{4}n$, we trivially have $\sum_{i=1}^j 2^{-n} \binom{n}{i} r_j \leq r_j \leq er^{-(j-1)}$. Consequently,

$$\begin{aligned} q_k &\leq \sum_{j=1}^{n-1} \left(2^{-n} \sum_{i=1}^j \binom{n}{i} \right) r_j + 2^{-n} \\ &\leq 2^{-n} 1.5 \sum_{j=1}^{\lfloor n/4 \rfloor} \binom{n}{j} r_j + \sum_{j=\lceil n/4 \rceil}^{n-1} en^{-(j-1)} + 2^{-n} \\ &\leq 2^{-n} 1.5 \sum_{j=0}^{\lfloor n/4 \rfloor} \frac{n^j}{j!} 4n^{-j} + O(n^{-(n/4)+1}) \leq 2^{-n} 6e + O(n^{-(n/4)+1}). \end{aligned}$$

Hence, as above, the expected run time is at least $(1 - q_k) \frac{1}{p_k} = (1 - O(2^{-n})) \frac{1}{p_k}$. \square

We note that the $O(\frac{1}{n})$ term in the bound for arbitrary initialization cannot be avoided in general, simply because when starting with a search point that is a neighbor of the optimum, the first iteration with probability

at least $\frac{1}{en}$ generates the optimum. The $O(2^{-n})$ term in the bound for random initialization is apparently necessary because with probability 2^{-n} .

We also note that we did not optimize the implicit constants in the $O(\frac{1}{n})$ and $O(2^{-n})$ term. With more care, these could be replaced by $(1 + o(1))\frac{1}{e-1}\frac{1}{n}$ and $(1 + o(1))\frac{e}{e-1}2^{-n}$, respectively.

7 A Bound for Long k -Paths

Long k -paths, introduced in [Rud96], have been studied in various places; we point the reader to [Sud09] for a discussion, which also contains the formalization that we use. A lower bound for long k -paths using FLM with viscosities was given in [Sud13].

We use [Sud09, Lemma 3] (phrased as a definition below) and need to know no further details about what a long k -path is. In fact, our proof uses all the ideas of the proof of [Sud13], but cast in terms of our FLM with visit probabilities, which, we believe, makes the proof simpler and the core ideas given by [Sud13] more prominent. Note that [Sud13] first needs to extend the FLM with viscosities by introducing an additional parameter before it is applicable in this case.

Definition 17. *Let k, n be given such that k divides n . A long k -path is function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ such that*

- *The 0-bit string has a fitness of 0; there are $m = k2^{n/k} - k$ bit strings of positive fitness, and all these values are distinct; all other bit strings have negative fitness. We call the bit strings with non-negative fitness as being on the path and consider them ordered by fitness (this way we can talk about the “next” element on the path and similar).*
- *For each bit string with non-negative fitness and each $i < k$, the bit string with i -next higher fitness is exactly a Hamming distance of i away.*
- *For each bit string with non-negative fitness and each $i \geq k$, the bit string with i -next higher fitness is at least a Hamming distance of k away.*

For an explicit construction of a long k -path, see [DJW02, Sud09]. The long k -paths are designed such that optimization proceeds by following the (long) path and true shortcuts are unlikely, since they require jumping at least k .

The following lower bound for optimizing long k -paths with the $(1 + 1)$ EA is given in [Sud13]. Note that n is the length of the bit strings, m is the length of the path and p is the mutation rate.

$$m \frac{1 - 2p}{p(1 - p)^n} \frac{1 - 2p}{1 - p} \left(1 - \left(\frac{p}{1 - p} \right)^k \right)^m. \quad (8)$$

We want to show here that we can derive the essentially same bound with the same ideas but less technical details.

Note that the lower bound given in [Sud13] is only meaningful for $k \geq \sqrt{n/\log(1/p)}$, as the last term of the bound would otherwise be close to 0:

$$\begin{aligned} \left(1 - \left(\frac{p}{1 - p} \right)^k \right)^m &\leq (1 - p^k)^m \leq \exp(-mp^k) \\ &\leq \exp(-2^{n/k} p^k) = \exp(-2^{n/k - k \log(1/p)}). \end{aligned}$$

We have that $n/k - k \log(1/p)$ is positive if and only if $n/\log(1/p) \geq k^2$.

In fact, if $k = \omega\left(\sqrt{n/\log(1/p)}\right)$, we have

$$\begin{aligned} \left(1 - \left(\frac{p}{1 - p} \right)^k \right)^m &\geq \left(1 - (2p)^k \right)^m \\ &\geq 1 - m(2p)^k \\ &\geq 1 - 2^{3n/k - k \log(1/p)} \\ &= 1 - 2^{\sqrt{n}o(\log(1/p)) - \sqrt{n}\omega(\log(1/p))} \\ &= 1 - 2^{-\sqrt{n}\omega(\log(1/p))} \\ &\geq 1 - 2^{-\sqrt{n}}. \end{aligned}$$

This also entails

$$p \leq \exp(-n/k^2).$$

With our fitness level method, we obtain the following lower bound. It differs from Sudholt's bound (8) by an additional term m , which reduces the lower bound. Analyzing why this term does not appear in Sudholt's analysis, we note that the $\gamma_{i,j}$ chosen in [Sud13] are underestimating the true probability to jump to elements of the path that are more than k steps (on the path) away. When this is corrected, as confirmed to us by the author, Sudholt's proof would also only show our bound below. Consequently, there is currently no proof for (8).

Theorem 18. Consider the $(1+1)$ EA on a long k -path of length m with mutation rate $p \leq 1/2$ starting at the all-0 bit string (the start of the path).²

Let T be the (random) time for the $(1+1)$ EA to find the optimum. Then

$$E[T] \geq m \frac{1-2p}{p(1-p)^n} \frac{1-2p}{1-p} \left(1 - m \left(\frac{p}{1-p} \right)^{k-1} \right)^m.$$

Proof. We are setting up to apply Theorem 8. We partition the search space in the canonical way such that, for all $i \leq m$ with $i > 0$, A_i contains the only i -th point of the path and nothing else, and A_0 contains all points not on the path. In order to simplify the analysis, we will first change the behavior of the algorithm such that it discards any offspring which differs from its parent by at least k bits. This will allow us to apply Theorem 8 quickly and cleanly, afterwards we will show that the progress of this modified algorithm is very close to the progress of the original algorithm.

In this modified process, we first consider the probability p_i to leave a given level $i < m$. For this, the algorithm has to jump up exactly $j < k$ fitness levels, which is achieved by flipping a specific set of j bits; the probability for this is

$$\begin{aligned} p_i &= \sum_{j=1}^{k-1} p^j (1-p)^{n-j} \leq (1-p)^n \sum_{j=1}^{\infty} \left(\frac{p}{1-p} \right)^j \\ &= (1-p)^n \frac{p/(1-p)}{1-p/(1-p)} \\ &= p(1-p)^n \frac{1}{1-2p}. \end{aligned}$$

Next we consider the probability v_i to visit a level i . We want to apply Lemma 10, so let some $x \in A_{<i}$ be given, on level $\ell(x)$. Let $d = i - \ell(x)$. Note that d is the Hamming distance between x and the unique point in A_i . Thus, in case of $d \geq k$, we have $\Pr[x \rightarrow A_i] = 0$, so suppose $d < k$. Then we have

$$\begin{aligned} \Pr \left[x \rightarrow A_i \mid x \rightarrow \bigcup_{j=i}^m A_j \right] &= \frac{\Pr[x \rightarrow A_i]}{\Pr[x \rightarrow \bigcup_{j=i}^m A_j]} \\ &= \frac{p^d (1-p)^{n-d}}{\sum_{j=d}^k p^j (1-p)^{n-j}} \\ &= \frac{1}{\sum_{j=d}^k p^{j-d} (1-p)^{d-j}} \end{aligned}$$

²This simplifying assumption about the start point was also made in [Sud13].

$$\begin{aligned}
&= \frac{1}{\sum_{j=0}^{k-d} p^j (1-p)^{-j}} \\
&\geq \frac{1}{\sum_{j=0}^{\infty} p^j (1-p)^{-j}} \\
&= 1 - \frac{p}{1-p} \\
&= \frac{1-2p}{1-p}.
\end{aligned}$$

By Lemma 10, we can use this last term as v_i in Theorem 8 (it also fulfills the second condition of Lemma 10, since the process starts deterministically in the 0 string). Note that neither p_i nor v_i depends on i . Using Theorem 8 and recalling that we have m levels, we get a lower bound of

$$m \frac{1-2p}{p(1-p)^n} \frac{1-2p}{1-p}.$$

Note that this is exactly the term derived in [Sud13] except for a term correcting for the possibility of jumps of more than k bits, which we also still need to correct for.

We now show that this probability of making a successful jump of distance at least k is small. To that end we will show that it is very unlikely to leave a fitness level with a large jump rather than just move to the next level.

Suppose the algorithm is currently at $x \in A_i$. Leaving x with a jump of at least k to a specific element on the path is less likely the longer the jump is (since $p \leq 1/2$). Thus, we can upper bound the probability of jumping to an element of the path which is more than k away as $p^k(1-p)^{n-k}$. Thus, conditional on leaving the fitness level, the probability of leaving it with a $\geq k$ -jump is

$$\begin{aligned}
\Pr[x \rightarrow A_{\geq i+k} \mid x \rightarrow A_{> i}] &= \frac{\Pr[x \rightarrow A_{\geq i+k}]}{\Pr[x \rightarrow A_{> i}]} \\
&\leq \frac{mp^k(1-p)^{n-k}}{p(1-p)^{n-1}} \\
&= m \left(\frac{p}{1-p} \right)^{k-1}.
\end{aligned}$$

Thus, the probability of never making an accepted jump of at least k is bounded from below by the probability to, independently once for each of the m fitness levels, leave the fitness level with a 1-step rather than a jump of at least k :

$$\left(1 - m \left(\frac{p}{1-p} \right)^{k-1} \right)^m.$$

By pessimistically assuming that the process takes a time of 0 in case it ever makes an accepted jump of at least k , we can lower-bound the expected time of the original process to reach the optimum as the product of the expected time of the modified process times the probability to never make progress of k or more. \square

8 Conclusion

In this work, we proposed a simple and natural way to prove lower bounds via fitness level arguments. The key to our approach is that the true run time can be expressed as the sum of the waiting times to leave a fitness level, weighted with the probability that this level is visited at all. When applying this idea, usually the most difficult part is estimating the probabilities to visit the levels, but as our examples `LEADINGONES`, `ONEMAX`, jump functions, and long paths show, this is not overly difficult and clearly easier than setting correctly the viscosity parameters of the previous fitness level method for lower bounds. For this reason, we are optimistic that our method will be an effective way to prove other lower bounds in the future, most easily, of course, for problems where upper bounds were proven via fitness level arguments as well.

Our method makes most sense for elitist evolutionary algorithms even though by regarding the best-so-far individual any evolutionary algorithm gives rise to a non-decreasing level process (at the price that the estimates for the level leaving probabilities become weaker). We are optimistic that our method can be extended to non-elitist algorithms, though. We note that the level visit probability v_i for an elitist algorithm is equal to the expected number of separate visits to this level (simply because each level is visited exactly once or never). When defining the v_i as the expected number of times the i -th level is visited, our upper and lower bounds of Theorems 8 and 9 remain valid (the proof would use Wald's equation). We did not detail this in our work since our main focus were the elitist examples regarded in [Sud13], but we are optimistic that this direction could be interesting to prove lower bounds also for non-elitist algorithms.

References

- [ABD20] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. First steps towards a runtime analysis when starting with a good so-

- lution. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*, pages 560–573. Springer, 2020.
- [ABD21] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. Lazy parameter tuning and control: choosing all parameters randomly from a power-law distribution. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM, 2021.
- [AD20] Denis Antipov and Benjamin Doerr. Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda))$ genetic algorithm on jump functions. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*, pages 545–559. Springer, 2020.
- [ADK20] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. The $(1 + (\lambda, \lambda))$ GA is even faster on multimodal problems. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1259–1267. ACM, 2020.
- [BBD21] Riade Benbaki, Ziyad Benomar, and Benjamin Doerr. A rigorous runtime analysis of the 2-MMAS_{ib} on jump functions: ant colony optimizers can cope well with local optima. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM, 2021.
- [BDDV20] Maxim Buzdalov, Benjamin Doerr, Carola Doerr, and Dmitry Vinokurov. Fixed-target runtime analysis. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, 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.
- [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.
- [COY17] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. On the runtime analysis of the Opt-IA artificial immune system. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 83–90. ACM, 2017.

- [COY18] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. Fast artificial immune systems. In *Parallel Problem Solving from Nature, PPSN 2018, Part II*, pages 67–78. Springer, 2018.
- [DD16] Benjamin Doerr and Carola Doerr. The impact of random initialization on the runtime of randomized search heuristics. *Algorithmica*, 75:529–553, 2016.
- [DDK18] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica*, 80:1732–1768, 2018.
- [DDY20] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science*, 801:1–34, 2020.
- [DFK⁺16] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima with diversity mechanisms and crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 645–652. ACM, 2016.
- [DFK⁺18] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22:484–497, 2018.
- [DFW10] Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Quasi-random evolutionary algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2010*, pages 1457–1464. ACM, 2010.
- [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.
- [DJW98] Stefan Droste, Thomas Jansen, and Ingo Wegener. A rigorous complexity analysis of the $(1 + 1)$ evolutionary algorithm for separable functions with boolean inputs. *Evolutionary Computation*, 6:185–196, 1998.

- [DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.
- [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*. ACM, 2021.
- [DKLL20] Benjamin Doerr, Timo Kötzing, J. A. Gregor Lagodzinski, and Johannes Lengler. The impact of lexicographic parsimony pressure for ORDER/MAJORITY on the run time. *Theoretical Computer Science*, 816:144–168, 2020.
- [DL16] Duc-Cuong Dang and Per Kristian Lehre. Runtime analysis of non-elitist populations: from classical optimisation to partial information. *Algorithmica*, 75:428–461, 2016.
- [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.
- [Doe19] Benjamin Doerr. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science*, 773:115–137, 2019.
- [Doe20a] Benjamin Doerr. Does comma selection help to cope with local optima? In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1304–1313. ACM, 2020.

- [Doe20b] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 1–87. Springer, 2020. Also available at <https://arxiv.org/abs/1801.06733>.
- [DZ21] Benjamin Doerr and Weijie Zheng. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, 2021. To appear.
- [Fel68] William Feller. *An Introduction to Probability Theory and Its Applications*, volume I. Wiley, third edition, 1968.
- [FK13] Matthias Feldmann and Timo Kötzing. Optimizing expected path lengths with ant colony optimization using fitness proportional update. In *Foundations of Genetic Algorithms, FOGA 2013*, pages 65–74. ACM, 2013.
- [GKS99] Josselin Garnier, Leila Kallel, and Marc Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7:173–203, 1999.
- [GW17] Christian Gießen and Carsten Witt. The interplay of population size and mutation probability in the $(1 + \lambda)$ EA on OneMax. *Algorithmica*, 78:587–609, 2017.
- [GW18] Christian Gießen and Carsten Witt. Optimal mutation rates for the $(1 + \lambda)$ EA on OneMax through asymptotically tight drift analysis. *Algorithmica*, 80:1710–1731, 2018.
- [HPR⁺18] Hsien-Kuei Hwang, Alois Panholzer, Nicolas Rolin, Tsung-Hsi Tsai, and Wei-Mei Chen. Probabilistic analysis of the $(1+1)$ -evolutionary algorithm. *Evolutionary Computation*, 26:299–345, 2018.
- [HW19] Hsien-Kuei Hwang and Carsten Witt. Sharp bounds on the runtime of the $(1+1)$ EA via drift analysis and analytic combinatorial tools. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 1–12. ACM, 2019.
- [HY01] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:51–81, 2001.

- [Jäg07] Jens Jägersküpper. Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theoretical Computer Science*, 379:329–347, 2007.
- [Joh10] Daniel Johannsen. *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, 2010.
- [JW02] Thomas Jansen and Ingo Wegener. The analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica*, 34:47–66, 2002.
- [JZ14] Thomas Jansen and Christine Zarges. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, 545:39–58, 2014.
- [Leh10] Per Kristian Lehre. Negative drift in populations. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 244–253. Springer, 2010.
- [Leh11] Per Kristian Lehre. Fitness-levels for non-elitist populations. In *Genetic and Evolutionary Computation Conference, GECCO 2011*, pages 2075–2082. ACM, 2011.
- [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>.
- [LOW19] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwick. On the time complexity of algorithm selection hyperheuristics for multimodal optimisation. In *Conference on Artificial Intelligence, AAI 2019*, pages 2322–2329. AAAI Press, 2019.
- [LS14] Jörg Lässig and Dirk Sudholt. General upper bounds on the runtime of parallel evolutionary algorithms. *Evolutionary Computation*, 22:405–437, 2014.
- [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.

- [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.
- [RA19] Jonathan E. Rowe and Aishwaryaprajna. The benefits and limitations of voting mechanisms in evolutionary optimisation. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 34–42. ACM, 2019.
- [Rud96] Günter Rudolph. How mutation and selection solve long path problems in polynomial expected time. *Evolutionary Computation*, 4:195–205, 1996.
- [Rud97] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kováč, 1997.
- [RW20] Amirhossein Rajabi and Carsten Witt. Self-adjusting evolutionary algorithms for multimodal optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2020*, pages 1314–1322. ACM, 2020.
- [RW21a] Amirhossein Rajabi and Carsten Witt. Stagnation detection in highly multimodal fitness landscapes. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM, 2021.
- [RW21b] Amirhossein Rajabi and Carsten Witt. Stagnation detection with randomized local search. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2021*, pages 152–168. Springer, 2021.
- [Sud09] Dirk Sudholt. The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410:2511–2528, 2009.
- [Sud10] Dirk Sudholt. General lower bounds for the running time of evolutionary algorithms. In *Parallel Problem Solving from Nature, PPSN 2010, Part I*, pages 124–133. Springer, 2010.
- [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.

- [Weg01] Ingo Wegener. Theoretical aspects of evolutionary algorithms. In *Automata, Languages and Programming, ICALP 2001*, pages 64–78. Springer, 2001.
- [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.
- [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.
- [WVHM18] Darrell Whitley, Swetha Varadarajan, Rachel Hirsch, and Anirban Mukhopadhyay. Exploration and exploitation without mutation: solving the jump function in $\Theta(n)$ time. In *Parallel Problem Solving from Nature, PPSN 2018, Part II*, pages 55–66. Springer, 2018.