A New Analysis Method for Evolutionary Optimization of Dynamic and Noisy Objective Functions

Raphaël Dang-Nhu
École Polytechnique
Palaiseau, France

Thibault Dardinier
École Polytechnique
Palaiseau, France

Benjamin Doerr
École Polytechnique
Laboratoire d'Informatique (LIX)
Palaiseau, France

Gautier Izacard
École Polytechnique
Palaiseau, France

Dorian Nogneng
École Polytechnique
Laboratoire d'Informatique (LIX)
Palaiseau, France

ABSTRACT

Evolutionary algorithms, being problem-independent and randomized heuristics, are generally believed to be robust to dynamic changes and noisy access to the problem instance. We propose a new method to obtain rigorous runtime results for such settings. In contrast to many previous works, our new approach mostly relies on general parameters of the dynamics or the noise models, such as the expected change of the dynamic optimum or the probability to have a dynamic change in one iteration. Consequently, we obtain bounds which are valid for large varieties of such models. Despite this generality, for almost all particular models regarded in the past our bounds are stronger than those given in previous works. As one particular result, we prove that the $(1 + \lambda)$ EA can optimize the OneMax benchmark function efficiently despite a constant rate of 1-bit flip noise. For this, a logarithmic size offspring population suffices (the previous-best result required a super-linear value of λ). Our results suggest that the typical way to find the optimum in such adverse settings is not via a steady approach of the optimum, but rather via an exceptionally fast approach after waiting for a rare phase of low dynamic changes or noise.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**; *Optimization with randomized search heuristics*;

KEYWORDS

Dynamic optimization; noisy optimization; runtime analysis

1 INTRODUCTION

Evolutionary algorithms (EAs) are randomized optimization heuristics. They are composed of generic building blocks inspired by natural evolution, which are suitably combined into an algorithm for a particular problem. Both the randomized nature and the fact that EAs contain many problem-unspecific ingredients suggest that EAs could be particularly suitable to cope with stochastic disturbances of the problem such as dynamically changing problem data or noisy objective functions. This belief has been verified in many experimental works, see the surveys [1, 12].

In contrast to a huge body of experimental literature, much fewer mathematical works analyze the performance of evolutionary algorithms in dynamic or noisy settings. Apparently, one reason are the complicated stochastic interactions between the algorithm and the dynamics or the noise. As an example, we note that already the simple problem of how the basic $(1 + 1)$ EA optimizes an n-dimensional dynamic OneMax function in which the optimum changes in each iteration by flipping each bit independently with probability $p' = c \ln(n)/n^2$, is not well understood (see the discussion of previous works for more details).

In this work, we propose a new approach to conduct runtime analyses for dynamic or noisy optimization problems. It builds on the following simple observation, which was made, e.g., for the dynamic OneMax problem, already in [13]. For many dynamic or noisy optimization processes, there is an easy early part of the process, in which the progress of the algorithm is not much hindered by the dynamics or the noise. This part is usually easy to analyze and does not contribute much to the total runtime. More interesting is the later, harder part of the optimization process, in which the average progress of the algorithm is less than the average damage caused by the dynamics or the noise. That an evolutionary algorithm can find the optimum in reasonable time at all despite the presence of such adverse conditions is a strong point for evolutionary algorithms. From the perspective of runtime analysis, however, this part is highly challenging since most of the classical methods either aim at translating a positive expected progress into a good runtime bound or a negative expected progress into a high lower bound.

We overcome the technical difficulties of this second part of the optimization process in a simple (but as we shall see, effective) manner: we try to make progress from the equilibrium point (where the algorithm progress and dynamics or noise damage cancel in expectation) only in short periods without dynamic changes or noise. More precisely, we argue that if for some t (i) with a certain probability p_1 no dynamic change or no noise occurs during t iterations and (ii) the undisturbed algorithm finds the optimum

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205563

starting from the equilibrium point in t iterations with probability at least p_2, then (roughly) $\frac{t}{p_1 p_2}$ iterations suffice to find the optimum in the dynamic or noisy setting.

Needless to say, this analysis method is wasteful in the sense that we disregard all ways to the optimum that do have some small disturbances on the way from the equilibrium point to the optimum. Nevertheless, the fact that with our approach we shall improve most existing results (see the discussion of our results in the separate sections on dynamic optimization and noisy optimization) shows that our approach despite this theoretical wastefulness is very strong.

The second big advantage of our new method is its universality. As the sketch of the main idea above shows, our analyses are relatively independent of the particular dynamics or the noise model. To determine the equilibrium point, we only need to understand the expected damage created by the dynamics or the noise. In the second part of the analysis, it suffices to know the probability that we have a dynamic change of the problem instance or a noisy fitness evaluation in one iteration.

The fact that most of our results are significantly stronger than the previous ones also suggests that our analysis approach of waiting for a rare phase of low activity of the dynamics or the noise and then exploiting this via an exceptionally fast approach of the optimum is possibly a realistic view on such processes. We support this with some preliminary experiments.

Organization of the paper: To ease the reading, we treat separately the problems of analyzing evolutionary algorithms in the presence of dynamically changing problem data and in the presence of noise. While the main idea of our new method is identical is both settings, sufficiently many details are different so that a unified treatment would become overly technical. We note that all previous runtime analyses as well discuss either dynamic optimization or noisy optimization. Each of the following two long sections on dynamic optimization and noisy optimization consists of a short description of previous and our results followed by a technical part containing precise statements and proof ideas.

2 DYNAMIC OPTIMIZATION

Many real-world optimization problems are inherently dynamic, that is, their problem data is changing at least mildly during the optimization process. Formally, this means that during the run of the algorithm, the problem instance to be solved can be replaced by a different problem instance. Usually it is assumed that the new instance is similar to the previous one and that these dynamic changes are governed by some simple randomized process.

2.1 Previous Work and Our Results

2.1.1 Previous Works. There is a long list of experimental works showing how to use evolutionary algorithms in dynamic optimization. For reasons of space, we refer to the surveys [2, 14]. There is by now also a moderate amount of theoretical results [3, 4, 8, 9, 13, 16]. The first runtime analyses on dynamic problems were conducted in the two groundbreaking papers by Droste [8, 9]. Both discuss how the (1 + 1) EA optimizes a dynamic version of the OneMax function. The (generalized) OneMax function $\textsc{OneMax}_z$ with optimum $z \in \{0,1\}^n$ is defined by $\textsc{OneMax}_z(x) = |\{i \in [1..n] \mid x_i = z_i\}|$. The dynamics regarded in both works of Droste is that after each iteration of the evolutionary algorithm with small probability the current problem instance $\textsc{OneMax}_z$ is changed to $\textsc{OneMax}_{z'}$, where z' is chosen randomly according to some given distribution. Since there is an obvious one-to-one correspondence between the $\textsc{OneMax}_z$ functions and their optima $z \in \{0,1\}^n$, we may conveniently talk of the *dynamics changing the optimum* (and this implies changing the whole instance).

In [8], the *1-bit dynamics* is regarded, which in each iteration with probability p replaces the current optimum z by a random Hamming neighbor z'. The main result of the paper is that if there exists a constant c such that $p \leq c\frac{\ln(n)}{n}$, then the *runtime* (also called *optimization time*) T of the (1 + 1) EA, that is, the number of fitness evaluations until the current optimum is evaluated for the first time, satisfies $E[T] = O(n^{ce+1+o(1)} \ln(n))$, where $e \approx 2.72$ is the Euler constant.

Since dynamics only moving to a Hamming neighbor are slightly artificial, Droste [9] analyzes the same problem with the *bitwise dynamics*. In this dynamics, in each iteration the current optimum z is replaced by a string z' obtained from z by flipping each bit independently with probability p'. If for some constant c we have $p' \leq c\frac{\ln(n)}{n^2}$, then the optimization time T satisfies $E[T] = O(n^{4ce+1+o(1)} \ln(n))$. Note that for identical values of c, both dynamics have the same average change of the optimum, but the runtime guarantee for the bitwise dynamics is almost the 4-th power of the one for the 1-bit dynamics.

There is no reason to believe that this larger bound on the runtime should be the truth, since for equal values of c, both dynamics are very similar: 1-bit flips appear with essentially the same rate $r_1 = (1 + o(1))c\ln(n)/n$, higher-order bit flips do not exist in the 1-bit model, but appear in the bitwise model also only with rate $O(\frac{\log^2 n}{n^2})$. So it seems that the significantly different runtime guarantees are caused by the methods used, namely an analysis of the precise Markov chain. Such analyses can give very good results (and in fact, the result of [8] is the one that we cannot improve), but they are also known to be very sensitive to small changes of the chain. Hence it is not surprising that different results are obtained for two very similar dynamics.

In [13], the first progress on this problem after more than ten years, the runtime guarantee for the bitwise dynamic is improved to $E[T] \leq 22n^{\alpha ec+2} \ln(n)$, where $\alpha = 3\frac{\ln(2e^2/3)}{e} \approx 1.76$. While this bound is significantly better than the one of [9] when c is not too small, it again remains above the one for the 1-bit dynamics and in particular cannot give sub-quadratic runtime guarantees. We speculate that again the general analysis method via drift analysis used in [13] is non-optimal. Drift analysis tries to translate information about the expected progress in a randomized process (*drift*) into a hitting time. Since in the hard part of the optimization process (beyond the equilibrium point) the drift is pointing away from the target, it appears contradictory to use drift analysis to derive upper bounds on the hitting time of the optimum. To make this possible, in [13] a clever rescaling of the fitness is invented which has the property that the drift with respect to this rescaled fitness is indeed directed towards the target. Nevertheless, it would not be surprising if this counter-intuitive approach would not give the best results.

2.1.2 Our Results. We propose a novel and possibly more natural approach to analyzing the runtime of evolutionary algorithms in dynamic settings. For the difficult part, in which the drift is away from the optimum, we exploit as only way to the optimum the fact that we have intervals without dynamic changes and that in these intervals the algorithm finds it easier to make progress. This remains not totally trivial, since the intervals without dynamic changes are relatively short, so we need to prove that the algorithm in these short intervals finds the optimum with some sufficiently significant probability. Since we are thus waiting for an exceptionally fast progress of the algorithm, we cannot use the traditional methods (e.g., drift analysis). Still, with this approach we only need to analyze the performance of the undisturbed algorithm, which appears to be feasible.

The big advantage of this method is that it mostly relies on the probability that no dynamic change happens, whereas the particular distribution of the dynamic change is less important. More precisely, we shall assume the quite general model that in each iteration, the dynamic change of the optimum is obtained from sampling a bitmask $m \in \{0,1\}^n$ from some arbitrary distribution D (the same in each iteration). Then z is replaced by $z' = z \oplus m$, the exclusive-or of z and m (in other words, we flip exactly those bits in z which are one in m). For a random sample m from D, let $E_D := E[\|m\|_1]$ be the expected number of bits the dynamics flips in z and let $p_D := \Pr[m \neq (0,\dots,0)]$ be the probability that a dynamic change appears in one iteration (that is, that $z \neq z'$).

For this general model, our main technical result on dynamic optimization is that for each $t \in \mathbb{N}$, the expected optimization time of the (1 + 1) EA on this dynamic OneMax function is

$$E[T] \leq en(\ln(n)+1) + \frac{t + neE_D}{(1-p_D)^t \left(1 - \left(1 - \frac{1}{en}\right)^t\right)^{\lceil enE_D \rceil}}.$$

Thus we obtain good runtime bounds for concrete dynamics by minimizing this expression over t taking into account the concrete values for E_D and p_D.

For both the 1-bit dynamics with $p \leq c\ln(n)/n$ and the bitwise dynamics with $p' \leq c\ln(n)/n^2$, we have $p_D \leq c\frac{\ln(n)}{n}$ and $E_D \leq c\frac{\ln(n)}{n}$. Hence optimizing over t gives the same runtime guarantee for both dynamics, namely

$$E[T] \leq (\gamma + o(1))n^{\beta ec+1}$$

with $\beta = 2\ln(2) \approx 1.39$ and $\gamma = 2e\ln(2) \approx 3.77$.

The above described method is not restricted to OneMax (or similar functions) and is also applicable to processes that do not have any equilibrium point (due to the fact that the damage caused by the dynamics outnumbers the progress of the algorithm in the whole search space). Here is an example. The (generalized) LeadingOnes function LeadingOnes_z with optimum $z \in \{0,1\}^n$ is defined by $\text{LeadingOnes}_z(x) = \max\{i \in [0..n] \mid \forall j \leq i, x_j = z_j\}$. With the same method as above, we easily prove that the 1-bit dynamics with $p \leq c\ln(n)/n^2$ and the bitwise dynamics with $p' \leq c\ln(n)/n^3$ both lead to an expected optimization time of the (1 + 1) EA of $E[T] \leq (\delta + o(1))n^{2+\delta c}$, where $\delta = \frac{e-1}{2} \approx 0.86$. To the best of our knowledge, no previous results exist on the optimization of dynamic LeadingOnes functions.

2.2 Runtime Analysis for Dynamic OneMax Functions

2.2.1 Precise Description of the Dynamic Model. In [8] and [9], Droste proposes two different dynamic OneMax functions. We shall regard the following significant extension. Let D be any distribution on $\{0,1\}^n$. We take as initial cost function c_0 some OneMax function OneMax_z. Inductively, we define a sequence of cost functions $(c_i : \{0,1\}^n \to \mathbb{N})_{i\in\mathbb{N}}$. If at step i the cost function c_i is OneMax_z, we sample m in $\{0,1\}^n$ from D and set $c_{i+1} = \text{OneMax}_{z\oplus m}$, where $\oplus$ is the exclusive-or. This model loosely resembles the XOR benchmark of Stanhope and Daida [17], with the difference that there a dynamic change was performed only each g generations (for a non-random g) and that the m always is a random bit string having exactly d ones (for some non-random d).

Our model defines two characteristic parameters, namely the probability p_D that a (non-trivial) dynamic change appears and the expected number E_D of bits flipped by the dynamics. More precisely, for a sample m from D, we have $p_D = \Pr[m \neq (0,\dots,0)]$ and $E_D = E[\|m\|_1]$. Our new analysis method will be such that our runtime guarantees only depend on these parameters. Note that for a concrete dynamic model, these numbers usually are easy to compute.

Our setting clearly includes the two dynamic models regarded by Droste. For the 1-bit dynamics, we sample m by choosing with probability p a random bit string with exactly one 1; otherwise we have $m = (0,\dots,0)$. We have $p_D = E_D = p$. For the bitwise dynamics, we generate $m = (m_1,\dots,m_n) \in \{0,1\}^n$ randomly such that, for each $i \in [1..n]$ independently, we have $\Pr[m_i = 1] = p'$. In that case, $E_D = np'$ and $p_D = 1 - (1-p')^n$. Note that when $p' = o(\frac{1}{n})$, and this will be the typical situation, then $p_D = (1 + o(1))np'$.

In [8], Droste proposed the following equivalent formulation for the random process arising from running the (1 + 1) EA on this dynamic OneMax model. Instead of modifying the optimum of the OneMax fitness function, he rather applies this modification to the current search point. If a and b are two bitstrings, let us denote $H(a,b)$ the Hamming distance between a and b. Since $H(x, z \oplus m) = H(x \oplus m, z)$, the effect on the fitness is the same, and this holds not only for Droste's dynamics, but for all dynamics that fit into our setting. For this reason, we can assume in the remainder that we always optimize the classic OneMax function and after each iteration replace the current search point x by $x \oplus m$ with $m \sim D$. For convenience in the analysis, as often when drift arguments are employed, we shall regard the minimization version of a OneMax function. Hence we consider the cost function $c(x) = H(x,(1,\dots,1))$. To fix the notation, we formulate the resulting randomized process as Algorithm 1.

The stochastic aspect of the algorithm makes $x = (x_i)_{i\in\mathbb{N}}$ a Markov chain. We denote by x_{2i} the state of the individual at the beginning of iteration i and by x_{2i+1} the individual after the iteration, but before the possible modification through the dynamics. We define T to be the hitting time of the process, i.e. $T = \inf\{i \geq 0 \mid x_{2i} = (1,\dots,1)\}$.

In what follows, we often have to consider probabilities or expectations conditioned on the initial search point.

Notation 1. *Let $y \in \{0,1\}^n$, we denote $\Pr_y[\,\cdot\,] = \Pr[\,\cdot \mid x_0 = y]$ and $E_y[\,\cdot\,] = E[\,\cdot \mid x_0 = y]$.*

Algorithm 1 Process equivalent to the (1 + 1) EA minimizing a dynamically changing cost function $(c_i)_{i\in\mathbb{N}}$

Choose $x_0 \in \{0,1\}^n$ uniformly at random
for $i = 0$ to ∞ **do**
  Sample y_i from x_{2i} by flipping each bit with probability $\frac{1}{n}$.
  if $c(y_i) \le c(x_{2i})$ **then**
    $x_{2i+1} := y_i$
  else $x_{2i+1} := x_{2i}$
  Sample m_i from D
  $x_{2i+2} := x_{2i+1} \oplus m_i$

2.2.2 Easy Process up to the Equilibrium Point. In this subsection, we use multiplicative drift analysis to show that the optimization process of the dynamic OneMax function quickly reaches the *equilibrium point* $k^* := \lceil neE_D \rceil$, which is (essentially) the smallest cost value k at which we still have a positive drift $E[c(x_{2i}) - c(x_{2i+2}) \mid c(x_{2i}) = k]$.

While it appears very natural that multiplicative drift is the right tool for a dynamic OneMax process up to the equilibrium point, transforming this intuition into a formal proof is not totally trivial. For the particular dynamic OneMax process with bitwise changes of the optimum, Theorem 8 in [13] analyzes this "easy" part of the optimization process via moment-generating functions.

We succeed with multiplicative drift, but we need some additional arguments to cope with the facts (i) that we use a general dynamic model described only by the parameters p_D and E_D, and (ii) that the target k^* of the process is not the minimum of the state space as required by the multiplicative drift theorem [7]. Still, the approach via multiplicative drift gives a reasonably easy proof and a very natural result—the time to reach the equilibrium point from an initial search point that is d cost levels away from the equilibrium is (apart from lower order terms) exactly the time needed to gain the last d fitness level in a static optimization of OneMax.

We identify two components of the drift with opposite effects. With mutation rate $\frac{1}{n}$, the typical progress of the (1 + 1) EA is made when exactly a 0 is flipped into a 1. It leads to a pessimistic, but reasonable, evaluation of the expectancy of progress to $\frac{k}{en}$. On the other hand, the cost of the search point increases when the dynamic effects flip 1s of the search point into 0s. The average number of such flips is not greater than E_D. Therefore, we obtain a lower bound of the negative component of the drift of $-E_D$. The formalization of this argument yields the following result.

Theorem 1. *Let $k^* = \lceil neE_D \rceil$ and $\tau_0 = \inf\{i \in \mathbb{N} \mid c(x_{2i}) \le k^*\}$. For all $x_0 \in \{0,1\}^n$ such that $c(x_0) > k^*$,*

$$E_{x_0}[\tau_0] \le en(\ln(c(x_0) - k^*) + 1) \le en(\ln(n) + 1).$$

2.2.3 Exceptionally Fast Progress from the Equilibrium Point to the Optimum in Periods Without Dynamic Changes. The second building block of our analysis method is an estimate of the probability that the algorithm starting at k^* finds the optimum in a relatively short period of time without dynamic changes. During these periods, the dynamic process is equivalent to the process of the (1 + 1) EA optimizing a static OneMax function. In the typical applications of our method, the length of this period without dynamic changes is much smaller than the expected time the (1+1) EA needs to go from the cost level k^* to the optimum. For this reason, we have to study the extreme lower tail of the distribution of static hitting time T_S.

To ease this analysis, we introduce below a modified process, which is easier to analyze and which, in the sense of stochastic domination (cf. [6]), is a pessimistic approximation of the dynamics-free optimization process on OneMax. The *new process* starts with the same search point as the original optimization process. The two processes are equal except that the new process as mutation operator never flips a 1 and flips each 0 independently with probability $\frac{1}{en}$.

By induction, we show that the new process dominates the original process in terms of the cost: If Z_i is the search point of the new process at time i and x_{2i} is the search point of the original process, then $c(Z_i) \ge c(x_{2i})$. Consequently, denoting by T_N the hitting time of the optimum of the new process, we have $T_N \ge T_S$. This hitting time of the new process is easy to analyze, which gives the following result.

Theorem 2. *For all $x \in \{0,1\}^n$ and $t \in \mathbb{N}$, we have*

$$\Pr_x[T_S \le t] \ge \Pr_x[T_N \le t] = (1 - (1 - \tfrac{1}{en})^t)^{c(x)}.$$

With this estimate of the distribution of the hitting time of the static process, we obtain a lower bound of the probability of success without dynamics in a short period of length t. We define the stopping time τ_1 as the length of the first short period without dynamics. The probability $\Pr_x[T \le \tau_1]$ to reach the optimum in this short period can be intuitively lower bounded by the product of the probability to have no dynamics in t steps, and the probability that the static process reaches the optimum in time less than t.

Definition 3. *For $t \in \mathbb{N}$, we denote*

$$\tau_1 = \min\{t, \inf\{i \ge 1 \mid x_{2i-1} \ne x_{2i}\}\}.$$

Theorem 4. *Let $t \in \mathbb{N}$, $x \in \{0,1\}^n$ such that $c(x) \le k^*$. We have*

$$\Pr_x[T \le \tau_1] \ge (1 - p_D)^t \cdot \left(1 - \left(1 - \tfrac{1}{en}\right)^t\right)^{k^*}.$$

2.2.4 Upper Bound on the Expected Hitting Time. In Theorem 5, we combine the different arguments to obtain an upper bound on the expected hitting time. The first term $en(\ln(n) + 1)$ corresponds to the expected time to reach the equilibrium point k^*, according to Theorem 1. The second term corresponds to the progress from k^*. To obtain it, the core argument of the proof is to consider successive periods without dynamics, and to estimate how much of these are necessary, in average, to have a success. Similarly to geometrical distributions, the obtained expected hitting time is inversely proportional to the probability of reaching the optimum in one of these periods. Formally, this probability is defined as $\Pr[T \le \tau_1]$, and we lower bounded it in the previous subsection. In the numerator, the factor t is the maximum number of steps without dynamics that we consider. The additional neE_D term corresponds to a pessimistic estimate of the expected time to reach again the equilibrium point k^* after suffering from the first dynamic change.

Theorem 5. *For all $x_0 \in \{0,1\}^n$ and $t \in \mathbb{N}$,*

$$E_{x_0}[T] \le en(\ln(n) + 1) + \frac{t + neE_D}{(1 - p_D)^t \cdot \left(1 - \left(1 - \frac{1}{en}\right)^t\right)^{\lceil neE_D \rceil}}.$$

2.2.5 Sufficient Condition for a Polynomial Expected Hitting Time. We now give sufficient conditions on p_D and E_D for the runtime to be polynomial. In both the one bit-flip model [8] and the bitwise model [9], those two parameters have the same order of magnitude. In this case, taking t proportional to n gives the best runtime estimate. This corresponds to the first case of Theorem 6.

One can as well imagine a model where $p_D = o(E_D)$. Notice that the fact that p_D is always smaller than E_D allows us to suppose without losing generality that $p_D = o\left(\frac{\ln(n)}{n}\right)$. This corresponds to the second case of the following theorem.

Theorem 6. *Suppose that one of the following conditions holds.*

(i) $p_D = O\left(\frac{\ln(n)}{n}\right)$, *and* $E_D = O\left(\frac{\ln(n)}{n}\right)$.

(ii) $p_D = o\left(\frac{\ln(n)}{n}\right)$, *and* $E_D = O\left(\frac{\ln(n)}{n} n^{\frac{1}{nep_D}}\right)$.

Then the expected hitting time is polynomial.

2.2.6 Tight Bounds on the Expected Hitting Time. The most natural situation, as also in Droste [8, 9], is that p_D and E_D have the same order of magnitude. For this case, we now derive more explicit upper bounds on the expected hitting time.

Theorem 7. *Let* $\beta = 2\ln(2) \approx 1.39$, $\gamma = e\beta \approx 3.77$, *and* $\delta = e + \gamma \approx 6.49$. *Then the following assertions hold.*

(i) If $p_D \leq \left(\frac{1}{\gamma} + O\left(\frac{\ln(n)}{n}\right)\right)\frac{\ln(\ln(n))}{n}$ *and* $E_D \leq \frac{1}{\gamma}\frac{\ln(\ln(n))}{n}$, *then*

$$E[T] \leq (\delta + o(1))n\ln(n).$$

(ii) If $p_D \leq \left(c + O\left(\frac{\ln(n)}{n}\right)\right)\frac{\ln(n)}{n}$ *and* $E_D \leq c\frac{\ln(n)}{n}$, *then*

$$E[T] \leq (\gamma + o(1))n^{1+\beta ec}.$$

The following Corollary 8 analyzes the specific case of 1-bit dynamics. It is an immediate consequence of Theorem 7, since in that case we have $E_D = p_D = p$.

Corollary 8. *Let* $\beta = 2\ln(2) \approx 1.39$ *and* $\gamma = e\beta \approx 3.77$. *In the specific model of 1-bit dynamics with probability* p, *described in [8], if* $c > 0$ *is a constant such that* $p \leq c\frac{\ln(n)}{n}$, *then*

$$E[T] \leq (\gamma + o(1))n^{1+\beta ec}.$$

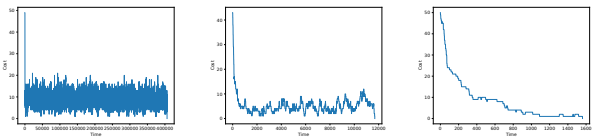
In [8] the bound $E[T] = O(n^{1+ec+o(1)}\ln(n))$ was obtained, which is better than ours. Contrary to this proof, our method does not use the specificity of the model. Consequently, we immediately obtain a similar bound for the bitwise dynamics.

Corollary 9. *Let* $\beta = 2\ln(2) \approx 1.39$ *and* $\gamma = e\beta \approx 3.77$. *In the specific model of bitwise dynamics with probability* p' *of flipping each bit, described in [9], if* $c > 0$ *is a constant such that* $p' \leq c\frac{\ln(n)}{n^2}$, *then*

$$E[T] \leq (\gamma + o(1))n^{1+\beta ec}.$$

This bound is significantly stronger than the two bounds shown in the past by Droste in [9] and Kötzing et al. in [13].

2.2.7 Experiments. In the theoretical analysis, we argued based on the assumption that there is an equilibrium point k^* which is easy to reach but difficult to leave towards the optimum. The fact that with this view on the process we could show very good runtime bounds indicates that our model of the optimization process is not too far from the truth. Since it is interesting also beyond our runtime guarantees to understand how evolutionary algorithms cope with dynamic changes or noise, we now conduct some very preliminary experiments.



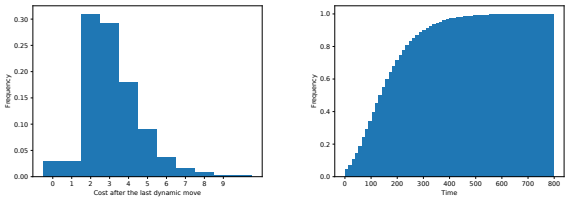
(a) $p' = \ln(n)/n$ **(b)** $p' = 0.5\ln(n)/n$ **(c)** $p' = 1/n$

Figure 1: Evolution of the cost of the search point for $n = 100$

In Figure 1, we show exemplarily three runs of the (1 + 1) EA optimizing the dynamic OneMax function in the presence of 1-bit dynamic changes of the optimum appearing with rate p. Depicted is the cost of the search point over time for $n = 100$ and $p = \ln(n)/n$, $0.5\ln(n)/n$, and $1/n$. The theoretical k^* in these cases is roughly 13, 7 and 3 respectively. For the two larger dynamic rates, indeed we observe in the plots that the cost oscillates around a point close to this theoretical k^* for a longer period of time. For the small dynamic rate of $1/n$, again in line with our theoretical predictions, the time spent around the k^* is small (theory would predict $O(n)$) and not distinguishable from the generally slow progress close to the optimum.

Our second (pessimistic) assumption in the runtime analyses is that from the equilibrium point we only make progress in short phases without any dynamic changes. This is clearly too strong to be absolutely true. To understand the characteristics of the final approach of the optimum, we depict in Figure 2(a) from which cost level on in the final approach to the optimum the optimization process was not subject to a dynamic change. To be able to conduct a reasonable number of experiments (namely $m = 10000$), we used the parameters $n = 100$ and $p = 0.25\ln(n)/n$. In this setting, we have $k^* = 4$. The results shown in Figure 2(a) are not fully conclusive, but with the median of the experimental results being 3 and very few values below 2, we at least mildly confirm the hypothesis that there are exceptionally few dynamic changes in the last part of the optimization process. A more conclusive study would need to take a much larger computational effort to both have k^* larger (so that the length of the final approach covers more fitness levels) and n larger (so that the asymptotics are more dominant – note that in our experiment, we observe a number of runs that finish with a dynamic change, that is, the dynamic move of the optimum transforms a non-optimal search point into an optimal one).

In Figure 2(b), we show the length of the optimization process starting from the last dynamic change of the optimum. Note that the expected time for the undisturbed algorithm to gain the last fitness level is already $(1+o(1))en \approx 272$, so the fact that the average time from the moment of the last dynamic change (as depicted in Figure 2(a)) is even smaller indicated that this final approach to the optimum is indeed exceptionally fast.



(a) Cost after the last dynamic move

(b) Cumulative histogram of the time to reach the optimum after the last dynamic move

Figure 2: The final approach to the optimum for $n = 100$ and $p = 0.25\frac{\ln(n)}{n} \approx 0.012$. With these parameters, the negative drift caused by the dynamics is roughly three times stronger than the positive drift produced by the algorithm at the penultimate fitness level.

3 NOISY OPTIMIZATION

In the context of optimization, *noise* means that our access to the true problem instance is obscured by small random perturbations. Since evolutionary algorithms as black-box optimizers access the problem instance only via the fitness function, evolutionary computation is mostly concerned with noisy objective functions, which instead of revealing the true fitness of a search point only return a disturbed value.

3.1 Previous Work and Our Results

3.1.1 Previous Work. While there is ample experimental experience on optimizing noisy objective functions via evolutionary algorithms, see, e.g., the surveys [1, 12], the theoretical understanding of noisy evolutionary optimization is still limited. The first runtime analysis for a noisy fitness function is also due to Droste. In [10], he considers the optimization of the OneMax benchmark function under *1-bit prior noise*. In this noise model, with probability q the algorithm does not learn the true fitness of a search point $x \in \{0,1\}^n$, but instead receives the fitness of a randomly chosen Hamming neighbor of x. Droste shows that for $q = O(\ln n/n)$, the expected runtime of the (1 + 1) EA on the OneMax function is polynomial. Unlike in his works on dynamic optimization [8, 9], the exponent of this polynomial is not made explicit. By redoing his calculations with more precision, it seems to us that his result shows that if $q \leq c\ln(n)/n$ for some constant c, then the optimization time (number of fitness evaluations until the true optimum is evaluated) T satisfies $E[T] = O(n^{16ec/3+1}\ln(n))$. Once again note that Droste uses the constant c such that $q \leq c\log_2(n)/n$.

In [11], a general noise model is analyzed which, besides partial evaluation noise and two posterior noise models, includes the 1-bit prior noise model and its natural counter-part, the bitwise prior noise model. In the *bitwise prior noise model*, the noisy fitness of a search point x is the true fitness of a search point $\tilde{x}$ obtained by flipping each bit of x independently with probability q. The general, technical result of [11] yields that the expected runtime in the bitwise noise model is $\Theta(n\log n)$ if $q = O(1/n^2)$, it is polynomial if $q = O(\ln(n)/n^2)$, and it is superpolynomial if $q = \omega(\ln(n)/n^2)$.

In [15], an extension of the bitwise noise model is presented and analyzed for some parameter combinations. In the (p_{noise}, q)-model, there is noise with probability p_{noise}, and in this case the disturbance is a bitwise noise with probability q. If $q = 1/n$, the tight range of p_{noise} allowing a polynomial runtime is $O(\ln(n)/n)$. The optimization of the LeadingOnes function by the (1 + 1) EA is also studied in this setting. It has been proven in [15] that the runtime is polynomial if $q = 1/n$ and $p_{noise} = O(\ln n/n^2)$.

All these works are mainly focused on identifying regions where for some parameters the expected runtime is polynomial without giving any explicit bounds over the runtime, although for practical applications we might be interested in having a more precise description of the polynomial bound.

3.1.2 Our Results. In this part we show that the same approach as in the dynamic setting can be adapted to analyze evolutionary optimization in a noisy environment. The plan is similar: the drift brings the search point to an equilibrium point from where the probability of a quick success without disturbance is not too small.

The disturbance is modeled by a noisy cost function $\tilde{c}$. In each optimization step, the evaluations of the costs of the search point and the offspring are replaced by the values given by the noisy cost function (see Algorithm 2 for a formal description). The algorithm makes the wrong decision when the true cost comparison between the parent and the offspring is inverted by the noisy cost function. Therefore, the maximal probability that the order between two bitstrings is inverted by the noisy cost function - denoted p_{inv} - is a natural parameter for the noise. We provide a general upper bound of the expected optimization time of the (1 + 1) EA on the OneMax function, as a function of p_{inv}

$$E[T] \leq \frac{e}{1-p_{inv}}n(1+\ln(n)) + (\gamma + o(1))ne^{n\beta p_{inv} + O(np_{inv}^2 + p_{inv})},$$

where $\beta = (2e-1)\ln\left(2-e^{-1}\right) + (1-e)\ln\left(1-e^{-1}\right)$ and $\gamma = (2e-1)\left(\ln\left(\frac{2e-1}{e-1}\right)+1\right)$. As a corollary, we identify three main regimes for $E[T]$, depending on the range of p_{inv}

$$\begin{cases} O(n\ln(n)) & \text{if } p_{inv} \leq \frac{c\ln(\ln(n))}{n} \text{ with } c \leq 1/\beta; \\ O(n\ln^{c\beta}(n)) & \text{if } p_{inv} \leq \frac{c\ln(\ln(n))}{n} \text{ with } c > 1/\beta; \\ O(n^{1+\beta c}) & \text{if } p_{inv} \leq \frac{c\ln(n)}{n}. \end{cases}$$

The use of a general parameter as p_{inv} makes our approach very flexible. The previous upper bounds apply to a wide variety of settings. In particular, it allows us to refine the results obtained in [11] for both prior and posterior noise, by extending the range of the $O(n\ln n)$ regime and by identifying another intermediate regime. In addition, we extend the range of values for which the runtime is polynomial in the (p_{noise}, q)-model. In all cases, despite the generality of our method, we are able to obtain competitive explicit bounds over the expected optimization time.

As with the dynamic setting, our method can be extended to the analysis of the expected optimization time of the LeadingOnes cost function, depending on the same parameter p_{inv}. We show that if the constant c is such that $p_{inv} \leq c\ln n/n^2$ for all n, then the expected optimization time of the (1 + 1) EA satisfies

$$E[T] = (\delta + o(1))n^{2+c(e-1)(1-e^{-1})/2}.$$

where $\delta = (e-1)/2$. This result has several applications. For instance, it allows to extend the range of parameters in the (p_{noise}, q)-model for which the expected runtime is polynomial.

Finally, our method provides new insight into the robustness against noise of evolutionary optimization with populations. As one particular result, we prove that the $(1+\lambda)$ EA can optimize the OneMax benchmark function efficiently despite a constant rate of 1-bit flip noise. More precisely, if the probability of 1-bit flip is smaller than a fixed constant (≈ 0.21), then $\lambda = \Theta(\ln(np))$ essentially suffices to optimize OneMax with the typical expected runtime of $O(n\ln(n))$.

3.1.3 Noise Model. The optimization of the OneMax function has been studied under different noise models. Explicit noise models such as the 1-bit noise and a generalization of the bitwise noise have been analyzed in [10] and [15] respectively. A more general approach applying to a variety of models is presented in [11].

Here we consider the search space $\{0,1\}^n$ of bitstrings of length n. For each $x \in \{0,1\}^n$ we denote by $c(x) = H(x,(1,\dots,1))$ the cost of x, thus making the problem a minimization problem. The disturbance is modeled by a noisy cost function. In comparison with the standard $(1+1)$ EA, its noisy version differs by the fact that at each step, both the cost of the search point and the cost of the offspring are replaced by the noisy cost, where for all $x \in \{0,1\}^n$ the noisy cost is a random variable denoted $\tilde{c}(x)$. We simply assume that for all $x \in \{0,1\}^n$, $\tilde{c}(x)$ depends only on x. Note that the explicit models studied in [10, 15] are included in this model. This randomized process can be represented by Algorithm 2.

Algorithm 2 $(1+1)$ EA minimizing a cost function c in a noisy setting

```
Choose x^0 ∈ {0,1}^n uniformly at random
for i = 0 to ∞ do
    Sample x'_i from x_i by flipping each bit with probability 1/n
    if c̃(x'_i) ≤ c̃(x_i) then
        x_{i+1} ← x'_i
    else x_{i+1} ← x_i
```

Formally the parameter p_{inv} sketched above can be defined as:

$$p_{inv} = \max_{x,y \in \{0,1\}^n} \Pr[\tilde{c}(x) \le \tilde{c}(y) \mid c(x) > c(y)].$$

We also assume there exists a constant $\varepsilon > 0$ such that $p_{inv} \le 1-\varepsilon$. The stochastic aspect of the algorithm makes $(x_i)_{i\in\mathbb{N}}$ and $(x'_i)_{i\in\mathbb{N}}$ two Markov chains, where x'_i is the offspring of x_i obtained by flipping independently each bit of x_i with probability $1/n$. The aim of the following sections is to give an upper bound on the expectancy of the hitting time of the process, defined as $T = \inf\{i \ge 0 \mid x_i = (1,\dots,1)\}$.

3.2 Optimization of Noisy OneMax Functions

3.2.1 Easy Process up to the Equilibrium Point. As in the study of the $(1+1)$ EA in a dynamic setting, we use multiplicative drift analysis to show that the optimization process quickly reaches the equilibrium point $k^* = \left\lceil \frac{enp_{inv}}{1-p_{inv}} \right\rceil$, which is intuitively the threshold cost value from which the drift becomes negative.

We identify two components of the drift, with opposite effects on the optimization process. As in the dynamic case, we pessimistically only consider steps of progress in which exactly a 0 is flipped into a 1. Besides, in the noisy setting, progress is made only when the algorithm is able to recognize a good offspring of the search point. Therefore, when the cost of the current search point equals k, $(1-p_{inv})\frac{k}{en}$ is a lower bound on the positive component of the drift. In the case where an inversion of cost values happens, and an offspring is selected wrongly, the typical move backwards is the expectancy of the number of bits flipped between the search point and its offspring. Since the mutation rate is $\frac{1}{n}$, it gives an estimation of the second drift component of $-p_{inv}$. The formalization of this reasoning with multiplicative drift analysis yields the following.

Theorem 10. *Let* $k^* = \left\lceil \frac{enp_{inv}}{1-p_{inv}} \right\rceil$ *and* $\tau_0 = \inf\{i \in \mathbb{N} \mid c(x_i) \le k^*\}$. *For all* $x \in \{0,1\}^n$,

$$\mathbb{E}_x[\tau_0] \le \frac{e}{1-p_{inv}} n(1+\ln(n)).$$

The previous theorem gives an upper bound on the time needed to reach a search point with a cost lower than k^*.

3.2.2 Exceptionally Fast Progress from the Equilibrium Point to the Optimum in Periods Without Effects of the Noise. Similarly to what we have done for the dynamic setting, we estimate the probability that the algorithm starting at k^* finds the optimum in a relatively short period of time without effects of the noise.

For all τ, let us denote by $\mathbf{N}_\tau$ the event that the noisy comparison between x_τ and x'_τ is identical to the true comparison, or that the true cost of x_τ and x'_τ are equal. The noisy process conditioned by the event $\bigcap_{1\le\tau\le t} \mathbf{N}_\tau$ is equivalent to the process of the $(1+1)$ EA optimizing a static OneMax function. Indeed, either the noisy and the true comparison give the same result, and the algorithm is able to recognize the best bitstring, or $c(x_\tau)$ and $c(x'_\tau)$ are equal, and we can indifferently select the search point or the offspring.

As a result, we can reuse our study of the distribution of the static hitting time T_S (see Theorem 2). Besides, the definition of p_{inv} results in the following lower bound

$$\Pr_{x_0}\left[\bigcap_{1\le\tau\le t} \mathbf{N}_\tau\right] \ge \left(p_{inv}\left(1-\tfrac{1}{n}\right)^n + (1-p_{inv})\right)^t.$$

Combining the two bounds, we obtain

$$\Pr_{x_0}[T \le t] \ge (1-p_{inv}(1-(1-\tfrac{1}{n})^n))^t (1-(1-\tfrac{1}{en})^t)^{k^*}$$

3.2.3 Upper Bound on the Expected Hitting Time. In Theorem 11, we combine the different arguments to obtain an upper bound on the expected hitting time, following an approach that is similar to Theorem 5. The first term $\frac{en(1+\ln(n))}{1-p_{inv}}$ corresponds to the expected time to obtain a search point x with a cost $c(x) \le k^*$. The second term corresponds to the progress from the equilibrium point k^*. We consider small successive periods of time t without any effects of the noise. This yields the following result, where the term $\frac{ne}{1-p_{inv}}$ acts as a penalty, which corresponds to the expected time to reach again the cost threshold k^* after a backward move of the search point due to noise.

Theorem 11. *For all $x_0 \in \{0,1\}^n$ and $t \in \mathbb{N}$,*

$$E_{x_0}[T] \leq \frac{en(\ln(n)+1)}{1-p_{inv}} + \frac{t+ne/(1-p_{inv})}{\left(1-p_{inv}\left(1-\left(1-\frac{1}{n}\right)^n\right)\right)^t \left(1-\left(1-\frac{1}{en}\right)^t\right)^{k^*}}.$$

Furthermore assume that there exists $\lambda > 0$ such that $t = \lfloor \lambda n \rfloor$ and that $\lim_{n\to\infty} p_{inv} = 0$. Then for $\lambda = e\ln\left(\frac{2-e^{-1}}{1-e^{-1}}\right)$ with $\beta = (2e-1)\ln\left(2-e^{-1}\right) + (1-e)\ln\left(1-e^{-1}\right) \approx 2.96$ and $\gamma = (2e-1)\ln\left(\frac{2e-1}{1-e^{-1}}\right) \approx 8.64$ the following inequality holds

$$E[T] \leq \frac{e}{1-p_{inv}} n(1+\ln(n)) + (\gamma + o(1))ne^{\beta p_{inv} + O(np_{inv}^2 + p_{inv})}.$$

The previous theorem has the following consequences.

Theorem 12. *The $(1+1)$ EA optimizes OneMax in an expected number of iterations of*

$$\begin{cases} O(n\ln(n)) & \text{if } p_{inv} \leq \frac{c\ln\ln n}{n} \text{ with } c \leq 1/\beta, \\ O(n\ln^{\beta c}(n)) & \text{if } p_{inv} > \frac{c\ln\ln n}{n} \text{ with } c \geq 1/\beta, \\ O(n^{1+\beta c}) & \text{if } p_{inv} \leq \frac{c\ln n}{n}. \end{cases}$$

Therefore analyzing the runtime of the $(1+1)$ EA on OneMax problems can be reduced to the analysis of p_{inv}. In the following subsections we analyze some explicit noise models with the same strategy. First an upper bound on p_{inv} is obtained in the context of the given model, then we replace this expression in Theorem 11.

3.2.4 Prior Noise Model. Prior noise refers to a model where the noise comes from not evaluating the true search point. Hence for $x \in \{0,1\}^n$, $\tilde{c}(x)$ can be rewritten as $c(\tilde{x})$, where $\tilde{x}$ is a disturbed version of x. In order to stay within the framework of our model, it is sufficient that the distribution of $\tilde{x}$ depends only on x.

First we consider the (p_{noise}, q)-model. In this model x is disturbed with probability p_{noise}, and in this case, $\tilde{x}$ is obtained by flipping each bit of x independently with probability q. In this setting we can show that $p_{noise}(1-(1-q)^{n+1})$ is an upper bound on p_{inv}. Then Theorem 11 yields the following corollary.

Corollary 13. *Let $\beta \approx 2.96$ as in Theorem 11. Let T be the runtime of the $(1+1)$ EA on the noisy OneMax function in the (p_{noise}, q) model.*

(i) Let $q = o(1/n)$. If $p_{noise}q \leq c\ln(n)/n^2$ for some constant c, then $E[T] = O(n^{1+\beta c})$. If $p_{noise}q \leq c\ln\ln(n)/n^2$ for some constant c, then

$$E[T] = \begin{cases} O(n\ln(n)) & \text{if } c \leq 1/\beta, \\ O(n\ln^{\beta c}(n)) & \text{otherwise.} \end{cases}$$

(ii) Let $q \leq c_q/n$ for some constant $c_q > 0$. If $p_{noise} \leq c_p\ln(n)/n$ for some constant $c_p > 0$, then $E[T] = O(n^{1+\beta c_p(1-e^{-c_q})})$. If $p_{noise} \leq c_p\ln\ln(n)/n$ for some constant $c_p > 0$, then

$$E[T] = \begin{cases} O(n\ln(n)) & \text{if } c_p(1-e^{-c_q}) \leq 1/\beta, \\ O(n\ln^{\beta c}(n)) & \text{otherwise.} \end{cases}$$

In the 1-bit noise model exactly one uniformly chosen bit changes with probability q, while no bit is changed with probability $1-q$. By considering the different possible cases of inversion in this model, we show that $p_{inv} \leq q + O(\frac{q}{n} + q^2)$. Then Theorem 11 gives the following corollary.

Corollary 14. *Let $\beta \approx 2.96$ as in Theorem 11. Let T be the runtime of the $(1+1)$ EA on the noisy OneMax function in the 1-bit noise model.*

(i) If $q \leq c\ln(n)/n$ for some constant c, then $E[T] = O(n^{1+\beta c})$.
(ii) If $q \leq c\ln(\ln(n))/n$ for some constant c, then

$$E[T] = \begin{cases} O(n\ln(n)) & \text{if } c \leq 1/\beta; \\ O(n\ln^{\beta c}(n)) & \text{otherwise.} \end{cases}$$

3.2.5 Additive Posterior Noise. In the posterior noise model the value of the true fitness of the search point is disturbed while the noise is directly applied to the search point before evaluation in the prior noise model. Here we only consider additive posterior noise as described in [11]. Let D be some distribution with variance σ^2. Then for all $x \in \{0,1\}^n$, $\tilde{c}(x)$ is a random variable which follows the same law as $c(x) + D$. By reading carefully the proof of Corollary 10 in [11], we obtain that $p_{inv} \leq 2\sigma^2$. Hence the following corollary.

Corollary 15. *Let $\beta \approx 2.96$ as in Theorem 11. Let T be the runtime of the $(1+1)$ EA on the noisy OneMax function in this additive posterior noise model.*

(i) If $\sigma^2 \leq c\ln(n)/n$ for some constant c, then $E[T] = O(n^{1+2\beta c})$.
(ii) If $\sigma^2 \leq c\ln(\ln n)/n$ for some constant c, then

$$E[T] = \begin{cases} O(n\ln(n)) & \text{if } c \leq 1/2\beta; \\ O(n\ln^{2\beta c}(n)) & \text{otherwise.} \end{cases}$$

REFERENCES

[1] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing* 8, 2 (2009), 239–287.
[2] Jürgen Branke. 2012. *Evolutionary Optimization in Dynamic Environments*. Vol. 3. Springer Science & Business Media.
[3] Tianshi Chen, Yunji Chen, Ke Tang, Guoliang Chen, and Xin Yao. 2011. The impact of mutation rate on the computation time of evolutionary dynamic optimization. *CoRR* abs/1106.0566 (2011).
[4] Duc-Cuong Dang, Thomas Jansen, and Per Kristian Lehre. 2017. Populations can be essential in tracking dynamic optima. *Algorithmica* 78, 2 (2017), 660–680.
[5] Duc-Cuong Dang and Per Kristian Lehre. 2015. Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms. In *Proc. of FOGA'15*. 62–68.
[6] Benjamin Doerr. 2018. Better runtime guarantees via stochastic domination. In *Proc. of EvoCOP'18*. 1–17.
[7] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative drift analysis. *Algorithmica* 64, 4 (2012), 673–697.
[8] Stefan Droste. 2002. Analysis of the (1+ 1) EA for a dynamically changing OneMax-variant. In *Proc. of CEC'02*, Vol. 1. 55–60.
[9] Stefan Droste. 2003. Analysis of the (1+1) EA for a dynamically bitwise changing OneMax. In *Proc. of GECCO'03*. 909–921.
[10] Stefan Droste. 2004. Analysis of the (1+1) EA for a noisy OneMax. In *Proc. of GECCO'04*. 1088–1099.
[11] Christian Gießen and Timo Kötzing. 2016. Robustness of populations in stochastic environments. *Algorithmica* 75, 3 (2016), 462–489.
[12] Yaochu Jin and Jürgen Branke. 2005. Evolutionary optimization in uncertain environments—a survey. *IEEE Trans. Evolutionary Computation* 9, 3 (2005), 303–317.
[13] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. 2015. (1+1) EA on generalized dynamic OneMax. In *Proc. of FOGA'15*. 40–51.
[14] Trung Thanh Nguyen, Shengxiang Yang, and Jürgen Branke. 2012. Evolutionary dynamic optimization: a survey of the state of the art. *Swarm and Evolutionary Computation* 6 (2012), 1–24.
[15] Chao Qian, Chao Bian, Wu Jiang, and Ke Tang. 2017. Running time analysis of the (1+1)-EA for OneMax and LeadingOnes under bit-wise noise. *CoRR* abs/1711.00956 (2017).
[16] Feng Shi, Martin Schirneck, Tobias Friedrich, Timo Kötzing, and Frank Neumann. 2017. Reoptimization times of evolutionary algorithms on linear functions under dynamic uniform constraints. In *Proc. of GECCO'17*. 1407–1414.
[17] Stephen A. Stanhope and Jason M. Daida. 1999. (1+ 1) genetic algorithm fitness dynamics in a changing environment. In *Proc. of CEC'99*, Vol. 3. IEEE, 1851–1858.