The University of
Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Dang, Duc-Cuong and Lehre, Per Kristian (2016) Runtime analysis of non-elitist populations: from classical optimisation to partial information. Algorithmica, 75 (3). pp. 428-461. ISSN 1432-0541

Runtime analysis of non-elitist populations: from classical optimisation to partial information *

Duc-Cuong Dang and Per Kristian Lehre

School of Computer Science, University of Nottingham, UK †

Abstract

Although widely applied in optimisation, relatively little has been proven rigorously about the role and behaviour of populations in randomised search processes. This paper presents a new method to prove upper bounds on the expected optimisation time of population-based randomised search heuristics that use non-elitist selection mechanisms and unary variation operators. Our results follow from a detailed drift analysis of the population dynamics in these heuristics. This analysis shows that the optimisation time depends on the relationship between the strength of the selective pressure and the degree of variation introduced by the variation operator. Given limited variation, a surprisingly weak selective pressure suffices to optimise many functions in expected polynomial time. We derive upper bounds on the expected optimisation time of non-elitist Evolutionary Algorithms (EA) using various selection mechanisms, including fitness proportionate selection. We show that EAs using fitness proportionate selection can optimise standard benchmark functions in expected polynomial time given a sufficiently low mutation rate.

As a second contribution, we consider an optimisation scenario with *partial information*, where fitness values of solutions are only partially available. We prove that non-elitist EAs under a set of specific conditions can optimise benchmark functions in expected polynomial time, even when vanishingly little information about the fitness values of individual solutions or populations is available. To our knowledge, this is the first runtime analysis of randomised search heuristics under partial information.

1 Introduction

Randomised Search Heuristics (RSHs) such as Evolutionary Algorithms (EAs) or Genetic Algorithms (GAs) are general purpose search algorithms which require little knowledge of the problem domain for their implementation. Nevertheless, they are often successful in practice [3, 36]. Despite their often complex behaviour, there have been significant advances in the theoretical understanding

---

of these algorithms over the recent years [1, 17, 31]. One contributing factor behind these advances may have been the clear strategy to initiate the analysis on the simplest settings before proceeding to more complex scenarios, while at the same time developing appropriate analytical techniques. Therefore, most analytical techniques of the current literature were first designed for the single-individual setting, i.e., the $(1 + 1)$ EA and the like (see [17] for a overview). Some techniques have emerged later for analysing EAs with populations. The family tree technique was introduced in [35] to analyse the $(\mu+1)$ EA. However, the analysis does not cover offspring populations. A performance comparison of $(\mu + \mu)$ EA for $\mu = 1$ and $\mu > 1$ was conducted in [16] using Markov chains to model the search processes. Based on a similar argument to fitness-level [34], upper bounds on the expected runtime of the $(\mu + \mu)$ EA were derived in [2]. The fitness-level argument also assisted the analysis of parallel EAs in [22]. However, the analysis is restricted to EAs with truncation selection and does not apply to general selection schemes. Drift analysis [13] was used in [30] and in [24] to show the inefficiency of standard fitness proportionate selection without scaling. The approach involves finding a function that maps the state of an entire population to a real number measuring the distance between the current population and the set of optimal solutions. The required distance function is highly complex even for a simple function like ONEMAX.

In this paper, we are interested in estimating upper bounds on the expected runtime of a large class of algorithms that employs *non-elitist* populations. More precisely, the technique developed in this paper is applied to algorithms covered by the scheme of Algorithm 1. The general term Population Selection-Variation Algorithm is used to emphasise that it does not only cover EAs, but also other population-based RSHs. In this scheme, each solution of the current population is generated by first sampling a solution from the previous population with a so-called sampling mechanism p_{sel}, then by perturbing the sampled solution with a so-called variation operator p_{mut}. The scheme defines a class of algorithms which can be instantiated by specifying the variation operator p_{mut}, and the selection mechanism p_{sel}. Here, we assume that the algorithm optimises a fitness function $f : \mathcal{X} \to \mathbb{R}$, implicitly given by the selection mechanism p_{sel}. Particularly, this description of p_{sel} allows the scheme to cover other optimisation scenarios than just *static/classical* optimisation. The variation operator p_{mut} is restricted to *unary* ones, i.e., those where each individual has only one parent. Further discussions on higher-arity variation operators can be found in [4].

Algorithm 1 Population Selection-Variation Algorithm

Require: Finite state space $\mathcal{X}$,
    and initial population $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$.
1: **for** $t = 0, 1, 2, \ldots$ until a termination condition is met **do**
2:    **for** $i = 1$ to λ **do**
3:       Sample $I_t(i) \in [\lambda]$ according to $p_{\text{sel}}(P_t)$.
4:       $x := P_t(I_t(i))$.
5:       Sample x' according to $p_{\text{mut}}(x)$.
6:       $P_{t+1}(i) := x'$.
7:    **end for**
8: **end for**

Algorithm 1 has been studied in a sequence of papers [23, 24, 26], while a

specific instance of the scheme known as the $(1, \lambda)$ EA was analysed in [32]. Its runtime depends critically on the balance between the selective pressure imposed by p_{sel}, and the amount of variation introduced by p_{mut} [26]. When the selective pressure falls below a certain *threshold* which depends on the mutation rate, the expected runtime of the algorithm becomes exponential [23]. Conversely, if the selective pressure exceeds this threshold significantly, it is possible to show using a *fitness-level* argument [34] that the expected runtime is bounded from above by a polynomial [24].

The so-called fitness-level technique is one of the simplest ways to derive upper bounds on the expected runtime of *elitist* EAs [34]. The idea is to partition the search space $\mathcal{X}$ into so-called *fitness levels* $A_1, \ldots, A_{m+1} \subseteq \mathcal{X}$, such that for all $1 \leq j \leq m$, all the search points in fitness level A_j have inferior function value to the search points in fitness level A_{j+1}, and all global optima are in the last level A_{m+1}. Due to the elitism, which means the offspring population has to compete with the best individuals of the parent population, the EA will never lose the highest fitness level found so far. If the probability of mutating any search point in fitness level A_j into one of the higher fitness levels is at least s_j, then the expected time until this occurs is at most $1/s_j$. The expected time to overcome all the inferior levels, i.e., the expected runtime, is by linearity of expectation no more than $\sum_{j=1}^{m} 1/s_j$. This simple technique can sometimes provide tight upper bounds of the expected runtime. Recently in [33], an extension of the method has been shown to be able to derive tight lower bounds, the key idea is to estimate the number of fitness levels being skipped on average.

Non-elitist algorithms, such as Algorithm 1, may lose the current best solution. Therefore, to guarantee the optimisation of f, a set of conditions has to be applied so that the population does not frequently fall down to lower fitness levels. Such conditions were introduced in [24], which imply a large enough population size and a strong enough *selective pressure* relative to the variation operator. In particular, the probability that Algorithm 1 in line 3 selects an individual x among the best γ-fraction of the population, and the variation operator in line 5 does not produce an inferior individual x', must be at least $(1 + \delta)\gamma$, for all $\gamma \in (0, \gamma_0]$, where δ and γ_0 are positive constants. When the conditions are satisfied, [24] concludes that the expected runtime is bounded from above by $O(m\lambda^2 + \sum_{j=1}^{m} 1/s_j)$. The proof divides the run of the algorithm into phases. A phase is considered successful if the population does not fall to a lower fitness level during the phase. The duration of each phase conditional on the phase being successful is analysed separately using drift analysis [15]. An unconditional upper bound on the expected runtime of the algorithm is obtained by taking into account the success probabilities of the phases.

In this paper, we present a new theorem which improves the results of [24]. The contributions of this paper are twofold: (i) a more precise and general upper bound for the fitness-level technique of [24] and (ii) an new application of the technique to the analysis of EAs under uncertainty or incomplete information. In (i), we improve the above bound to $O(m\lambda \ln \lambda + \sum_{j=1}^{m} 1/s_j)$ for the case of constant δ. Increasing the population size therefore has a smaller impact on the runtime than previously thought. This improvement is illustrated with upper bounds on the expected runtime of non-elitist EAs on many example functions and for various selection mechanisms. On the other hand, the new theorem makes the relationship between parameter δ and the runtime explicit. This observation allows us to prove that the standard fitness proportionate selection

can be made efficient without scaling, in contrast to the previous result [30, 24]. Particularly in (ii), using the improved technique we show that non-elitist EAs under a set of specific conditions are still able to optimise standard functions, such as OneMax and LeadingOnes, in expected polynomial time even when little information about the fitness values of individual solutions or populations is available during the search. To the best of our knowledge, this is the first time optimisation under incomplete information has been formalised for pseudo-Boolean functions and rigorously analysed for population-based algorithms. All these improvements are achieved due to a much more detailed analysis of the population dynamics, and the proof of the new theorem is constructed with drift analysis.

The remainder of the paper is organised as follows. Section 2 provides a set of preliminary results that are crucial for our improvement. Our new theorem is presented with its proof in Section 3. The new results for the set of functions which were previously presented in [24] are described in Section 4. Section 5 presents another application of the new theorem to runtime analysis of EAs under incomplete information. Finally, some conclusions are drawn. Some supplementary results can be found in the appendix.

2 Preliminaries

For any positive integer n, define $[n] := \{1, 2, \ldots, n\}$. The notation $[\mathcal{A}]$ is the Iverson bracket, which is 1 if the condition $\mathcal{A}$ is true and 0 otherwise. The natural logarithm is denoted by $\ln(\cdot)$, and the logarithm to the base 2 is denoted by $\log(\cdot)$. For a bitstring x of length n, define $|x|_1 := \sum_{i=1}^{n} x_i$. Without loss of generality, we assume throughout the paper the goal is to maximise some function $f \colon \mathcal{X} \to \mathbb{R}$, which we call the *fitness* function.

A random variable X is *stochastically dominated* by a random variable Y, denoted by $X \preceq Y$, if $\Pr(X > x) \leq \Pr(Y > x)$ for all $x \in \mathbb{R}$. Equivalently, $X \preceq Y$ holds if and only if $\mathbf{E}\left[f(X)\right] \leq \mathbf{E}\left[f(Y)\right]$ for any non-decreasing function $f \colon \mathbb{R} \to \mathbb{R}$. The *indicator function* $\mathbb{1}_{\mathcal{E}} \colon \Omega \to \mathbb{R}$ for an event $\mathcal{E} \subseteq \Omega$ is defined as

$$\mathbb{1}_{\mathcal{E}}(\omega) := \begin{cases} 1 & \text{if } \omega \in \mathcal{E}, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

For any event $\mathcal{E} \subseteq \Omega$ and time index $t \in \mathbb{N}$, we denote the probability of an event $\mathcal{E}$ conditional on the σ-algebra $\mathscr{F}_t$ by

$$\Pr{}_t(\mathcal{E}) := \mathbf{E}\left[\mathbb{1}_{\mathcal{E}} \mid \mathscr{F}_t\right]$$

We denote the expectation of a random variable X conditional on the σ-algebra $\mathscr{F}_t$ and the event $\mathcal{E}$ by

$$\mathbf{E}_t\left[X \mid \mathcal{E}\right] := \frac{\mathbf{E}\left[X \,;\, \mathcal{E} \mid \mathscr{F}_t\right]}{\Pr{}_t(\mathcal{E})}$$

where the semi-colon notation " ; " is defined as (see e.g. [20], page 49)

$$\mathbf{E}\left[X \,;\, \mathcal{E} \mid \mathscr{F}_t\right] := \mathbf{E}\left[X \cdot \mathbb{1}_{\mathcal{E}} \mid \mathscr{F}_t\right].$$

Algorithm 1 keeps a vector $P_t \in \mathcal{X}^\lambda, t \geq 0$, of λ search points. In analogy with evolutionary algorithms, the vector will be referred to as a *population*, and the vector elements as *individuals*. Each iteration of the inner loop is called a *selection-variation* step, and each iteration of the outer loop, which counts for λ iterations of the inner loop, is called a *generation*. The initial population is sampled uniformly at random. In subsequent generations, a new population P_{t+1} is generated by independently sampling λ individuals from the existing population P_t according to p_{sel}, and perturbing each of the sampled individuals by a variation operator p_{mut}.

The ordering of the elements in a population vector $P \in \mathcal{X}^\lambda$ according to non-increasing f-value will be denoted $x_{(1)}, x_{(2)}, \ldots, x_{(\lambda)}$, i.e., such that $f(x_{(1)}) \geq f(x_{(2)}) \geq \cdots \geq f(x_{(\lambda)})$. For any constant $\gamma \in (0,1)$, the individual $x_{(\lceil \gamma \lambda \rceil)}$ will be referred to as the γ-*ranked individual* of the population.

Similar to the analysis of randomised algorithms [10], the runtime of the algorithm when optimising f is defined to be the first point in time, counted in terms of number of solution evaluations, when a global optimum x^* of f, i.e., $\forall x \in \mathcal{X}, f(x^*) \geq f(x)$, appears in P_t. However, the number of solution evaluations in a run of Algorithm 1 is implicitly defined, i.e., it is equal to the number of selection-variation steps multiplied by the number of fitness evaluations in p_{sel}. Therefore, in our general result the runtime of the algorithm will be stated as the number of selection-variation steps, while in specific cases the latter is translated into number of fitness evaluations.

Variation operators are formally represented as transition matrices $p_{\mathrm{mut}} : \mathcal{X} \times \mathcal{X} \to [0,1]$ over the search space, where $p_{\mathrm{mut}}(x \mid y)$ represents the probability of perturbing an individual y into an individual x. Selection mechanisms are represented as probability distributions over the set of integers $[\lambda]$, where the conditional probability $p_{\mathrm{sel}}(i \mid P_t)$ represents the probability of selecting individual $P_t(i)$, i.e., the i-th individual from population P_t. From Algorithm 1, it follows that each individual within a generation t is sampled independently from the same distribution p_{sel}.

In contrast to *rank-based* selection mechanisms in which the decisions are made based on the ranking of the individuals with respect to the fitness function, some selection mechanisms rely directly on the fitness values. Thus the performance of an algorithm making use of such selection mechanisms may depend on how much the fitness values differ.

Definition 1. *A function $f : \mathcal{X} \to \mathbb{R}$ is called θ-distinctive for some $\theta > 0$ if for all $x, y \in \mathcal{X}$ and $f(x) \neq f(y)$ we have $|f(x) - f(y)| \geq \theta$.*

In this paper, we will work exclusively with fitness-based partitions of the search space, of which the formal definition is the following.

Definition 2 ([34])**.** *Given a function $f : \mathcal{X} \to \mathbb{R}$, a partition of $\mathcal{X}$ into $m + 1$ levels $A_1, \ldots, A_{m+1}$ is called f-based if all of the following conditions hold: (i) $f(x) < f(y)$ for all $x \in A_j$, $y \in A_{j+1}$ and $j \in [m+1]$; (ii) $f(y) = \max_{x \in \mathcal{X}}\{f(x)\}$ for all $y \in A_{m+1}$.*

The *selective pressure* of a selection mechanism refers to the degree to which the selection mechanism selects individuals that have higher f-values. To quantify the selective pressure in a selective mechanism p_{sel}, we define its *cumulative selection probability* with respect to a fitness function f as follows.

Definition 3 ([24]). *The* cumulative selection probability β *of a selection mechanism* p_{sel} *with respect to a fitness function* $f : \mathcal{X} \to \mathbb{R}$ *is defined for all* $\gamma \in (0, 1]$ *and* $P \in \mathcal{X}^\lambda$ *by*

$$\beta(\gamma, P) := \sum_{i=1}^{\lambda} p_{\text{sel}}(i \mid P) \cdot \left[f(P(i)) \geq f(x_{(\lceil \gamma \lambda \rceil)}) \right]$$

Informally, $\beta(\gamma, P)$ is the probability of selecting an individual with fitness at least as high as that of the γ-ranked individual. We write $\beta(\gamma)$ instead of $\beta(\gamma, P)$ when β can be bounded independently of the population vector P. Our main tool to prove the theorem in Section 3 is the following additive drift theorem.

Theorem 4 (Additive Drift Theorem [15]). *Let* $(X_t)_{t \geq 0}$ *be a stochastic process over state space* S, $d : S \to \mathbb{R}$ *be a distance function on* S *and* $\mathscr{F}_t$ *be the filtration induced by* $X_0, \ldots, X_t$. *Define* $T := \min\{t \mid d(X_t) \leq 0\}$. *If there exist* $B, \Delta > 0$ *such that for all* $t \geq 0$

1. $\Pr(d(X_t) < B) = 1$, *and*

2. $\mathbf{E}\left[d(X_t) - d(X_{t+1}) - \Delta \ ; \ d(X_t) > 0 \mid \mathscr{F}_t \right] \geq 0$,

then $\mathbf{E}[T] \leq B/\Delta$.

The drift theorem in evolutionary computation is typically applied to bound the expected time until the process reaches a potential of 0 (ie., $d(X_t) = 0$). The so-called *potential* or *distance* function d serves the purpose of measuring the progress toward the optimum. In our case, $X_t = X_t^\ell$ represents the number of individuals in the population that have advanced to a higher fitness level than the current level ℓ. We are interested in the time until $X_t \geq \gamma_0 \lambda$, ie., until some constant fraction of the population has advanced to the higher fitness level, sometimes called the take-over time. The earlier fitness-level theorem for populations [24] used a linear potential function on the form $d(x) = C - x$, and obtained an expected progress of δX_t for some constant δ. This situation is somehow inverse to that in *multiplicative drift* [7], as we are waiting for the process X_t to reach a large value. It is well known that d should be chosen such that Δ is a constant independent of X_t. In this paper, we therefore consider the potential function $d(x) = C - \ln(1 + cx)$, for appropriate choices of C and c. However, in order to bound the drift with this potential function, it is insufficient to only consider the expectation of X_{t+1}. The following lemma shows that it is sufficient to exploit the fact that X_{t+1} is binomially distributed.

Lemma 5. *If* $X \sim \text{Bin}(\lambda, p)$ *with* $p \geq (i/\lambda)(1 + \delta)$ *and* $i \geq 1$ *for some* $\delta > 0$, *then*

$$\mathbf{E}\left[\ln\left(\frac{1 + cX}{1 + ci} \right) \right] \geq c\varepsilon,$$

where $\varepsilon = \min\{1/2, \delta/2\}$ *and* $c = \varepsilon^4/24$.

Proof. Let $Y \sim \text{Bin}(\lambda, (i/\lambda)(1 + 2\varepsilon))$, then $Y \preceq X$. Therefore,

$$\mathbf{E}\left[\ln\left(\frac{1 + cX}{1 + ci} \right) \right] \geq \mathbf{E}\left[\ln\left(\frac{1 + cY}{1 + ci} \right) \right],$$

6

and it is sufficient to show that $\mathbf{E}\left[\ln\left(\frac{1+cY}{1+ci}\right)\right] \geq c\varepsilon$ to complete the proof. We consider the two following cases.

For $i \geq 8/\varepsilon^3$: Let $q := \Pr\left(Y \leq (1+\varepsilon)i\right)$, then by the law of total probability

$$\mathbf{E}\left[\ln\left(\frac{1+cY}{1+ci}\right)\right] \geq (1-q)\ln\left(\frac{1+c(1+\varepsilon)i}{1+ci}\right) - q\ln(1+ci)$$

$$= \ln\left(1 + \frac{c\varepsilon i}{1+ci}\right) - q\ln\left(1 + c(1+\varepsilon)i\right).$$

Note that $ci = (\varepsilon^4/24)i \geq (\varepsilon^4/24)(8/\varepsilon^3) = \varepsilon/3$, thus

$$\frac{ci\varepsilon}{ci+1} = \frac{\varepsilon}{1+1/ci} \geq \frac{\varepsilon}{1+3/\varepsilon} = \frac{\varepsilon^2}{\varepsilon+3} \geq \frac{\varepsilon^2}{1/2+3} = \frac{2\varepsilon^2}{7}. \tag{1}$$

We have the following based on Lemma 33 and (1)

$$\ln\left(1 + \frac{c\varepsilon i}{1+ci}\right) \geq \ln(1 + 2\varepsilon^2/7) \geq (2\varepsilon^2/7)(1 - \varepsilon^2/7)$$

$$\geq (2\varepsilon^2/7)(1 - 1/28) = 27\varepsilon^2/98.$$

Using Lemma 27 with $\mathbf{E}[Y] = (1+2\varepsilon)i$, and $e^x > x$, $\varepsilon \leq 1/2$, we get

$$q = \Pr\left(Y \leq \left(1 - \frac{\varepsilon}{1+2\varepsilon}\right)\mathbf{E}[Y]\right)$$

$$\leq \exp\left(-\frac{i\varepsilon^2}{2(1+2\varepsilon)}\right) \leq \exp\left(-\frac{i\varepsilon^2}{4}\right) < \frac{4}{i\varepsilon^2}.$$

We then have

$$q\ln(1 + c(1+\varepsilon)i) \leq qc(1+\varepsilon)i \leq \left(\frac{4}{i\varepsilon^2}\right)\left(\frac{\varepsilon^4(1+1/2)i}{24}\right) = \frac{\varepsilon^2}{4}.$$

Putting everything together, we get

$$\mathbf{E}\left[\ln\left(\frac{1+cY}{1+ci}\right)\right] \geq \varepsilon^2\left(\frac{27}{98} - \frac{1}{4}\right) = \frac{5\varepsilon^2}{196}$$

$$> (1/8)(\varepsilon/24)\varepsilon = (1/8)(c/\varepsilon^3)\varepsilon \geq c\varepsilon.$$

For $1 \leq i \leq 8/\varepsilon^3$: Note that $\varepsilon \leq 1/2$ implies $\mathbf{E}[Y] = (1+2\varepsilon)i \leq 2i$, and for binomially distributed random variables we have $\mathbf{Var}[Y] \leq \mathbf{E}[Y]$, so

$$\mathbf{E}\left[Y^2\right] = \mathbf{Var}[Y] + \mathbf{E}[Y]^2 \leq 2i(1+2i) \leq 2i(i+2i) = 6i^2. \tag{2}$$

Using Lemma 33, (2), $ci \leq \varepsilon/3$, and $\varepsilon \leq 1/2$, we get

$$\mathbf{E}\left[\ln\left(\frac{1+cY}{1+ci}\right)\right] = \mathbf{E}[\ln(1+cY)] - \ln(1+ci)$$

$$\geq \mathbf{E}[cY] - \frac{\mathbf{E}\left[c^2Y^2\right]}{2} - ci$$

$$\geq ci(1+2\varepsilon) - 3c^2i^2 - ci$$

$$\geq ci(2\varepsilon - 3(\varepsilon/3)) = ci\varepsilon \geq c\varepsilon.$$

7

Combining the two cases, we have, for all $i \geq 1$,

$$\mathbf{E}\left[\ln\left(\frac{1+cX}{1+ci}\right)\right] \geq \mathbf{E}\left[\ln\left(\frac{1+cY}{1+ci}\right)\right] \geq c\varepsilon. \qquad \square$$

We will also need the following three results.

Lemma 6 (Lemma 18 in [24]). *If $X \sim \mathrm{Bin}(\lambda, p)$ with $p \geq (i/\lambda)(1+\delta)$, then $\mathbf{E}\left[e^{-\kappa X}\right] \leq e^{-\kappa i}$ for any $\kappa \in (0, \delta)$.*

Proof. For the completeness of our main theorem, we detail the proof of [24] as follows. The value of the moment generating function $M_X(t)$ of the binomially distributed variable X at $t = -\kappa$ is

$$\mathbf{E}\left[e^{-\kappa X}\right] = M_X(-\kappa) = (1 - p(1 - e^{-\kappa}))^{\lambda}.$$

It follows from Lemma 31 and from $1 + \kappa < 1 + \delta$ that

$$p(1 - e^{-\kappa}) \geq \frac{i(1+\delta)}{\lambda}\left(\frac{\kappa}{1+\kappa}\right) \geq \frac{\kappa i}{\lambda}.$$

Altogether, $\mathbf{E}\left[e^{-\kappa X}\right] \leq (1 - \kappa i/\lambda)^{\lambda} \leq e^{-\kappa i}$. $\qquad \square$

We also use *negative drift* to prove the inefficiency of the $(1+1)$ EA under the partial information setting in Section 5. The following theorem, which is a corollary result of Theorem 2.3 in [13], was presented in [25].

Theorem 7 (Hajek's theorem [25]). *Let $(X_t)_{t \geq 0}$ be a stochastic process over some bounded state $S \in [0, \infty)$, $\mathscr{F}_t$ be the filtration generated by $X_0, ..., X_t$. Given $a(n)$ and $b(n)$ depending on a parameter n such that $b(n) - a(n) = \Omega(n)$, define $T := \min\{t \mid X_t \geq b(n)\}$. If there exist positive constants λ, ε, D such that*

(L1) $\mathbf{E}\left[X_{t+1} - X_t + \varepsilon \; ; \; X_t > a(n) \mid \mathscr{F}_t\right] \leq 0$,
(L2) $(|X_{t+1} - X_t| \mid \mathscr{F}_t) \preceq Y$ *with* $\mathbf{E}\left[e^{\lambda Y}\right] \leq D$,

then there exists a positive constant c such that

$$\Pr\left(T \leq e^{cn} \mid X_0 < a(n)\right) \leq e^{-\Omega(n)}.$$

Informally, if the progress (toward state $b(n)$) becomes negative from state $a(n)$ (L1) and big jumps are rare (L2), then $\mathbf{E}[T]$ is exponential, e.g., by Markov's inequality

$$\mathbf{E}\left[T \mid X_0 < a(n)\right] \geq \Pr\left(T > e^{cn} \mid X_0 < a(n)\right) e^{cn}$$
$$\geq \left(1 - e^{-\Omega(n)}\right) e^{cn} = e^{\Omega(n)}.$$

3 A Refined Fitness Level Theorem

For notational convenience, define for $j \in [m]$ the set $A_j^+ := \bigcup_{i=j+1}^{m+1} A_i$, i.e., the set of search points at higher fitness levels than A_j. We have the following theorem and corollaries for the runtime of non-elitist populations.

Theorem 8. *Given a function $f : \mathcal{X} \to \mathbb{R}$, and an f-based partition $(A_1, \ldots, A_{m+1})$, let T be the number of selection-variation steps until Algorithm 1 with a selection mechanism p_{sel} obtains an element in A_{m+1} for the first time. If there exist parameters $p_0, s_1, \ldots, s_m, s_* \in (0, 1]$, and $\gamma_0 \in (0, 1)$ and $\delta > 0$, such that*

(C1) $p_{\mathrm{mut}} \left(y \in A_j^+ \mid x \in A_j \right) \geq s_j \geq s_*$ *for all $j \in [m]$,*

(C2) $p_{\mathrm{mut}} \left(y \in A_j \cup A_j^+ \mid x \in A_j \right) \geq p_0$ *for all $j \in [m]$,*

(C3) $\beta(\gamma, P) p_0 \geq (1 + \delta)\gamma$ *for all $P \in \mathcal{X}^\lambda$ and $\gamma \in (0, \gamma_0]$,*

(C4) $\lambda \geq \dfrac{2}{a} \ln \left(\dfrac{16m}{ac\varepsilon s_*} \right)$ *with $a = \dfrac{\delta^2 \gamma_0}{2(1 + \delta)}$, $\varepsilon = \min\{\delta/2, 1/2\}$ and $c = \varepsilon^4/24$,*

then

$$\mathbf{E}\left[T\right] \leq \frac{2}{c\varepsilon} \left(m\lambda(1 + \ln(1 + c\lambda)) + \frac{p_0}{(1 + \delta)\gamma_0} \sum_{j=1}^{m} \frac{1}{s_j} \right).$$

The new theorem has the same form as the one in [24]. Similarly to the classical fitness-level argument, it assumes an f-based partition $(A_1, \ldots, A_{m+1})$ of the search space $\mathcal{X}$. Each subset A_j is called a *fitness level*. Condition (C1) specifies that for each fitness level A_j, the "upgrade probability", i.e., the probability that an individual in fitness level A_j is mutated into a higher fitness level, is bounded from below by a parameter s_j. Condition (C2) requires that there exists a lower bound p_0 on the probability that an individual will not "downgrade" to a lower fitness level. For example, in the classical setting of bitwise mutation with mutation rate $1/n$, it suffices to pick any parameter $p_0 \leq (1 - 1/n)e^{-1}$, which is less than the probability of not flipping any bits.

Condition (C3) requires that the selective pressure (see Definition 3) induced by the selection mechanism is sufficiently strong. The probability of selecting one of the fittest $\gamma\lambda$ individuals in the population, and not downgrading the individual via mutation (probability p_0), should exceed γ by a factor of at least $1 + \delta$. In applications, the parameter δ may depend on the optimisation problem and the selection mechanism.

The last condition (C4) requires that the population size λ is sufficiently large. The required population size depends on the number of fitness levels m, the parameter δ (c and ε are functions of δ) which characterises the selective pressure, and the upgrade probabilities s_j which are problem-dependent parameters. A population size of $\lambda = \Theta(\ln n)$, where n is the number of problem dimensions, is sufficient for many pseudo-Boolean functions.

If all the conditions (C1-4) are satisfied, then an upper bound on the expected runtime of the algorithm is guaranteed. Note that the upper bound has an additive term which is similar to the classical fitness-level technique, e.g., applied to the $(1 + 1)$ EA. However, this does not prevent us from proving that population-based algorithms are better choices than single-individual solution approaches, as we will see typical examples in the second part of the paper.

The refined theorem makes the relationship between the expected runtime and the parameters, including δ, explicit. Most notably, the assumption about δ being a constant as required by the old theorem in [24] is removed. This allows the new theorem to be applied in more complex settings. The following corollaries provide a simplification for (C4) and the corresponding result for arbitrary $\delta \in (0, 1]$.

Corollary 9. *For any $\delta \in (0, 1]$, condition (C4) of Theorem 8 holds if*

(C4') $\lambda \geq \dfrac{8}{\gamma_0 \delta^2} \left(\ln \left(\dfrac{m}{\gamma_0 \delta^7 s_*} \right) + 11 \right).$

Proof. For $\delta \in (0, 1]$, we have $\varepsilon = \min\{\delta/2, 1/2\} = \delta/2$. Therefore $1/\varepsilon = 2/\delta$, $1/c = 24/\varepsilon^4 = 384/\delta^4$, $1/a = (2/\gamma_0)(1/\delta + 1/\delta^2) \leq 4/(\gamma_0 \delta^2)$. So

$$\frac{2}{a} \ln \left(\frac{16m}{ac \varepsilon s_*} \right) \leq \frac{8}{\gamma_0 \delta^2} \ln \left(\frac{4 \cdot 2 \cdot 384 \cdot 16 \cdot m}{\gamma_0 \delta^7 s_*} \right) < \frac{8}{\gamma_0 \delta^2} \left(\ln \left(\frac{m}{\gamma_0 \delta^7 s_*} \right) + 11 \right).$$

Therefore, (C4') implies (C4). $\qquad\square$

If we further have $m > 1$, $1/\delta \in \text{poly}(m)$, $1/s_* \in \text{poly}(m)$ and $1/\gamma_0 \in O(1)$ then there exists a constant b such that (C4) is satisfied with $\lambda \geq (b/\delta^2) \ln m$.

Corollary 10. *For any $\delta \in (0, 1]$, the expected runtime of Algorithm 1 satisfying conditions (C1-4) of Theorem 8 is*

$$\mathbf{E}\left[T\right] \leq \frac{1536}{\delta^5} \left(m\lambda \left(1 + \ln \left(1 + \frac{\delta^4 \lambda}{384} \right) \right) + \frac{1}{\gamma_0} \sum_{j=1}^{m} \frac{1}{s_j} \right).$$

Proof. For $\delta \in (0, 1]$, we have $\varepsilon = \min\{\delta/2, 1/2\} = \delta/2$. Hence $2/(c\varepsilon) = 48/\varepsilon^5 = 1536/\delta^5$. In addition, we have $\frac{p_0}{(1+\delta)\gamma_0} \leq \frac{1}{\gamma_0}$, so by Theorem 8

$$\mathbf{E}\left[T\right] \leq \frac{1536}{\delta^5} \left(m\lambda \left(1 + \ln \left(1 + \frac{\delta^4 \lambda}{384} \right) \right) + \frac{1}{\gamma_0} \sum_{j=1}^{m} \frac{1}{s_j} \right).$$

$\qquad\square$

If δ is bounded from below by a constant, then the following corollary holds.

Corollary 11. *In addition to (C1-4) of Theorem 8, if γ_0 and δ can be fixed as constants with respect to m, then there exists a constant C such that*

$$\mathbf{E}\left[T\right] \leq C \left(m\lambda \ln \lambda + \sum_{j=1}^{m} \frac{1}{s_j} \right).$$

In the case of constant δ, the first term of the expected runtime is reduced from $O(m\lambda^2)$ as previously stated in [24] to $O(m\lambda \ln \lambda)$. In other words, the overhead when increasing the population size is significantly reduced compared to previously thought.

The main proof idea of the theorem is to estimate the expected time for the algorithm to leave each level j. The expected runtime is then their sum, given that population does not lose its best solutions too often. This condition is shown to hold using a Chernoff bound, e.g., relying on a sufficiently large population size. The process of leaving the current level j is pessimistically assumed to follow two phases: first the algorithm waits for the arrival of an *advanced* individual, i.e., the one at level of at least $j + 1$; then the γ_0-upper portion of the population is filled up with advanced individuals, at least through the selection of the existing ones and the application of harmless mutation. The algorithm is considered to have left level j when there are at least $\lceil \gamma_0 \lambda \rceil$ advanced individuals in the population. The full proof is formalised with drift analysis as follows.

Proof of Theorem 8. We use the following notation. The number of individuals with fitness level at least j at generation t is denoted by X_t^j for $j \in [m+1]$. The current fitness level of the population at generation t is denoted by Z_t, and $Z_t = \ell$ if $X_t^\ell \geq \lceil \gamma_0 \lambda \rceil$ and $X_t^{\ell+1} < \gamma_0 \lambda$. Note that Z_t is uniquely defined, as it is the fitness level of the γ_0-ranked individual at generation t. We are interested in the first point in time that $X_t^{m+1} \geq \gamma_0 \lambda$, or equivalently $Z_t = m+1$. This stopping time gives a valid upper bound on the runtime.

Since condition (C3) holds for all P, we will write $\beta(\gamma)$ instead of $\beta(\gamma, P)$ to simplify the notation. For each level j, define a parameter $q_j := 1 - (1 - \beta(\gamma_0)s_j)^\lambda$. Note that q_j is a lower bound on the probability of generating at least one individual at fitness level strictly better than j in the next generation conditioned on the event that the $\lceil \gamma_0 \lambda \rceil$ best individuals of the current population are at level j. Due to Lemma 31, we have

$$q_j \geq \frac{\beta(\gamma_0)\lambda s_j}{\beta(\gamma_0)\lambda s_j + 1}.$$

Using the following potential function, the theorem can now be proven solely relying on the additive drift argument.

$$g(P_t) := g_1(P_t) + g_2(P_t)$$
$$\text{with } g_1(P_t) := (m - Z_t)\ln(1 + c\lambda) - \ln(1 + cX_t^{Z_t+1})$$
$$\text{and } g_2(P_t) := \frac{1}{q_{Z_t}e^{\kappa X_t^{Z_t+1}}} + \sum_{j=Z_t+1}^{m} \frac{1}{q_j} \text{ with some } \kappa \in (0, \delta).$$

Here, parameter κ serves the purpose of "smoothing" the transition so that the progress depends mostly on g_1 when $X_t^{Z_t+1} > 0$ and depends mostly on g_2 when $X_t^{Z_t+1} = 0$. We will only need to show that κ exists in the given interval in order to use Lemma 6 later on. Note also that as far as $Z_t < m+1$, i.e., the process before the stopping time, we have $g(P_t) > 0$.

The function $g(P_t)$ is bounded from above by,

$$g(P_t) \leq m\ln(1 + c\lambda) + \sum_{j=1}^{m} \frac{1}{q_j}$$
$$\leq m\ln(1 + c\lambda) + \sum_{j=1}^{m} \left(1 + \frac{1}{\beta(\gamma_0)\lambda s_j}\right)$$
$$= m(1 + \ln(1 + c\lambda)) + \frac{1}{\beta(\gamma_0)\lambda} \sum_{j=1}^{m} \frac{1}{s_j}. \tag{3}$$

At generation t, we use $R = Z_{t+1} - Z_t$ to denote the random variable describing the next progress in fitness levels. To simplify further writing, let $\ell := Z_t$, $i := X_t^{\ell+1}$, $X := X_{t+1}^{\ell+1}$, and $\Delta := g(P_t) - g(P_{t+1}) = \Delta_1 + \Delta_2$ where

$$\Delta_1 := g_1(P_t) - g_1(P_{t+1}) = R\ln(1 + c\lambda) + \ln\left(\frac{1 + X_{t+1}^{\ell+R+1}}{1 + ci}\right),$$

$$\Delta_2 := g_2(P_t) - g_2(P_{t+1}) = \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+R}e^{\kappa X_{t+1}^{\ell+R+1}}} + \sum_{j=\ell+1}^{\ell+R} \frac{1}{q_j}.$$

11

Let $\mathscr{F}_t$ be the filtration induced by P_t. Define $\mathcal{E}_t$ to be the event that the population in the next generation does not fall down to a lower level, $\mathcal{E}_t : Z_{t+1} \geq Z_t$, and $\bar{\mathcal{E}}_t$ the complementary event. We have

$$\mathbf{E}_t\left[\Delta\right] = \left(1 - \Pr{}_t\left(\bar{\mathcal{E}}_t\right)\right)\mathbf{E}_t\left[\Delta \mid \mathcal{E}_t\right] + \Pr{}_t\left(\bar{\mathcal{E}}_t\right)\mathbf{E}_t\left[\Delta \mid \bar{\mathcal{E}}_t\right]$$
$$= \mathbf{E}_t\left[\Delta \mid \mathcal{E}_t\right] - \Pr{}_t\left(\bar{\mathcal{E}}_t\right)\left(\mathbf{E}_t\left[\Delta \mid \mathcal{E}_t\right] - \mathbf{E}_t\left[\Delta \mid \bar{\mathcal{E}}_t\right]\right).$$

We first compute the *conditional forward drift* $\mathbf{E}_t\left[\Delta \mid \mathcal{E}_t\right]$. For all $i \geq 0$ and $\ell \in [m]$, it holds for any integer $R \in [m - \ell + 1]$ that

$$\Delta = R\ln(1 + c\lambda) + \ln\left(\frac{1 + X_{t+1}^{\ell+R+1}}{1 + ci}\right)$$

$$+ \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+R}e^{\kappa X_{t+1}^{\ell+R+1}}} + \sum_{j=\ell+1}^{\ell+R}\frac{1}{q_j}$$

$$\geq \ln(1 + c\lambda) + \ln\left(\frac{1}{1 + ci}\right) + \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_{\ell+R}} + \sum_{j=\ell+1}^{\ell+R}\frac{1}{q_j}$$

$$= \ln\left(\frac{1 + c\lambda}{1 + ci}\right) + \frac{1}{q_\ell e^{\kappa i}} + \sum_{j=\ell+1}^{\ell+R-1}\frac{1}{q_j}$$

$$\geq \ln\left(\frac{1 + c\lambda}{1 + ci}\right) + \frac{1}{q_\ell e^{\kappa i}},$$

and for $R = 0$ that

$$\Delta = \ln\left(\frac{1 + cX}{1 + ci}\right) + \frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_\ell e^{\kappa X}} =: \Delta_0$$

$$\leq \ln\left(\frac{1 + c\lambda}{1 + ci}\right) + \frac{1}{q_\ell e^{\kappa i}}.$$

Given $\mathscr{F}_t$ and conditioned on $\mathcal{E}_t$, the support of R is indeed $\{0\} \cup [m - \ell + 1]$ and from the above relations it holds that $\Delta_0 \preceq \Delta$. Hence, $\mathbf{E}_t\left[\Delta \mid \mathcal{E}_t\right] \geq \mathbf{E}_t\left[\Delta_0 \mid \mathcal{E}_t\right]$ and we only focus on $R = 0$ to bound the drift from below. We separate two cases: $i = 0$ (event $\mathcal{Z}_t$) and $i \geq 1$ (event $\bar{\mathcal{Z}}_t$). Recall that each individual is generated independently from each other, so during $\bar{\mathcal{Z}}_t$ we have that $X \sim \text{Bin}(\lambda, p)$ where $p \geq \beta(i/\lambda)p_0 \geq (i/\lambda)(1+\delta)$. The inequality is due to condition (C3). Hence by Lemma 5, it holds for $i \geq 1$ (event $\bar{\mathcal{Z}}_t$) that

$$\mathbf{E}_t\left[\Delta_1 \mid \mathcal{E}_t, \bar{\mathcal{Z}}_t\right] \geq \mathbf{E}_t\left[\ln\left(\frac{1 + cX}{1 + ci}\right) \mid \mathcal{E}_t, \bar{\mathcal{Z}}_t\right] \geq c\varepsilon,$$

$$\mathbf{E}_t\left[\Delta_1 \mid \mathcal{E}_t, \mathcal{Z}_t\right] \geq \mathbf{E}_t\left[\ln(1) \mid \mathcal{E}_t, \mathcal{Z}_t\right] = 0.$$

By Lemma 6, we have $e^{-\kappa i} \geq \mathbf{E}\left[e^{-\kappa X}\right]$ for $i \geq 1$, thus

$$\mathbf{E}_t\left[\Delta_2 \mid \mathcal{E}_t, \bar{\mathcal{Z}}_t\right] \geq \frac{1}{q_\ell}(e^{-\kappa i} - \mathbf{E}_t\left[e^{-\kappa X} \mid \mathcal{E}_t, \bar{\mathcal{Z}}_t\right]) \geq 0.$$

For $i = 0$, $\Pr_t\left(X \geq 1 \mid \mathcal{E}_t, \mathcal{Z}_t\right) \geq q_\ell$, so

$$\mathbf{E}_t\left[\Delta_2 \mid \mathcal{E}_t, \mathcal{Z}_t\right] \geq \Pr{}_t\left(X \geq 1 \mid \mathcal{E}_t, \mathcal{Z}_t\right)\mathbf{E}_t\left[\frac{1}{q_\ell e^{\kappa i}} - \frac{1}{q_\ell e^{\kappa X}} \mid \mathcal{E}_t, \mathcal{Z}_t, X \geq 1\right]$$

12

$$\geq q_\ell \cdot \frac{1}{q_\ell}(e^{-\kappa \cdot 0} - e^{-\kappa \cdot 1}) = 1 - e^{-\kappa}.$$

So the conditional forward drift is $\mathbf{E}_t[\Delta \mid \mathcal{E}_t] \geq \min\{c\varepsilon, 1 - e^{-\kappa}\}$. Furthermore, κ can be picked in the non-empty interval $(-\ln(1 - c\varepsilon), \delta) \subset (0, \delta)$, so that $1 - e^{-\kappa} > c\varepsilon$ and $\mathbf{E}_t[\Delta \mid \mathcal{E}_t] \geq c\varepsilon$.

Next, we compute the *conditional backward drift*. This can be done for the worst case, i.e., the potential is increased from 0 to the maximal value. From (3) and $s_j \geq s_*$ for all $j \in [m]$, we have

$$\mathbf{E}_t\left[\Delta \mid \bar{\mathcal{E}}_t\right] \geq -\left(m(1 + \ln(1 + c\lambda)) + \frac{1}{\beta(\gamma_0)\lambda} \sum_{j=1}^m \frac{1}{s_j}\right)$$

$$\geq -m\left(1 + \ln(1 + c\lambda) + \frac{1}{\beta(\gamma_0)\lambda s_*}\right).$$

The probability of not having event $\mathcal{E}_t$ is computed as follows. Recall that $X_t^\ell \geq \lceil \gamma_0 \lambda \rceil$ and X_{t+1}^ℓ is binomially distributed with a probability of at least $\beta(\gamma_0)p_0 \geq (1 + \delta)\gamma_0$ due to (C3), so $\mathbf{E}_t\left[X_{t+1}^\ell\right] \geq (1 + \delta)\gamma_0\lambda$. The event $\bar{\mathcal{E}}_t$ happens when the number of individuals at fitness level ℓ is strictly less than $\lceil \gamma_0 \lambda \rceil$ in the next generation. The probability of such an event is

$$\mathrm{Pr}_t\left(\bar{\mathcal{E}}_t\right) = \mathrm{Pr}_t\left(X_{t+1}^\ell < \lceil \gamma_0 \lambda \rceil\right) \leq \mathrm{Pr}_t\left(X_{t+1}^\ell \leq \gamma_0 \lambda\right)$$

$$= \mathrm{Pr}_t\left(X_{t+1}^\ell \leq \left(1 - \frac{\delta}{1 + \delta}\right)(1 + \delta)\gamma_0 \lambda\right)$$

$$\leq \mathrm{Pr}_t\left(X_{t+1}^\ell \leq \left(1 - \frac{\delta}{1 + \delta}\right)\mathbf{E}_t\left[X_{t+1}^\ell\right]\right)$$

$$\leq \exp\left(-\frac{\delta^2 \mathbf{E}_t\left[X_{t+1}^\ell\right]}{2(1 + \delta)^2}\right) \quad \text{due to Lemma 27}$$

$$\leq \exp\left(-\frac{\delta^2(1 + \delta)\gamma_0 \lambda}{2(1 + \delta)^2}\right) = e^{-a\lambda}.$$

Recall condition (C4) on the population size that

$$\lambda \geq \frac{2}{a}\ln\left(\frac{16m}{ac\varepsilon s_*}\right) \Rightarrow \frac{8m}{ac\varepsilon s_*} \leq \frac{e^{\frac{a\lambda}{2}}}{2} \leq \frac{e^{a\lambda}}{a\lambda} \quad \text{due to Lemma 32}$$

$$\Rightarrow \mathrm{Pr}_t\left(\bar{\mathcal{E}}_t\right) \leq e^{-a\lambda} \leq \frac{c\varepsilon s_*}{8m\lambda}.$$

Altogether, we have the drift of

$$\mathbf{E}_t[\Delta] \geq c\varepsilon - \frac{c\varepsilon s_*}{8m\lambda}\left(c\varepsilon + m\left(1 + \ln(1 + c\lambda) + \frac{1}{\beta(\gamma_0)\lambda s_*}\right)\right)$$

$$= c\varepsilon - \frac{c\varepsilon}{8\lambda}\left(\frac{c\varepsilon s_*}{m} + s_* + s_* \ln(1 + c\lambda) + \frac{1}{\beta(\gamma_0)\lambda}\right)$$

$$\geq c\varepsilon - \frac{c\varepsilon}{8\lambda}(1 + 1 + c\lambda + 1) = c\varepsilon - \frac{c\varepsilon}{8\lambda}\lambda\left(c + \frac{3}{\lambda}\right)$$

$$\geq \frac{c\varepsilon}{2}.$$

13

The second inequality makes use of (C3), which is $\beta(\gamma_0)\lambda \geq (1+\delta)\gamma_0\lambda/p_0 > \gamma_0\lambda \geq 1$. Finally, applying Theorem 4 with the above drift, with the maximal potential from (3), and again using $1/\beta(\gamma_0) \leq p_0/((1+\delta)\gamma_0)$ from (C3) give the expected runtime in terms of variation-selection steps

$$\mathbf{E}\left[T\right] \leq \frac{2}{c\varepsilon}\left(m\lambda(1 + \ln(1+c\lambda)) + \frac{p_0}{(1+\delta)\gamma_0}\sum_{j=1}^{m}\frac{1}{s_j}\right).$$

□

We now discuss some limitations of the current result and possible directions to its future improvement. First, one can observe from the proof that the bound for λ/q_j is loosely estimated using Lemma 31 (e.g., see the partial sum argument in its proof). This implies the second term in the runtime being proportional to $\sum_{i=1}^{m}1/s_j$ which is similar to the classical fitness level [34]. So one may wonder if this term could have been improved with a better estimation for λ/q_j. The answer is no because of the following result.

Lemma 12. *For any $\gamma_0, \beta(\gamma_0), s_j \in (0,1)$ and any $\lambda \in \mathbb{N}$, we have*

$$\frac{\lambda}{1 - (1 - \beta(\gamma_0)s_j)^\lambda} > \frac{1}{s_j}$$

.

Proof. From the condition, we have $-\beta(\gamma_0)s_j \in (-1,0)$, it follows from Lemma 29 that $(1 - \beta(\gamma_0)s_j)^\lambda \geq 1 - \beta(\gamma_0)s_j\lambda$, then

$$\frac{\lambda}{1 - (1 - \beta(\gamma_0)s_j)^\lambda} \geq \frac{1}{\beta(\gamma_0)s_j} > \frac{1}{s_j}.$$

□

The bigger term is the expected waiting time for Algorithm 1 to generate one advanced individual. This is similar to the waiting time of a $(1+\lambda)$ EA to leave level j. So unless parallel implementations are considered it is impossible for the current approach to improve the second term $\sum_{j=1}^{m}1/s_j$ in the runtime.

In addition, in the second phase to leave the current fitness level (see the proof idea before the actual proof), we only consider the spreading of advanced solutions by harmless mutations whilst completely ignoring occasional updates from the current level. This could be a very pessimistic choice, and it is crucial in the future to fully understand the population dynamic during this phase to determine more precise expressions for the runtime.

In the following sections, we discuss the applications of the new theorem and its corollaries.

4   Optimisation of pseudo-Boolean functions

In this first application, we use Theorem 8 to improve the results of [24] on expected optimisation times of pseudo-Boolean functions using Algorithm 1 with bitwise mutations as variation operators (the so-called non-elitist EAs). The same notation as in the literature, χ/n, is used to denote the probability of flipping a bit position in a bitwise mutation. Formally, if x' is sampled from

$p_{\text{mut}}(x)$, then independently for each $i \in [n]$

$$x_i' = \begin{cases} 1 - x_i & \text{with probability } \chi/n, \text{ and} \\ x_i & \text{with probability } 1 - \chi/n. \end{cases}$$

The following selection mechanisms are considered:

- In a *tournament selection* of size $k \geq 2$, denoted as k-tournament, k individuals are uniformly sampled from the current population with replacement then the fittest one is selected. Ties are broken uniformly at random.
- In *ranking selection*, each individual is assigned a rank between 0 and 1, where the best one has rank 0 and the worst one has rank 1. Following [12], a function $\alpha : \mathbb{R} \to \mathbb{R}$ is considered a ranking function if $\alpha(x) \geq 0$ for all $x \in [0,1]$, and $\int_0^1 \alpha(x)dx = 1$. The selection mechanism chooses individuals such that the probability of selecting individuals ranked γ or better is $\int_0^\gamma \alpha(x)dx$. Note that this finite integral coincides with our definition of $\beta(\gamma)$. The *linear ranking* selection uses the ranking function $\alpha(x) := \eta(1 - 2x) + 2x$ with $\eta \in (1, 2]$.
- In a (μ, λ)-*selection*, parent solutions are selected uniformly at random among the best μ out of λ individuals in the current population.
- The *fitness proportionate selection* with power scaling parameter $\nu \geq 1$ is defined for maximisation problems as follows

$$\forall i \in [\lambda] \quad p_{\text{sel}}(i \mid P_t, f) := \frac{f(P_t(i))^\nu}{\sum_{j=1}^\lambda f(P_t(j))^\nu}.$$

Setting $\nu = 1$ gives the standard version of fitness proportionate selection.

4.1 Tighter upper bounds

For the optimisation of pseudo-Boolean functions and bitwise mutation operators with χ being a constant, it is already proven in [24] that δ and γ_0 of Theorem 8 can be fixed as constants by appropriate parameterisation of the selection mechanisms. The following lemma summarises these settings.

Lemma 13. *For any constants $\delta' > 0$ and $p_0 \in (0, 1)$, there exist constants $\delta > 0$ and $\gamma_0 \in (0, 1)$ such that condition (C3) of Theorem 8 is satisfied for*

- *k-tournament selection with $k \geq (1 + \delta')/p_0$,*
- *linear ranking selection with $\eta \geq (1 + \delta')/p_0$,*
- *(μ, λ)-selection with $\lambda/\mu \geq (1 + \delta')/p_0$,*
- *fitness proportionate selection on 1-distinctive functions with the maximal function value $f_{\max}$ and $\nu \geq \ln(2/p_0)f_{\max}$.*

Proof. See Lemmas 5, 6, 7 and 8 in [24]. □

Once δ and γ_0 are fixed as constants, Corollary 11 provides the runtime for the following functions.

$$\text{ONEMAX}(x) := \sum_{i=1}^n x_i = |x|_1,$$

Selection Mechanism	Parameter
Fitness proportionate (*)	$\nu > f_{\max} \ln(2e^\chi)$
Linear ranking	$\eta > e^\chi$
k-Tournament	$k > e^\chi$
(μ, λ)	$\lambda > \mu e^\chi$

Problem	Population size	$\mathbf{E}[T]$
ONEMAX	$\lambda \geq c \ln n$	$O(n\lambda \ln \lambda)$
LEADINGONES	$\lambda \geq c \ln n$	$O(n\lambda \ln \lambda + n^2)$
LINEAR	$\lambda \geq c \ln n$	$O(n\lambda \ln \lambda + n^2)$
ℓ-UNIMODAL	$\lambda \geq c \ln(n\ell)$	$O(\ell\lambda \ln \lambda + n\ell)$
JUMP$_r$	$\lambda \geq cr \ln n$	$O(n\lambda \ln \lambda + (n/\chi)^r)$

Table 1: Expected runtime in terms of fitness evaluations of Algorithm 1 with corresponding parameter settings. For clarity, all constant factors are omitted or summarised by a sufficiently large constant c. The result for fitness proportionate selection (*) only holds for the functions or their subclasses satisfying the 1-distinctive property, for example the ONEMAX function.

$$\text{LEADINGONES}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_i,$$

$$\text{JUMP}_r(x) := \begin{cases} |x|_1 + 1 & \text{if } |x|_1 \leq n - r \text{ or } |x|_1 = n \\ 0 & \text{otherwise} \end{cases},$$

$$\text{LINEAR}(x) := \sum_{j=1}^{n} c_i x_i.$$

We also analyse the runtime of ℓ-UNIMODAL functions. A pseudo-Boolean function f is called *unimodal* if every bitstring x is either optimal, or has a Hamming-neighbour x' such that $f(x') > f(x)$. We say that a unimodal function is ℓ-UNIMODAL if it has ℓ distinct function values $f_1 < f_2 < \cdots < f_\ell$. Note that LEADINGONES is a particular case of ℓ-UNIMODAL with $\ell = n + 1$ and ONEMAX is a special case of LINEAR with $c_i = 1$ for all $i \in [n]$. Corresponding to the new results reported in Table 1, the following theorem improves the results previously reported [24].

Theorem 14. *Algorithm 1 with bitwise mutation rate χ/n for any constant $\chi > 0$, and where p_{sel} is either linear ranking selection, k-tournament selection, or (μ, λ)-selection where the parameter settings satisfy column "Parameter" in the first part of Table 1 has expected runtimes as indicated in column $\mathbf{E}[T]$ in the second part, given the population sizes respecting column "Population Size" of the same part. The result of fitness proportionate selection only holds for the functions or their subclasses satisfying the 1-distinctive property.*

Corollary 15. *Algorithm 1 with either linear ranking, or fitness proportionate or k-tournament, or (μ, λ)-selection optimises ONEMAX in $O(n \ln n \ln \ln n)$ and LEADINGONES in $O(n^2)$ expected fitness evaluations. With either linear ranking, or k-tournament, or (μ, λ) selection, the algorithm optimises LINEAR in*

$O(n^2)$ *and* ℓ-UNIMODAL *with* $\ell \leq n^d$ *in* $O(n\ell)$ *expected fitness evaluations for any constant* $d \in \mathbb{N}$.

Proof. The corollary is obtained from the theorem by taking the smallest population size. It is clear that we have $O(n \ln n \ln \ln n)$ for ONEMAX and $O(n(\ln n \ln \ln \ln n + n)) = O(n^2)$ for LINEAR and LEADINGONES. In the case of ℓ-UNIMODAL, because $\ell \leq n^d$ we have $c \ln(n\ell) \leq c(d+1) \ln n$. It suffices to pick $\lambda = c(d+1) \ln n$ and the runtime is bounded by $O(\ell(\ln n\ell \ln \ln n\ell + n)) = O(n\ell)$.

The proof of the theorem is similar to the one in [24], except the main tool is Corollary 11. We recall those arguments shortly as follows. The f-based partitions and upgrade probabilities are similar to the ones in [21, 22, 33]. For a LINEAR function f, without loss of generality we assume the weights $c_1 \geq c_2 \geq \cdots \geq c_n \geq 0$, then set $m := n$ and choose the partition

$$A_j := \left\{ x \in \{0,1\}^n \mid \sum_{i=1}^{j} c_i \leq f(x) < \sum_{i=1}^{j+1} c_i \right\} \text{ and } A_{m+1} := \{1^n\}.$$

For ONEMAX, LEADINGONES, and ℓ-UNIMODAL functions, with ℓ distinct function values $f_1 < \cdots < f_\ell$, we set $m := \ell - 1$ and use the partition

$$A_j := \{x \in \{0,1\}^n \mid f(x) = f_j\}.$$

For LINEAR functions, and for ℓ-UNIMODAL functions which also include the case $\ell = n$ of LEADINGONES, it is sufficient to flip one specific bit, and no other bits to reach a higher fitness level. For these functions, we therefore choose for all j the upgrade probabilities $s_* := (\chi/n)(1 - \chi/n)^{n-1} =: s_j$, so $s_j, s_* \in \Omega(1/n)$.

For ONEMAX, it is sufficient to flip one of the $n - j$ 0-bits, and no other bits to escape fitness level j. For this function, we therefore choose the upgrade probabilities $s_j := (n-j)(\chi/n)(1-\chi/n)^{n-1}$ and $s_* := s_{n-1}$, thus $s_j \in \Omega(1-j/n)$ and $s_* \in \Omega(1/n)$.

For JUMP$_r$, we set $m := n - r + 2$, and choose

$$\begin{aligned} A_1 &:= \{x \in \{0,1\}^n \mid n - r < |x|_1 < n\}, \\ A_j &:= \{x \in \{0,1\}^n \mid |x|_1 = j - 2\} \quad \forall j \in [2, n - r + 2], \\ A_{m+1} &:= \{1^n\}. \end{aligned}$$

In order to escape fitness level A_1 and A_m, it is sufficient to flip at most $\lceil r/2 \rceil$ and r 0-bits respectively and no other bits. For the other fitness levels A_j, it suffices to flip one of $n-j$ 0-bits and no other bits. Hence, we choose the upgrade probabilities $s_1 := (\chi/n)^{\lceil r/2 \rceil}(1-\chi/n)^{n-\lceil r/2 \rceil}$, $s_m := (\chi/n)^r(1-\chi/n)^{n-r} =: s_*$ and $s_j := (n-j)(\chi/n)(1-\chi/n)^{n-1}$ for all $j \in [2, n-r+2]$, thus $s_1, s_m, s_* \in \Omega((\chi/n)^r)$ and $s_j \in \Omega(1-j/n)$.

By the above partitions, the first condition (C1) is satisfied. Next, we set the parameter p_0 to be a lower bound of the probability of not flipping any bits $(1 - \chi/n)^n$, condition (C2) is therefore satisfied. For any constant $\theta \in (0,1)$, it holds for all $n > 2\chi^2/(-\ln(1-\theta))$ that

$$\left(1 - \frac{\chi}{n}\right)^n \geq \left[\left(1 - \frac{\chi}{n}\right)^{\frac{n}{\chi} - 1}\right]^{\chi + \frac{2\chi^2}{n}} \geq (1 - \theta)e^{-\chi}.$$

We can set $p_0 = (1-\theta)e^{-\chi}$ which is a constant. According to the parameter settings of Table 1 and Lemma 13 there exist constants δ and γ_0 so that (C3) is satisfied. Regarding (C4), for LINEAR functions (including ONEMAX), $m/s_* = O(n^2)$. For ℓ-UNIMODAL functions (including LEADINGONES), $m/s_* = O(n\ell)$. For JUMP$_r$, $m/s_* = O(n^{r+1}/\chi^r)$. Condition (C4) is satisfied if the population size λ is set according to column "Population size" of Table 1. All conditions are satisfied, and the upper bounds in Table 1 follow. $\qquad\square$

Note that for most of the functions in the corollary, the population does not incur any overhead compared with the (1+1) EA. The only exception is ONEMAX for which there is a small overhead factor of $O(\ln\ln n)$.

4.2 Polynomial runtime with standard fitness proportionate selection

It is well-known that fitness proportionate selection without scaling ($\nu = 1$) is inefficient, i.e., it requires exponential optimisation time on simple functions (see [14, 24, 30]). However, these previous studies often concern the standard mutation rate $1/n$, thus it may be possible to optimise ONEMAX and LEADINGONES in expected polynomial time without scaling but with a different mutation rate. Note also that the result of the previous section does not confirm such a possibility because the condition of Lemma 13 is equivalent to $\nu \geq n\ln 2 + n\ln(1/p_0)$ which cannot be satisfied for $\nu = 1$ and any $p_0 \in (0,1)$.

Based on Theorem 8, we can indeed determine sufficient conditions for polynomial optimisation time of those functions. The conditions imply the mutation rate being reduced to $\Theta(1/n^2)$ and a sufficiently large population.

Theorem 16. *Algorithm 1 with standard fitness proportionate selection, bit-wise mutation where $\chi = 1/(6n)$ and population $\lambda = bn^2\ln n$ for some constant b, optimises ONEMAX and LEADINGONES in $O(n^8 \ln n)$ expected fitness evaluations.*

Proof. We use the same partitions as in the proof of Theorem 14. Again for ONEMAX, it suffices to flip one of the $n - j$ bits and to keep the others unchanged to leave level A_j. By Lemma 29, the associated probability $(n-j)(\chi/n)(1-\chi/n)^{n-1} \geq (n-j)(\chi/n)(1-\chi/n)^n$ can be bounded from below by $(n-j)(\chi/n)(1-\chi) \geq (n-j)(1/(6n^2))(1-1/6) = (n-j)(5/(36n^2)) =: s_j$ and in the worst case $5/(36n^2) =: s_*$. For LEADINGONES, it suffices to flip the leftmost 0-bit and keep the others unchanged, hence $s_j := (5/(36n^2)) =: s_*$. We pick p_0 as $(1-\chi/n)^n = (1-\chi/n)^{(n/\chi-1)n\chi/(n-\chi)} \geq e^{-\chi(1+\chi/(n-\chi))} \geq e^{-2\chi} = e^{-1/(3n)} =: p_0$. So conditions (C1) and (C2) of Theorem 8 are satisfied with these choices.

Given that there are at least $\gamma\lambda$ individuals at level A_j^+, define f_γ to be the fitness value of the γ-ranked individual. Because of the 1-distinctive property of the two functions, a lower bound of $\beta(\gamma)$ can be deduced by assuming that all individuals below the γ-ranked one have fitness value $f_\gamma - 1$.

$$\forall \gamma \leq 1/2 \quad \beta(\gamma) \geq \frac{f_\gamma\gamma\lambda}{(\lambda-\gamma\lambda)(f_\gamma-1)+f_\gamma\gamma\lambda} = \frac{\gamma}{(1-\gamma)(1-1/f_\gamma)+\gamma}$$

$$\geq \frac{\gamma}{(1-\gamma)(1-1/n)+\gamma} = \frac{\gamma}{1-1/n+\gamma/n}$$

18

$$\geq \frac{\gamma}{1 - 1/n + 1/(2n)} \geq \gamma e^{1/(2n)}.$$

Then,

$$\forall \gamma \leq 1/2 \quad \beta(\gamma)p_0 \geq e^{1/(2n)-1/(3n)}\gamma = e^{1/6n}\gamma \geq (1 + 1/(6n))\gamma.$$

Therefore, (C3) is satisfied with $\gamma_0 = 1/2$ and $\delta = 1/(6n)$. It follows from Corollary 9 that there exists a constant b such that (C4) is satisfied with $\lambda = (b/\delta^2) \ln n = bn^2 \ln n$. Since all conditions are satisfied, the expected time to optimise ONEMAX follows from Corollary 10.

$$O\left(n^5 \left(n^3 \ln n(1 + \ln(1 + (1/n^4)n^2 \ln n)) + \sum_{j=1}^{n-1} \frac{n^2}{n-j}\right)\right) = O\left(n^8 \ln n\right).$$

Similarly, the expected time to optimise LEADINGONES is

$$O\left(n^5 \left(n^3 \ln n(1 + \ln(1 + (1/n^4)n^2 \ln n)) + \sum_{j=1}^{n-1} n^2\right)\right) = O\left(n^8 \ln n\right). \qquad \square$$

Note that the selective pressure δ is small in fitness proportionate selection without scaling. However, with sufficiently low mutation rate, the low selective pressure does not prevent the algorithm from optimising standard functions within expected polynomial time. In the next section, we explore further consequences of this observation in the scenarios of optimisation under incomplete information.

5 Optimisation under incomplete information

In real-world optimisation problems, complete and accurate information about the quality of candidate solutions is either not available or prohibitively expensive to obtain. Optimisation problems with noise, dynamic objective function or stochastic data, generally known as *optimisation under uncertainty* [19], are examples of the unavailability. On the other hand, expensive evaluations often occur in engineering, especially in structural and engine design. For example, in order to determine the fitness of a solution, the solution has to be put in a real experiment or a simulation which may be time/resource consuming or even requiring collection and processing of a large amount of data. Such complex tasks give rise to the use of *surrogate model methods* [18] to assist EAs where a cheap and approximate procedure fully or partially replaces the expensive evaluations. The full replacement is equivalent to the case of unavailability. We summarise this kind of problem as optimisation only relying on *imprecise*, *partial* or *incomplete* information (for now) about the problem.

While EAs have been widely and successfully used in this challenging area of optimisation [18, 19], only few rigorous theoretical studies have been dedicated to fully understand the behaviours of EAs under such environments. In [8, 9, 11], the inefficiency of $(1 + 1)$ EA has been rigorously demonstrated for noisy and dynamic optimisation of ONEMAX and LEADINGONES. The reason behind this inefficiency is that in such environments, it is difficult for the algorithm to

compare the quality of solutions, e. g., often it will choose the wrong candidate. On the other hand, such effects could be reduced by having a population, for example the model of infinite population in [27] or finite elitist populations in [11].

In this section, we initiate runtime analysis of evolutionary algorithms where only partial or incomplete information about fitness is available. Two scenarios are investigated: (i) in *partial evaluation of solutions*, only a small amount of information about the problem is revealed in each fitness evaluation, we formulate a model that makes this scenario concrete for pseudo-Boolean optimisation (ii) in *partial evaluation of populations*, not all individuals in the population are evaluated. For both scenarios, we rigorously prove that given appropriate parameterisation, non-elitist evolutionary algorithms can optimise many functions in expected polynomial time even with little information available.

5.1 Partial evaluation of pseudo-Boolean functions

We consider pseudo-Boolean functions over bitstrings of length n. It is well known that for any pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, there exists a set $S := \{S_1, \ldots, S_k\}$ of k subsets $S_i \subseteq [n]$ and an associated set $W = \{w_i\}$ where $w_i \in \mathbb{R}$ such that

$$f(x) = \sum_{i=1}^{k} w_i \prod_{j \in S_i} x_j. \tag{4}$$

For example, we have $k = n$ and $S_i := \{i\}$ for linear functions, in which ONEMAX is the particular case where $w_i = 1$ for all $i \in [n]$. For LEADINGONES, we also get $k = n$ and $w_i = 1$ but $S_i := [i]$. In a classical optimisation problem, full access to (S, W) is guaranteed so that the exact value of $f(x)$ is returned in each evaluation of a solution x.

In an optimisation problem under partial information, the access to (S, W) is random and incomplete in each evaluation, e. g., restricted to only random subsets. Therefore, a random value $F_c(x)$ is returned instead of $f(x)$ in each evaluation of a solution x. Here c is the parameter of the source of randomness. There are many ways to define the randomisation, however given no prior information such as which combinatorial optimisation gives rise to a formulation of type (4), it is natural to consider the randomisation over the access to the subsets of S (or the weights of W). Therefore, in this paper we focus on the following model.

$$F_c(x) := \sum_{i=1}^{k} w_i R_i \prod_{j \in S_i} x_j \text{ with } R_i \sim \text{Bernoulli}(c). \tag{5}$$

Informally, each subset S_i has probability c of being taken into account in the evaluation. ONEMAX and LEADINGONES functions are considered as the typical examples. The corresponding optimisation problems are: ONEMAX(c), each 1-bit has only probability c to be added up in each fitness evaluation; LEADINGONES(c), each product $\prod_{j=1}^{i} x_j$ (associated to an $i \in [n]$) has only probability c of contributing to the fitness value.

For simplification, we restrict p_{sel} in our analysis to binary tournament selection. The algorithm is detailed in Algorithm 2. The runtime of the algorithm

is still defined in terms of the discovery of a true optimal solution. In other words, we focus on *when* the algorithm finds a true optimal solution for the first time and ignore *how* such an achievement is recognised. Nevertheless, the use of Theorem 8 will guarantee that when the runtime is reached, a large number of solutions in the population, more precisely the γ_0-portion, are indeed the true optimum.

Algorithm 2 EAs (2-Tournament, Partial Information)

1: Sample $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$, where $\mathcal{X} = \{0, 1\}^n$.
2: **for** $t = 0, 1, 2, \ldots$ until a termination condition is met **do**
3:     **for** $i = 1$ to λ **do**
4:         Sample two parents $x, y \sim \text{Unif}(P_t)$.
5:         $f_x := F_c(x)$ and $f_y := F_c(y)$.
6:         **if** $f_x > f_y$ **then**
7:             $z := x$
8:         **else if** $f_x < f_y$ **then**
9:             $z := y$
10:        **else**
11:            $z \sim \text{Unif}(\{x, y\})$
12:        **end if**
13:         Flip independently each bit position in z with probability χ/n.
14:         $P_{t+1}(i) := z$.
15:     **end for**
16: **end for**

Theorem 8 requires lower bounds for the cumulative selection probability function $\beta(\gamma)$, so that the necessary condition for the mutation rate χ/n can be established. The value of $\beta(\gamma)$ depends on the probability that the fitter of individual x and y is selected in lines 6–12 of Algorithm 2. Formally, for any x and y where $f(x) > f(y)$, we want to know a lower bound on the probability that the algorithm selects x in those lines. The corresponding event is denoted by $z = x$.

Lemma 17. *Let f be either the OneMax or the LeadingOnes function on $\{0, 1\}^n$. For any input $x, y \in \{0, 1\}^n$ with $f(x) > f(y)$ of the 2-tournament selection in Algorithm 2, we have $\Pr(z = x) \geq (1/2)(1 + c \Pr(X = Y))$ where X and Y are identical independent random variables following distribution $\text{Bin}(f(y), c)$.*

Proof. For OneMax(c), each bit of the $f(x)$ 1-bits of x has probability c being counted, so $F_c(x) \sim \text{Bin}(f(x), c)$ in Algorithm 2. The same argument holds for LeadingOnes(c), i.e., each block of consecutive 1-bits starting from the first position has probability c being contributed to the fitness value and there are $f(x)$ blocks in total, so $F_c(x) \sim \text{Bin}(f(x), c)$. Similarly, $F_c(y) \sim \text{Bin}(f(y), c)$ holds for bitstring y on the two functions.

We now remark that $f(x) = f(y) + (f(x) - f(y))$, we can decompose $F_c(x)$ and $F_c(y)$ further. Let X, Y and Δ be independent random variables such that $X \sim \text{Bin}(f(y), c)$, $Y \sim \text{Bin}(f(y), c)$ and $\Delta \sim \text{Bin}(f(x) - f(y), c)$, then we have $F_c(x) = X + \Delta$ and $F_c(y) = Y$.

By the law of total probability,

$$\Pr(z = x) = \Pr(X > Y) + \Pr(X = Y)\left(\Pr(\Delta > 0) + \frac{\Pr(\Delta = 0)}{2}\right)$$

$$= \Pr(X > Y) + \Pr(X = Y)\left(\Pr(\Delta \geq 0) - \frac{\Pr(\Delta = 0)}{2}\right)$$

$$= \Pr(X > Y) + \Pr(X = Y)\left(1 - \frac{(1-c)^{(f(x)-f(y))}}{2}\right)$$

$$\geq \Pr(X > Y) + \Pr(X = Y)\left(1 - \frac{1-c}{2}\right).$$

The inequality is due to $(1-c) \in (0,1)$ and $f(x) - f(y) \geq 1$. Because X and Y are identically distributed and independent, we have $\Pr(X < Y) = \Pr(X > Y)$. We also have the total probability $\Pr(X > Y) + \Pr(X = Y) + \Pr(X < Y) = 1$. The two results imply $\Pr(X > Y) = (1 - \Pr(X = Y))/2$. Put into the previous calculation of $\Pr(z = x)$,

$$\Pr(z = x) \geq \frac{1 - \Pr(X = Y)}{2} + \Pr(X = Y)\left(1 - \frac{1-c}{2}\right)$$

$$= \frac{1}{2}\left(1 + c\Pr(X = Y)\right). \qquad \square$$

Corollary 18. *Under the same assumption as in Lemma 17, we have*

$$\Pr(z = x) \geq \frac{1}{2}\left(1 + \frac{64c/81}{6\sqrt{(n-1)c(1-c)} + 1}\right).$$

Proof. From Lemma 17, we apply the result of Lemma 34 for $X, Y \sim \text{Bin}(n - 1, c)$ (in the worst case, we have $f(y) = n - 1$) and $d = 3$. $\qquad \square$

Lemma 19. *For any $\gamma \in (0,1)$, the cumulative selection probability of Algorithm 2 is at least*

$$\beta(\gamma) \geq \gamma\left(1 + \frac{(1-\gamma)64c/81}{6\sqrt{(n-1)c(1-c)} + 1}\right),$$

for $\textsc{OneMax}(c)$ and $\textsc{LeadingOnes}(c)$ functions.

Proof. Without loss of generality, for any inputs x and y of the tournament selection we assume that $f(x) \geq f(y)$. Recall that $\beta(\gamma)$ is the probability of picking an individual with fitness at least equal to the fitness of the γ-ranked individual, i.e., belonging to the upper γ-portion of the population. Therefore, it suffices if either x and y are picked from the portion, or only x is picked from the portion and then wins the tournament.

$$\beta(\gamma) \geq \gamma^2 + 2\gamma(1 - \gamma)\Pr(z = x)$$
$$= \gamma(1 + (1 - \gamma)(2\Pr(z = x) - 1)).$$

From Corollary 18, we have

$$2\Pr(z = x) - 1$$

22

$$\geq 2 \left(\frac{1}{2} \left(1 + \frac{64c/81}{6\sqrt{(n-1)c(1-c)+1}} \right) \right) - 1$$

$$= \frac{64c/81}{6\sqrt{(n-1)c(1-c)+1}}.$$

So

$$\beta(\gamma) \geq \gamma \left(1 + \frac{(1-\gamma)64c/81}{6\sqrt{(n-1)c(1-c)+1}} \right).$$

$\square$

Corollary 20. *For any $c \in (0,1)$ and any constant $\gamma_0 \in (0,1)$, then there exists constant $a \in (0,1)$ such that $\beta(\gamma) \geq \gamma(1+2\delta)$ for all $\gamma \in (0,\gamma_0]$ where $\delta = \min\{ac, a\sqrt{c/n}\}$.*

Proof. From Lemma 19, for all $\gamma \in (0,\gamma_0]$ we have

$$\beta(\gamma) \geq \gamma \left(1 + \frac{uc}{v\sqrt{nc}+1} \right),$$

where $u = (1-\gamma_0)64/81, v = 6$.

If $c \leq 1/n$, then $\sqrt{nc} \leq 1$ and $\dfrac{uc}{v\sqrt{nc}+1} \geq \left(\dfrac{u}{v+1} \right) c$.

If $c > 1/n$, then $\sqrt{nc} > 1$ and

$$\frac{uc}{v\sqrt{nc}+1} > \frac{uc}{v\sqrt{nc}+\sqrt{nc}} = \left(\frac{u}{v+1} \right) \sqrt{\frac{c}{n}}.$$

The statement now follows by choosing $a = u/(2(v+1))$. $\square$

We now rigorously prove that for $1/c \in \text{poly}(n)$, a population-based EA can optimise $\textsc{OneMax}(c)$ in expected polynomial time. On the other hand, the (1+1) EA needs exponential time in expectation for any constant $c < 1$.

Theorem 21. *Given $c \in (0,1)$ such that $1/c \in \text{poly}(n)$, there exist constants a and b such that Algorithm 2 with $\chi = \delta/3$ and $\lambda = b \ln n/\delta^2$ where $\delta = \min\{ac, a\sqrt{c/n}\}$ optimises $\textsc{OneMax}(c)$ in expected time*

$$\begin{cases} O\left(\dfrac{n \ln n}{c^7} \right) & \text{if } c \leq 1/n \text{ and} \\ O\left(\dfrac{n^{9/2} \ln n}{c^{7/2}} \right) & \text{if } c > 1/n. \end{cases}$$

Proof. Corollary 20 and $1/c \in \text{poly}(n)$ imply $1/\delta \in \text{poly}(n)$, $\delta \in (0,1)$ and $\chi < 1/3$. We then use the partition $A_j := \{x \in \{0,1\}^n \mid |x|_1 = j\}$ to analyse the runtime. The probability of improving a solution at fitness level j by mutation is lower bounded by the probability that a single 0-bit is flipped and not the other bits,

$$(n-j) \left(\frac{\chi}{n} \right) \left(1 - \frac{\chi}{n} \right)^{n-1} > \left(1 - \frac{j}{n} \right) \left(\chi \left(1 - \frac{\chi}{n} \right)^n \right).$$

It follows from Lemma 29 and $\chi < 1/3$ that $(1 - \chi/n)^n \geq (1 - \chi) > 2/3$. Therefore, $\chi(1 - \chi/n)^n > (\delta/3)(2/3) = 2\delta/9$, and it suffices to choose the parameters $s_j := (1 - j/n)(\delta/9)$ and $s_* := \delta/(9n)$ so that (C1) is satisfied. In addition, $(1 - \chi/n)^n$ is the probability of not flipping any bit in the mutation, hence picking $p_0 = 1 - \chi$ satisfies (C2).

It now follows from Corollary 20 that for all $\gamma \in (0, \gamma_0]$

$$
\begin{aligned}
\beta(\gamma)p_0 &\geq \gamma(1 + 2\delta)(1 - \chi) = \gamma(1 + 2\delta)(1 - \delta/3) \\
&= \gamma(1 - \delta/3 + 2\delta - 2\delta^2/3) \geq \gamma(1 - \delta/3 + 2\delta - 2\delta/3) \\
&= \gamma(1 + \delta).
\end{aligned}
$$

Therefore, (C3) is satisfied with the given value of δ. Because $1/\delta \in \mathrm{poly}(n)$ we have $1/s_* \in \mathrm{poly}(n)$ and by Corollary 9, there exists a constant b such that condition (C4) is satisfied for $\lambda = (b/\delta^2) \ln m$. All conditions are satisfied, and by Corollary 10, the expected optimisation time is

$$
O\left(\frac{1}{\delta^5} \left(m\lambda(1 + \ln(1 + \delta^4 \lambda)) + \sum_{j=1}^{m} \frac{1}{s_j} \right) \right)
$$

$$
= O\left(\frac{1}{\delta^5} \left(\left(\frac{n \ln n}{\delta^2} \right)(1 + \ln(1 + \delta^2 \ln n)) + \frac{n}{\delta} \sum_{j=1}^{n} \frac{1}{j} \right) \right).
$$

By the definition of δ, it follows that $\delta = O(1/\sqrt{n})$, so $1 + \ln(1 + \delta^2 \ln n) = O(1)$. Furthermore, $\frac{n}{\delta} \sum_{j=1}^{n} 1/j = O(n \ln n/\delta)$ is dominated by the left term $n \ln n/\delta^2$. Hence, the optimisation time is $\mathbf{E}[T] = O(n \ln n/\delta^7)$. The theorem follows by noting that $\delta = ac$ if $c \leq 1/n$, and $\delta = a\sqrt{c/n}$ otherwise. $\qquad\square$

We now consider the partial evaluation for LEADINGONES, i.e., the optimisation problem LEADINGONES(c). The following result holds.

Theorem 22. *For any $c \in (0, 1)$ where $1/c \in \mathrm{poly}(n)$, there exist constants a and b such that Algorithm 2 with $\chi = \delta/3$ and $\lambda = b \ln n/\delta^2$ where $\delta = \min\{ac, a\sqrt{c/n}\}$ optimises LEADINGONES(c) in expected time*

$$
\begin{cases}
O\left(\dfrac{n \ln n}{c^7} \right) & \text{if } c \leq 1/n \text{ and} \\
O\left(\dfrac{n^{9/2} \ln n}{c^{7/2}} + \dfrac{n^5}{c^3} \right) & \text{if } c > 1/n.
\end{cases}
$$

Proof. Conditions (C2)-(C4) are shown exactly as in the proof of Theorem 21. For condition (C1), we use the canonical partition

$$
A_j := \{x \in \{0, 1\}^n \mid \text{LEADINGONES}(x) = j\}.
$$

The probability of improving a solution at fitness level j by mutation is lower bounded by the probability that the leftmost 0-bit is flipped, and no other bits are flipped,

$$
\left(\frac{\chi}{n} \right) \left(1 - \frac{\chi}{n} \right)^{n-1} > \left(\frac{1}{n} \right)(\chi(1 - \chi)) \geq \frac{2\delta}{9n}.
$$

We therefore choose $s_j = s_* = \delta/(9n)$ and the runtime is

$$O\left(\frac{1}{\delta^5}\left(\left(\frac{n\ln n}{\delta^2}\right)(1 + \ln(1 + \delta^2 \ln n)) + \frac{n^2}{\delta}\right)\right)$$
$$= O\left(\frac{n\ln n}{\delta^7} + \frac{n^2}{\delta^6}\right).$$

The result now follows by noting that $\delta = ac$ if $c \leq 1/n$, and $\delta = a\sqrt{c/n}$ otherwise. □

We have shown that non-elitist EAs with populations of polynomial sizes and without elitism can optimise $\textsc{OneMax}(c)$ and $\textsc{LeadingOnes}(c)$ in expected polynomial runtime, precisely in terms of partial evaluation calls, for small c, i.e., $1/c \in \text{poly}(n)$. Particularly, for any constant $c \in (0, 1)$, the non-elitist EAs can optimise $\textsc{LeadingOnes}(c)$ in $O(n^5)$ and $\textsc{OneMax}(c)$ in $O(n^{9/2}\ln n)$. We now show that the classical $(1 + 1)$ EA, which is summarised in Algorithm 3, already requires exponential runtime on $\textsc{OneMax}(c)$ for any constant $c < 1$.

Algorithm 3 $(1+1)$ EA (Partial Information)

1: Sample $x_0 \sim \text{Unif}(\{0, 1\}^n)$.
2: **for** $t = 0, 1, 2, \ldots$ until a termination condition is met **do**
3:    $x' := x_t$.
4:    Flip independently each bit position in x' with probability $1/n$.
5:    **if** $F_c(x') \geq F_c(x_t)$ **then**
6:       $x_{t+1} := x'$.
7:    **else**
8:       $x_{t+1} := x_t$.
9:    **end if**
10: **end for**

Theorem 23. *For any constant $c \in (0, 1)$, the expected optimisation time of $(1+1)$ EA on $\textsc{OneMax}(c)$ is $e^{\Omega(n)}$.*

Proof. We use Theorem 7 with X_t as the Hamming distance from 0^n (all-zero bitstring) to x_t, so $X_t := H(0^n, x_t)$ and $b(n) := n$. The use of $F_c(x)$ implies that the algorithm can accept degraded solutions with less 1-bits than the current solution. This typically happens when all bit positions of x' are not flipped except a 1-bit then the evaluation of x' does not recognise the change, such an event is denoted by $\mathcal{E}$. To analyse $\Pr(\mathcal{E})$, let us denote the event that a specific bit position i being flipped and not the others, and $F_c(x')$ does not evaluate position i, by $\mathcal{E}_i$.

$$\Pr(\mathcal{E}_i) = (1 - c)\left(\frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1-c}{ne}.$$

Assuming that there are currently j 1-bits in x_t, events $\mathcal{E}_i$ with respect to those bits are non-overlapping. In addition, under the condition that one of those events $\mathcal{E}_i$ happens (so with probability $j\Pr(\mathcal{E}_i)$), we use X to denote the number of the remaining 1-bits in x_t being evaluated, and Y the number of evaluated 1-bits in x'. Indeed, X and Y are identical and independent variables

following the same binomial distribution $\mathrm{Bin}(f(x), c)$, so $\Pr(Y \geq X) \geq 1/2$. We get,

$$\Pr(\mathcal{E}) \geq j \cdot \frac{1-c}{ne} \cdot \Pr(Y \geq X) \geq \frac{j(1-c)}{2ne}.$$

The drift is then

$$\mathbf{E}\left[X_{t+1} - X_t \mid \mathscr{F}_t\right] \leq -1 \cdot \frac{j(1-c)}{2ne} + (n-j) \cdot \frac{1}{n} = 1 - \frac{j}{n}\left(\frac{2e+1-c}{2e}\right).$$

We have a negative drift when

$$j \geq n\left(\frac{2e}{2e+1-c}\right) = n\left(1 - \frac{1-c}{2e+1-c}\right).$$

From that, for any constant $\varepsilon \in (0, 1-c)$, there exists $a(n) := n\left(1 - \frac{1-c-\varepsilon}{2e+1-c}\right)$ so that

$$\mathbf{E}\left[X_{t+1} - X_t \; ; \; X_t > a(n) \mid \mathscr{F}_t\right]$$
$$\leq 1 - \left(1 - \frac{1-c-\varepsilon}{2e+1-c}\right)\left(\frac{2e+1-c}{2e}\right) = -\frac{\varepsilon}{2e}.$$

We have here $b(n) - a(n) = \Omega(n)$ and the condition (L1) of Theorem 7 is satisfied. Condition (L2) holds trivially due to the property of Hamming distance $|H(x_t, 0^n) - H(x_{t+1}, 0^n)| \leq H(x_t, x_{t+1}) \leq H(x_t, x')$. So

$$\left(|X_t - X_{t+1}| \mid \mathscr{F}_t\right) \preceq Z \text{ where } Z := H(x_t, x').$$

Then $Z \sim \mathrm{Bin}(n, 1/n)$ and $\mathbf{E}\left[e^{\lambda Z}\right] \leq e$ for $\lambda = \ln 2$. Note also that if each bit position of x_0 is initialised uniformly at random from $\{0, 1\}$, then X_0 will be highly concentrated near $n/2 \leq a(n)$. It then follows from Theorem 7 that $(1+1)$ EA requires expected exponential runtime to optimise $\textsc{OneMax}(c)$. $\quad\square$

5.2 Partial evaluation of populations

In the previous section, we have seen that EAs with populations do not need complete but only little information about the fitness of a solution in each evaluation to discover a true optimal solution in expected polynomial runtime. One might wonder if the same could be true if in each generation, only few solutions of the population make use of their fitnesses while the rest reproduces randomly. Here we talk about the information contained within a population, in contrast to the one contained within a solution.

We first give the motivation for how such a lack of information about the population can arise. In real-life applications, the evaluation of solutions can be both time-consuming (e.g., requiring extensive simulation) and inaccurate. We associate a probability $1 - r$ with the event that the quality of a solution is not available to the algorithm. We model such a scenario in Algorithm 4 by assuming that the outcome of any comparison between two individuals is only available to the algorithm with probability r.

Unlike Algorithm 2, the algorithm in this section uses complete information about the problem to evaluate solutions. However, the evaluation of individuals

---
Algorithm 4 EAs (2-Tournament, Partially Eval. Pop.)

1: Sample $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$, where $\mathcal{X} = \{0,1\}^n$.
2: **for** $t = 0, 1, 2, \ldots$ until a termination condition is met **do**
3:     **for** $i = 1$ to λ **do**
4:         Sample two parents $x, y \sim \text{Unif}(P_t)$.
5:         $z := \begin{cases} \text{argmax}\{f(x), f(y)\} & \text{with probability } r \\ x & \text{otherwise} \end{cases}$
6:         Flip independently each bit position in z with probability χ/n.
7:         $P_{t+1}(i) := z$
8:     **end for**
9: **end for**
---

in the parent populations is not systematic but random. Two individuals x and y are sampled uniformly at random from the population as parents (line 4). With probability r, the individuals are evaluated and the fitter individual is selected (line 5). Otherwise, a parent x is arbitrarily chosen independently of the fitness value. Like before, efficient implementation is possible but that does not change the outcome of our analysis. The analysis is more straightforward compared to the previous section.

Lemma 24. *For any $r \in (0,1)$ and constant $\gamma_0 \in (0,1)$, there exists a constant $a \in (0,1)$ such that Algorithm 4 satisfies $\beta(\gamma) \geq \gamma(1 + 2\delta)$ for all $\gamma \in (0, \gamma_0)$, where $\delta = ar$.*

Proof. An individual among the best γ-portion of the population is selected if either (i) a uniformly chosen individual is chosen as parent (with probability $1 - r$), and this individual belongs to the best γ-portion, or (ii) the tournament selection happens (with probability r), and at least one of the selected parents belongs to the γ-portion.

$$\beta(\gamma) \geq \gamma(1 - r) + r(1 - (1 - \gamma)^2)$$
$$= \gamma(1 + (1 - \gamma)r).$$

So for all $\gamma \in (0, \gamma_0]$, $\beta(\gamma) \geq \gamma(1 + (1 - \gamma_0)r)$ and the statement follows by choosing $a = (1 - \gamma_0)/2$. $\qquad \square$

The lemma can be generalised to k-tournament selection. However, too larg tournament sizes may lead to overheads in the average number of evaluations per generation. This number is a random variable following the binomial distribution $\text{Bin}(k\lambda, r)$. Hence, we focus on $k = 2$. Similarly to the previous section, we allow small r, i.e., $1/r \in \text{poly}(n)$. The following theorem holds for the runtime of $\text{OneMax}(c)$.

Theorem 25. *There exist constants a and b such that Algorithm 4 optimises* OneMax *if $r \in (0,1)$ and $1/r \in \text{poly}(n)$, $\chi = \delta/3$ and $\lambda = b \ln n / \delta^2$ where $\delta = ar$ in $O(n \ln n \ln \ln n)/r^7$.*

Proof. We use the same partition from Theorem 21 and the proof idea is similar. We first remark that by Lemma 24 and from $1/r \in \text{poly}(n)$ we have $1/\delta \in$

poly(n), $\delta \in (0,1)$ and $\chi < 1/3$. The probability of improving a solution at fitness level j by mutation is at least

$$(n-j)\left(\frac{\chi}{n}\right)\left(1-\frac{\chi}{n}\right)^{n-1} > \left(1-\frac{j}{n}\right)\left(\frac{2\delta}{9}\right).$$

So we can pick $s_j := (1-j/n)(\delta/9)$, $s_* := \delta/(9n)$ and $p_0 = 1-\chi$ so that (C1) and (C2) are satisfied.

It now follows from Lemma 24 that for all $\gamma \in (0, \gamma_0]$

$$\beta(\gamma)p_0 \geq \gamma(1+2\delta)(1-\chi) = \gamma(1+2\delta)(1-\delta/3) \geq \gamma(1+\delta).$$

Therefore, (C3) is satisfied with the given value of δ. Because $1/\delta \in \mathrm{poly}(n)$, then $1/s_* \in \mathrm{poly}(n)$ and by Corollary 9, there exists a constant b such that condition (C4) is satisfied with $\lambda = (b/\delta^2)\ln m$. All conditions are satisfied, and by Corollary 10 the expected optimisation time is

$$O\left(\frac{1}{\delta^5}\left(\left(\frac{n\ln n}{\delta^2}\right)(1+\ln(1+\delta^2\ln n)) + \frac{n}{\delta}\sum_{j=1}^{n}\frac{1}{j}\right)\right)$$

$$= O\left(\frac{n\ln n(1+\ln(1+\delta^2\ln n))}{\delta^7} + \frac{n}{\delta^6}\sum_{j=1}^{n}\frac{1}{j}\right)$$

$$= O\left(\frac{n\ln n\ln\ln n}{\delta^7}\right).$$

The result now follows by noting that $\delta = ar$. □

Remark that the theorem overestimates the optimisation time for small δ, e.g., if $\delta^2\ln n = O(1)$ then the term $\ln\ln n$ can be ignored. In addition, when the parameter r is small, the fitness function is evaluated only a few times per generation. More precisely, the fitness function is evaluated $2N$ times, where $N \sim \mathrm{Bin}(\lambda, r)$. The proof of Theorem 8 pessimistically assumes that the fitness function is evaluated 2λ times per generation. A more sophisticated analysis which takes this into account may lead to tighter bounds than those above.

We can also use Algorithm 4 to optimise LEADINGONES. The following theorem holds for its runtime.

Theorem 26. *There exist constants a and b such that Algorithm 4 optimise* LEADINGONES *under condition $r \in (0,1)$ and $1/r \in \mathrm{poly}(n)$, with $\chi = \delta/3$ and $\lambda = b\ln n/\delta^2$ where $\delta = ar$ in $O(n\ln n/r^7 + n^2/r^6)$.*

Proof. We use the same approach as the proof of Theorem 22, including the partition. So we have the same $s_j = s_* = \delta/(9n)$, and the expected runtime for any $r \in (0,1)$ with $1/r \in \mathrm{poly}(n)$ is

$$O\left(\frac{n\ln n}{\delta^7} + \frac{n^2}{\delta^6}\right) = O\left(\frac{n\ln n}{r^7} + \frac{n^2}{r^6}\right).$$

□

Our results have shown that EAs do not need the fitness values of all individuals in their populations to efficiently optimise a function. Another interpretation is that strong competition between all individuals before reproduction is not necessary for efficient evolution. Note that non-elitism can be considered as a way of reducing the competitiveness in populations. These observations have many analogies with evolution in biology.

6 Conclusions

We have presented a new version of the main theorem introduced in [24]. The new theorem offers a general tool to analyse the expected runtime of randomised search heuristics with non-elitist populations on many optimisation problems. Sharing similarities with the classical fitness-level technique, our method is straightforward and easy-to-use. The structure of the new proof was simplified and a more detailed analysis of the population dynamics has been provided. This leads to significantly improved upper bounds for the expected runtimes of EAs on many pseudo-Boolean functions. Furthermore, the new fitness-level technique makes the relationship between the selective pressure and the runtime of the algorithm explicit. Surprisingly, a weak selective pressure is sufficient to optimise many functions in expected polynomial time. This observation has interesting consequences such as the proofs of the efficiency of population-based EAs in solving optimisation problems under uncertainty.

References

[1] A. Auger, A. Auger, and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments.* World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2011.

[2] T. Chen, J. He, G. Sun, G. Chen, and X. Yao. A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39(5):1092–1106, 2009.

[3] R. Chiong, T. Weise, and Z. Michalewicz, editors. *Variants of Evolutionary Algorithms for Real-World Applications.* Berlin/Heidelberg: Springer-Verlag, 2012.

[4] D. Corus, D.-C. Dang, A. V. Eremeev, and P. K. Lehre. Level-based analysis of genetic algorithms and other search processes. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN'14, pages 912–921, Ljubljana, Slovenia, 2014. Springer International Publishing.

[5] D.-C. Dang and P. K. Lehre. Evolution under partial information. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'14, pages 1359–1366, 2014.

[6] D.-C. Dang and P. K. Lehre. Refined upper bounds on the expected runtime of non-elitist populations from fitness-levels. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'14, pages 1367–1374, New York, NY, USA, 2014. ACM.

[7] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[8] S. Droste. Analysis of the (1+1) EA for a dynamically bitwise changing OneMax. In *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation*, GECCO'03, pages 909–921, Berlin, Heidelberg, 2003. Springer-Verlag.

[9] S. Droste. Analysis of the (1+1) EA for a noisy OneMax. In *Proceedings of the 2004 International Conference on Genetic and Evolutionary Computation*, GECCO'04, pages 1088–1099, Berlin, Heidelberg, 2004. Springer-Verlag.

[10] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, NY, USA, 2009.

[11] C. Gießen and T. Kötzing. Robustness of populations in stochastic environments. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'14, pages 1383–1390, New York, NY, USA, 2014. ACM.

[12] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Proceedings of Foundations of Genetic Algorithms*, FOGA I, pages 69–93. Morgan Kaufmann, 1991.

[13] B. Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14(3):502–525, 1982.

[14] E. Happ, D. Johannsen, C. Klein, and F. Neumann. Rigorous analyses of fitness-proportional selection for optimizing linear functions. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'08, pages 953–960, 2008.

[15] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[16] J. He and X. Yao. From an individual to a population: an analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495–511, 2002.

[17] T. Jansen. *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Natural Computing Series. Springer, 2013.

[18] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[19] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[20] O. Kallenberg. *Foundations of Modern Probability*. Springer-Verlag, New York, 2002.

[21] T. Kötzing, F. Neumann, D. Sudholt, and M. Wagner. Simple max-min ant systems and the optimization of linear pseudo-boolean functions. In *Proceedings of Foundations of Genetic Algorithms*, FOGA XI, pages 209–218, 2011.

[22] J. Lässig and D. Sudholt. General scheme for analyzing running times of parallel evolutionary algorithms. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN'10, pages 234–243, 2010.

[23] P. K. Lehre. Negative drift in populations. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, PPSN'10, pages 244–253, Berlin, Heidelberg, 2010. Springer-Verlag.

[24] P. K. Lehre. Fitness-levels for non-elitist populations. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'11, pages 2075–2082, New York, NY, USA, 2011. ACM.

[25] P. K. Lehre. Drift analysis. In *Proceedings of the Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO'12, pages 1239–1258, New York, NY, USA, 2012. ACM.

[26] P. K. Lehre and X. Yao. On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 16(2):225–241, 2012.

[27] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4:113–131, 1996.

[28] D. S. Mitrinović. *Elementary Inequalities*. P. Noordhoff Ltd, Groningen, 1964.

[29] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[30] F. Neumann, P. S. Oliveto, and C. Witt. Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, GECCO'09, pages 835–842, 2009.

[31] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[32] J. E. Rowe and D. Sudholt. The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38, 2014.

[33] D. Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17(3):418–435, 2013.

[34] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In *Evolutionary Optimization*, volume 48, pages 349–369. Springer US, 2002.

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

[36] T. Yu, L. Davis, C. Baydar, and R. Roy, editors. *Evolutionary Computation in Practice*, volume 88 of *Studies in Computational Intelligence*. Springer, 2008.

Appendix

Lemma 27 (Chernoff's Inequality, see [10]). *If* $X = \sum_{i=1}^{m} X_i$, *where* $X_i \in \{0, 1\}, i \in [m]$, *are independent random variables, then* $\Pr(X \leq (1 - \varepsilon)\mathbf{E}[X]) \leq \exp\left(-\varepsilon^2 \mathbf{E}[X]/2\right)$ *for any* $\varepsilon \in [0, 1]$.

Lemma 28 (Chebyshev's Inequality, see [29]). *For any random variable* X *with finite expected value* μ *and finite non-zero variance* σ^2, *it holds that* $\Pr(|X - \mu| \geq d\sigma) \leq 1/d^2$ *for any* $d > 0$.

Lemma 29 (Bernoulli's inequality, see [28]). *For any integer* $n \geq 0$ *and any real number* $x \geq -1$, *it holds that* $(1 + x)^n \geq 1 + nx$.

Lemma 30 (Jensen's inequality, see [28]). *For any function* $f(x)$ *convex in* $[\alpha, \beta]$ *and* $a_i \in [\alpha, \beta]$ *with* $i \in [n]$,

$$f\left(\frac{1}{n}\sum_{i=1}^{n} a_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} f(a_i)$$

Lemma 31. *For* $n \in \mathbb{N}$ *and* $x \geq 0$, *we have* $1 - (1 - x)^n \geq 1 - e^{-xn} \geq \frac{xn}{1+xn}$.

Proof. From $e^x \geq x + 1$, it follows that $1 - (1 - x)^n \geq 1 - e^{-nx} \geq 1 - (1 + x)^{-n} = 1 - 1/\left(1 + \sum_{k=1}^{n}\binom{n}{k}x^k\right)$. Note that $x^k \geq 0$ for all $x \geq 0$, thus any partial sum of $\sum_{k=1}^{n}\binom{n}{k}x^k$ provides a valid lower bound. The result is obtained for the single term $k = 1$. □

Lemma 32. *For all* $x \in \mathbb{R}$, $e^{2x} \geq (x + 1)e^x > xe^x$ *and for* $x > 0$, $e^{2x}/x > e^x$.

Proof. Multiplying e^x to $e^x - x \geq 1 > 0$ and adding xe^x to both sides give the first result, then dividing them by $x > 0$ provides the second one. □

Lemma 33. *For all* $x \geq 0$, $x \geq \ln(1 + x) \geq x(1 - x/2)$.

Proof. For each inequality, it suffices to first show that the gap between the (supposedly) larger side and the (supposedly) smaller one is non-decreasing in x, e.g., by looking at the derivative, then show that the initial gap is 0 at $x = 0$. □

Lemma 34. *Let* X *and* Y *be identically distributed independent random variables with integer support, finite expected value* μ *and finite non-zero variance* σ^2, *it holds that*

$$\Pr(X = Y) \geq \frac{(1 - 1/d^2)^2}{2d\sigma + 1} \text{ for any } d \geq 1$$

Proof. For any $k, \ell \in \mathbb{R}$ with $\lceil k \rceil \leq \lfloor \ell \rfloor$, we have

$$\Pr(X = Y) = \sum_{i \in \mathbb{Z}} \Pr(X = i)^2 \geq \sum_{i = \lceil k \rceil}^{\lfloor \ell \rfloor} \Pr(X = i)^2$$

$$\geq \frac{\left(\sum_{i = \lceil k \rceil}^{\lfloor \ell \rfloor} \Pr(X = i) \right)^2}{\lfloor \ell \rfloor - \lceil k \rceil + 1} \geq \frac{\Pr(k < X < \ell)^2}{\lfloor \ell \rfloor - \lceil k \rceil + 1}$$

The second last equality is due to Lemma 30 with the convex function $f(x) = x^2$. It suffices to pick $k = \mu - d\sigma$ and $\ell = \mu + d\sigma$, so that by Lemma 28, we get

$$\Pr(X = Y) \geq \frac{\Pr(|X - \mu| < d\sigma)^2}{2d\lfloor \sigma \rfloor + 1}$$

$$= \frac{(1 - \Pr(|X - \mu| \geq d\sigma))^2}{2d\lfloor \sigma \rfloor + 1} \geq \frac{(1 - 1/d^2)^2}{2d\sigma + 1}. \qquad \square$$