Simplified Runtime Analysis of Estimation of Distribution Algorithms

Duc-Cuong Dang
ASAP Research Group
School of Computer Science
University of Nottingham
duc-cuong.dang
@nottingham.ac.uk

Per Kristian Lehre
ASAP Research Group
School of Computer Science
University of Nottingham
perkristian.lehre
@nottingham.ac.uk

ABSTRACT

Estimation of distribution algorithms (EDA) are stochastic search methods that look for optimal solutions by learning and sampling from probabilistic models. Despite their popularity, there are only few rigorous theoretical analyses of their performance. Even for the simplest EDAs, such as the Univariate Marginal Distribution Algorithm (UMDA) which assumes independence between decision variables, there are only a handful of results about its runtime, and results for simple functions such as OneMax are still missing.

In this paper, we show that the recently developed *level-based theorem* for non-elitist populations is directly applicable to runtime analysis of EDAs. To demonstrate this approach, we derive easily upper bounds on the expected runtime of the UMDA.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

Keywords

Runtime Analysis, Level-based Analysis, Estimation of Distribution Algorithm

1. INTRODUCTION

Estimation of Distribution Algorithm (EDA) [15] is a relatively new paradigm in Evolutionary Computation. Unlike traditional approaches of Evolutionary Algorithms (EAs) that work with explicit evolutionary/genetic operators such as mutation, recombination and selection, an EDA will attempt to build probabilistic models for solution sampling so that the probability of creating an optimal solution through the sampling is high. The algorithm often starts with a specific probabilistic model, which is gradually updated through selected solutions of intermediate samplings.

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ISBN 978-1-4503-3472-3/15/07...$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754814

Over the recent years, many variants of EDAs have been proposed, along with theoretical investigations on their convergence and scalability [11, 13, 18, 20, 21, 22, 23]. However, rigorous runtime analysis results for this particular class of algorithms on discrete domain are still sparse. The first analysis of this kind was conducted in [8] for the compact Genetic Algorithm (cGA) [12] on linear functions. cGA is one of the simplest EDAs which assume no dependence between decision variables. On bitstrings of length n, cGA uses a Poisson binomial distribution parametrised by a probability vector $(p_i)_{i\in[n]}$ as the probabilistic model. In each generation, the cGA generates two solutions and uses the best one to update the probability vector via a learning rate parameter $1/K$. Without any proper margin for each p_i, cGA can eventually end up with a distribution that can never generate an optimal solution, i.e., a premature convergence. Therefore, the runtimes for cGA on OneMax and BinVal functions which were reported in [8] are conditioned on the event that this does not occur. Very recently, the benefits of cGA in noisy environment were proven using runtime arguments in [10].

Another simple EDA is the Univariate Marginal Distribution Algorithm (UMDA) which was analysed in a series of papers [1, 2, 3, 4]. UMDA was proposed in [17], assuming as the cGA independence between decision variables. However, the UMDA samples a real population of solutions and uses the selected solutions to create a new vector $(p_i)_{i\in[n]}$ in each generation. The initial result of [2] was provided for LeadingOnes and a harder function known as TrapLeadingOnes under the so-called "no-random-error" assumption and with a sufficiently large population. The assumption was lifted due to the technique presented in [4]. Nevertheless, the analysis assumes an unrealistically large population size, leading in overall to a too high bound on the expected runtime. Note also that there are two versions of the algorithm based on whether or not margins are imposed to p_i, the difference between the two in terms on time complexity for various functions are discussed in [3]. More interestingly, [1] showed that UMDA without margins beats the $(1+1)$ EA on a particular function called SubString. However, it is not recommended to use UMDA without margins in practice, as the algorithm can always end up with a premature convergence.

To summarise, there are very few rigorous runtime results for cGA and UMDA, the two simplest EDAs. The analytical techniques used in those analyses are often complex, e.g., relying on the machinery of Markov chain theory, and the results only hold under extra assumptions. Even more sur-

prisingly, we could not find any results for UMDA on the OneMax function, a classic benchmark function used when initiating runtime analysis of new algorithms. This is very contrast to our current understanding of EAs, of which over the past decades [19], many rigorous runtime results have been shown for various settings, from the single-individual approach as the (1+1) EA or RLS to population-based ones like $(\mu + 1)$ EA, or even Genetic Algorithms (GA) [5]. One contributing factor behind these advances for EAs may have been the development of appropriate analytical techniques such as *drift analysis* or *fitness-level* method (e. g., see [14] for an overview).

Motivated by the above facts, this paper derives the runtime of UMDA with margins and truncation selection on the two standard functions OneMax and LeadingOnes by a straightforward argument. We make no extra assumptions about the optimisation process. Our runtime results will be stated in terms of number of solution evaluations. This is made possible by the recently developed level-based method [5] which describes the search process at a high level of abstraction, akin to an estimation of distribution algorithm. The method is applicable to any search process which samples new solutions independently from a distribution that only depends on the current generation.

The remainder of the paper is organised as follows. In the next section, we give the formal description of the UMDA. Next, we describe the two methods, *level-based* analysis, and Feige's bound which are our main analytical tools. Then the results for LeadingOnes and OneMax are presented with straightforward proofs. Finally some conclusions are drawn.

2. ALGORITHM

We consider the optimisation of functions $f : \mathcal{X} \to \mathbb{R}$ over the Boolean hypercube $\mathcal{X} = \{0, 1\}^n$. If $P \in \mathcal{X}^\lambda$ is a population of λ solutions, let $P(k, i)$ denote the value in the i-th bit position of the k-th solution in P. The Univariate Marginal Distribution Algorithm (UMDA) with (μ, λ)-truncation selection is defined in Algorithm 1. As a common practice in runtime analysis community, algorithms hereby are written without specifying the stopping condition. That is because we will mainly focus on computing the first point in time that an optimal solution is conceived.

Algorithm 1 UMDA

1: Initialise the vector $p_0 := (1/2, \ldots, 1/2)$.
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: **for** $k = 1$ to λ **do**
4: Sample the k-th individual $P_t(k, \cdot)$ according to

$$P_t(k, i) \sim \text{Bernoulli}(p_t(i)) \text{ for all } i \in [n].$$

5: **end for**
6: Sort the population P_t according to f.
7: Calculate a new vector p_{t+1} from P_t according to

$$p_{t+1}(i) := \begin{cases} \frac{m}{\mu} & \text{if } X_i < m \\ \frac{X_i}{\mu} & \text{if } m \le X_i \le \mu - m \\ 1 - \frac{m}{\mu} & \text{if } \mu - m < X_i, \end{cases}$$

 for all $i \in [n]$, where $X_i := \sum_{k=1}^{\mu} P_t(k, i)$.
8: **end for**

The algorithm has three parameters, the parent population size μ, the offspring population size λ, and a parameter $m < \mu$ controlling the size of the margins. It is necessary to set $m > 0$ to prevent a premature convergence, e. g., without this margin a $p_t(i)$ can goes to a non-optimal fixation, this prevents further exploration and causes an infinite runtime. Based on insights about optimal mutation rates in the (1+1) EA, we will use the parameter setting $m = \mu/n$ in the rest of this paper.

3. TECHNIQUES

3.1 Level-based Analysis

The level-based theorem for analysis of population-based algorithms is a general tool that provides upper bounds on the expected runtime of a wide class of meta-heuristics applied to various optimisation scenarios. For example, it has been used to study genetic algorithms with crossover [5], and evolutionary algorithms subject to uncertain optimisation, such as noise or missing information [6, 7].

The theorem applies to any search process that can be represented in the algorithmic scheme shown in Algorithm 2. It is assumed that the algorithm at any time $t \ge 0$ has a population P_t of λ individuals in some search space $\mathcal{X}$. The algorithm generates the next population P_{t+1} by sampling independently λ times from a probability distribution $D(P_t)$ which is parameterised by the current population P_t. As such, the algorithmic scheme can be considered as a generic estimation of distribution algorithm (EDA) which evolves a sequence of probability distributions $(D_t)_{t \ge 0}$ where $D_t = D(P_t)$. In particular, it is immediately clear that the UMDA in Algorithm 1 is a special case of this algorithmic scheme. The probability distribution $D(P_t)$ is computed in steps 6-7, and is defined for any search point $x \in \{0, 1\}^n$ by

$$\Pr(Y = x) = \prod_{j=1}^{n} p_t(j)^{x_j} (1 - p_t(j))^{1 - x_j}.$$

Note that in the compact genetic algorithm (cGA) and some other randomised search heuristics such as ant colony optimisation (ACO), the sampling distribution D_t does not only depend on the current population, but also on additional information, such as pheromone values. The level-based theorem does not apply to such algorithms.

Algorithm 2 Population Algorithm with Sampling

Require:
 Finite state space $\mathcal{X}$,
 Configurable probability distribution D over $\mathcal{X}$
1: $P_0 \sim \text{Unif}(\mathcal{X}^\lambda)$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: **for** $k = 1$ to λ **do**
4: $P_{t+1}(k) \sim D(P_t)$.
5: **end for**
6: **end for**

The level-based theorem assumes that the search space has been partitioned into a sequence of levels $A_1, \ldots, A_{m+1}$, such that the last level contains all the search points that are considered optimal solutions to the problem. If it can be shown that the distribution $D(P_t)$ is biased towards higher levels in a way that will be made precise below, the theorem provides upper bounds on the expected time to obtain

solutions in the final level A_{m+1}. To simplify the writing, we use A_j^+ to denote all levels after A_j in the sequence, or $A_j^+ := \cup_{i=j+1}^{m+1} A_i$.

THEOREM 1. *Given a partition $(A_1, \ldots, A_{m+1})$ of $\mathcal{X}$, define $T := \min\{t\lambda \mid |P_t \cap A_{m+1}| > 0\}$ to be the first point in time that elements of A_{m+1} appear in P_t of Algorithm 2. If there exist parameters $z_1, \ldots, z_m, z_* \in (0,1]$, $\delta > 0$, a constant $\gamma_0 \in (0,1)$ and a function $z_0 : (0, \gamma_0) \to \mathbb{R}$ such that for all $j \in [m]$, $P \in \mathcal{X}^\lambda$, $y \sim D(P)$ and $\gamma \in (0, \gamma_0)$ we have*

(C1) $\Pr\left(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0\lambda\right) \geq z_j \geq z_*$

(C2) $\Pr\left(y \in A_j^+ \mid |P \cap A_{j-1}^+| \geq \gamma_0\lambda, |P \cap A_j^+| \geq \gamma\lambda\right) \geq z_0(\gamma) \geq (1+\delta)\gamma$

(C3) $\lambda \geq \frac{2}{a} \ln\left(\frac{16m}{ac\varepsilon z_*}\right)$ *with* $a = \frac{\delta^2\gamma_0}{2(1+\delta)}$, $\varepsilon = \min\{\delta/2, 1/2\}$ *and* $c = \varepsilon^4/24$

then $\mathbf{E}[T] \leq \frac{2}{c\varepsilon}\left(m\lambda(1+\ln(1+c\lambda)) + \sum_{j=1}^{m} \frac{1}{z_j}\right)$

Informally, the two first conditions require a relationship between P and the distribution $D(P)$. In condition (C1), the theorem demands a certain probability z_j of creating an individual at level $j+1$ when some fixed portion γ_0 of the population is already at level j (or higher). Condition (C2) requires that in the fixed portion, the number of individuals at levels strictly higher than j (if those exist) tends to increase, e.g. by a multiplicative factor of $1+\delta$. The last condition (C3) requires a sufficiently large population size. When all the conditions are satisfied, an upper bound on the expected runtime (defined in terms of sampled/evaluated solutions) of the algorithm to reach A_{m+1} is guaranteed.

3.2 Feige's Inequality

To show that Condition (C1) and (C2) in the level-based theorem are satisfied, it will become necessary to compute lower bounds on the probability $\Pr(Y \geq j+1)$ where Y represents the level of an individual sampled from $D(P_t)$. It is often non-trivial to compute this probability directly, hence we resort to tail bounds that can be expressed in terms of the expectation of Y. While most tail inequalities provide upper bounds on $\Pr(Y \geq j+1)$, we will make use of a general result due to Feige to compute lower bounds [9]. The theorem is usually presented in the following form.

THEOREM 2. *Let $X_1, \ldots, X_n$ be non-negative, independent random variables with expectations $\mu_1, \ldots, \mu_n$ such that $\mu_i \leq 1$ for all i, define $X = \sum_{i=1}^n X_i$ and $\mu = \mathbf{E}[X]$. It holds for every $\delta > 0$ that*

$$\Pr(X < \mu + \delta) \geq \min\left\{\frac{1}{13}, \frac{\delta}{1+\delta}\right\}$$

For our purposes, it will be more convenient to use the following variant.

COROLLARY 3. *Let $Y_1, \ldots, Y_n$ be n independent random variables with support in $[0,1]$ and finite expectations, define $Y = \sum_{i=1}^n Y_i$ and $\mu = \mathbf{E}[Y]$. It holds for every $\delta > 0$ that*

$$\Pr(Y > \mu - \delta) \geq \min\left\{\frac{1}{13}, \frac{\delta}{1+\delta}\right\}$$

PROOF. For each i, define $x_i = 1 - Y_i$. It is clear that X_i also has support in $[0,1]$ and $\mathbf{E}[X_i] \leq 1$. Let $X = \sum_{i=1}^n X_i$, then $X = n - Y$ and $\mathbf{E}[X] = n - \mathbf{E}[Y] = n - \mu$. It follows from Theorem 2 that

$$\begin{aligned}\Pr(Y > \mu - \delta) &= \Pr(n - X > \mu - \delta) \\ &= \Pr(n - \mu + \delta > X) \\ &= \Pr(X < \mathbf{E}[X] + \delta) \\ &\geq \min\left\{\frac{1}{13}, \frac{\delta}{\delta+1}\right\}. \quad \square\end{aligned}$$

4. RUNTIME ANALYSIS ON LEADINGONES

As a warm-up example, and to illustrate the ease of use of our methods, we consider the standard benchmark function

$$\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j,$$

where the objective is to maximise the number of leading one-bits in the bitstring. It is well-known that the expected optimisation time of the (1+1) EA on LEADINGONES is $\Theta(n^2)$, and that this is optimal for the class of unary unbiased black-box algorithms [16]. In this analysis, we do not require Feige's inequality.

THEOREM 4. *The Univariate Marginal Distribution Algorithm (UMDA) with offspring population size $\lambda \geq b\ln(n)$ for some constant $b > 0$, parent population size $\mu = \gamma_0\lambda$ for any constants $\gamma_0 \leq \frac{1}{(1+\delta)13e}$ and $\delta > 0$, and margin $m = \mu/n$ has expected optimisation time $\mathcal{O}(n\lambda\ln(\lambda) + n^2)$ on* LEADINGONES.

PROOF. We apply Theorem 1 with the parameter $\gamma_0 := \mu/\lambda$, and use the canonical level-partition given by

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

In the following, we let Y denote the level of a sampled individual, or equivalently the number of leading one-bits in the solution. We first develop a generic bound on $\Pr(Y \geq j+1)$ which does not depend on γ. Then we use this bound to show that conditions (C1) and (C2) of Theorem 1 are satisfied.

For any population P_t, there exists an integer $j \geq 0$ and a real number $\gamma \in [0, \mu/\lambda)$ such that $|P_t \cap A_{j-1}^+| \geq \gamma_0\lambda = \mu$ and $|P_t \cap A_j^+| = \gamma\lambda < \mu$. In other words, among the μ best individuals in the current population, there are $\gamma\lambda$ individuals with a fitness at least $j+1$ and at least μ individuals with a fitness at least j. Hence

$$X_i = \mu \quad \text{and} \quad p_t(i) = 1 - 1/n \qquad \text{if } 1 \leq i \leq j.$$

Therefore,

$$\begin{aligned}\Pr(Y \geq j+1) &= \prod_{i=1}^{j+1} p_t(i) && (1)\\ &= \left(1 - \frac{1}{n}\right)^j p_t(j+1) && (2)\\ &\geq \frac{p_t(j+1)}{e}. && (3)\end{aligned}$$

If $\gamma = 0$, then none of the best μ individuals in the population has a one-bit in position $j+1$ and

$$X_{j+1} = 0 \quad \text{and} \quad p_t(j+1) = 1/n.$$

Condition (C1) can therefore be satisfied by choosing the same parameter $z_j = z_* = 1/(en)$ for all levels j.

If $\gamma > 0$, then among the best μ individuals in the population, there are $\gamma\lambda$ individuals with a one-bit in position $j+1$, and $\mu - \gamma\lambda$ individuals with a zero-bit in position $j+1$. Therefore,

$$X_{j+1} = \gamma\lambda \quad \text{and} \quad p_t(j+1) = \frac{\gamma\lambda}{\mu}.$$

It follows that

$$\Pr(Y \geq j+1) \geq \frac{p_t(j+1)}{e} = \frac{\gamma\lambda}{\mu e} \geq (1+\delta)\gamma$$

because the population size is $\lambda \geq (1+\delta)e\mu$. Condition (C2) is therefore satisfied.

Given that δ and γ are constants, the parameters a, ε and c are also constants. Furthermore, $1/z_* \in \text{poly}(n)$. Therefore, there must exist a constant b such that condition (C3) is satisfied whenever $\lambda \geq b\ln(n)$.

All three conditions are now satisfied, and the theorem implies that the UMDA has expected optimisation time

$$\mathcal{O}\left(n\lambda\ln(\lambda) + \sum_{j=1}^{n-1}\frac{1}{z_j}\right) = \mathcal{O}\left(n\lambda\ln(\lambda) + n^2\right). \quad \square$$

5. RUNTIME ANALYSIS ON ONEMAX

We now turn to another classical benchmark problem,

$$\textsc{OneMax}(x) = \sum_{i=1}^{n} x_i,$$

where the objective is to maximise the number of one-bits in the bitstring. It is well-known that the (1+1) EA solves this problem in expected time $\Theta(n\ln n)$, and this is optimal for the class of unary, unbiased black-box algorithms. While this problem may appear simpler than LeadingOnes, runtime analysis of UMDA on this problem is a much bigger challenge because the probability distribution induced by the UMDA follows a less clear trajectory than in the LeadingOnes problem. Surprisingly, no previous runtime analysis of UMDA seems available for OneMax. We demonstrate that the expected runtime can be obtained relatively easy with our methods. To obtain lower bounds on the tail of the level-distribution, we make use of the Feige inequality.

Theorem 5. *The Univariate Marginal Distribution Algorithm (UMDA) with offspring population size $\lambda \geq b\ln(n)$ for some constant $b > 0$, parent population size $\mu = \gamma_0\lambda$ for any constants $\gamma_0 \leq \frac{1}{(1+\delta)13e}$ and $\delta \in (0,1)$, and margins $m = \mu/n$, has expected optimisation time $\mathcal{O}(n\lambda\ln\lambda)$ on* OneMax.

Proof. We will use the canonical partition $(A_0, \ldots, A_n)$ given by

$$A_j := \{x \in \{0,1\}^n \mid \textsc{OneMax}(x) = j\}$$

Furthermore, we use the parameter $\gamma_0 := \mu/\lambda$, and let Y be the level of a sampled individual (or equivalently the number of one-bits in the sampled solution).

The probabilities (or components) of p_t can be categorised into three groups. Probabilities at the upper margin $1 - 1/n$, probabilities at the lower margin $1/n$, and intermediary probabilities in the closed interval $[1/\mu, 1 - 1/\mu]$. Due to linearity of the fitness function, the components of p_t can be rearranged without changing the distribution of Y. We can therefore assume w.l.o.g. the rearrangement so that there exists integers $k, \ell \geq 0$ satisfying

$$\begin{aligned}
1 \leq X_i < \mu \quad &\text{and} \quad p_t(i) = X_i/\mu \qquad &&\text{if } 1 \leq i \leq k,\\
X_i = \mu \quad &\text{and} \quad p_t(i) = 1 - 1/n \qquad &&\text{if } k < i \leq k+\ell, \text{and}\\
X_i = 0 \quad &\text{and} \quad p_t(i) = 1/n \qquad &&\text{if } k+\ell < i.
\end{aligned}$$

By these assumptions, it follows that

$$\sum_{i=k+1}^{k+\ell} X_i = \mu\ell \quad \text{and} \quad \sum_{i=k+\ell+1}^{n} X_i = 0. \tag{4}$$

In the following, we define $Y_{i,k}$ to be the number of sampled one-bits due to the subset $(p_t(i), \ldots, p_t(k))$ of components. For any population P_t, there exists an integer $j \geq 0$ and a real number $\gamma \in [0, \mu/\lambda)$ such that $|P_t \cap A_{j-1}^+| \geq \gamma_0\lambda = \mu$ and $|P_t \cap A_j^+| = \gamma\lambda < \mu$. In other words, among the μ best individuals in the current population, there are $\gamma\lambda$ individuals with a fitness at least $j+1$ and at least μ individuals with a fitness at least j. Hence, the total number of one-bits among the fittest μ individuals must satisfy

$$\sum_{i=1}^{n} X_i \geq \gamma\lambda(j+1) + (\mu - \gamma\lambda)j = \gamma\lambda + \mu j. \tag{5}$$

Combining Eqs. (4) and (5) when $k \geq 1$ give

$$\mathbf{E}[Y_{1,k}] = \sum_{i=1}^{k} p_t(i) = \frac{1}{\mu}\sum_{i=1}^{k} X_i \geq \frac{\gamma\lambda}{\mu} + j - \ell. \tag{6}$$

We first consider the case when $\gamma > 0$. This condition implies that among the best μ individuals in the population, there exists both individuals with at least $j+1$ one-bits and individuals with exactly j one-bits. Hence, there must exist at least one index i such that the probability $p_t(i)$ is not at one of the margins. Therefore $k > 0$ which allows us to use Eq. (6). Together with Feige's inequality, we get

$$\begin{aligned}
\Pr(Y_{1,k} \geq j - \ell + 1) &= \Pr\left(Y_{1,k} > j - \ell + \frac{\gamma\lambda}{\mu} - \frac{\gamma\lambda}{12\mu}\right)\\
&\geq \Pr\left(Y_{1,k} > \mathbf{E}[Y_{1,k}] - \frac{\gamma\lambda}{12\mu}\right)\\
&\geq \min\left\{\frac{1}{13}, \frac{\frac{\gamma\lambda}{12\mu}}{\frac{\gamma\lambda}{12\mu}+1}\right\}\\
&= \min\left\{\frac{1}{13}, \frac{\gamma\lambda}{\gamma\lambda+12\mu}\right\}\\
&\geq \frac{\gamma\lambda}{13\mu}.
\end{aligned}$$

The probability of sampling an individual with at least $j+1$ one-bits in the next generation is therefore bounded

from below by

$$\begin{aligned}\Pr(Y \geq j+1) &\geq \Pr(Y_{1,k} \geq j-\ell+1)\Pr(Y_{k+1,k+\ell} = \ell)\\ &\geq \frac{\gamma\lambda}{13\mu}\left(1-\frac{1}{n}\right)^{\ell}\\ &\geq \frac{\gamma\lambda}{13e\mu}\\ &\geq (1+\delta)\gamma.\end{aligned}$$

Consider now the case where $\gamma = 0$. We need to consider two sub-cases, either $k = 0$ or $k \geq 1$. If $k = 0$, then with our assumption, the j first probabilities are at the upper margin $1 - 1/n$, and the last $n - j$ probabilities are at the lower margin $1/n$. In order to obtain a search point with $j + 1$ one-bits, it is sufficient to sample exactly j one-bits in the first j positions and exactly one one-bit in the last $n - j$ positions. Hence, we have

$$\begin{aligned}\Pr(Y \geq j+1) &\geq \Pr(Y_{1,j} = j)\Pr(Y_{j+1,n} \geq 1)\\ &\geq \left(1-\frac{1}{n}\right)^{j}\left(\frac{n-j}{n}\right)\left(1-\frac{1}{n}\right)^{n-j-1}\\ &= \left(1-\frac{1}{n}\right)^{n-1}\left(\frac{n-j}{n}\right)\\ &\geq \left(\frac{n-j}{en}\right).\end{aligned}$$

In the sub-case where $k > 0$, we note from Eq. (6) that

$$\mathbf{E}[Y_{2,k}] = \mathbf{E}[Y_{1,k}] - p_t(1) \geq j - \ell - p_t(1)$$

Again, by Feige's inequality, we get

$$\begin{aligned}\Pr(Y_{2,k} \geq j-k) &= \Pr(Y_{2,k} > j-\ell-p_t(1)-(1-p_t(1)))\\ &\geq \Pr(Y_{2,k} > \mathbf{E}[Y_{2,k}] - (1-p_t(1)))\\ &\geq \min\left\{\frac{1}{13}, \frac{1-p_t(1)}{2-p_t(1)}\right\}\\ &> \frac{1-p_t(1)}{13}.\end{aligned}$$

The probability of sampling an individual with at least $j + 1$ one-bits in this configuration can now be bounded from below as

$$\begin{aligned}&\Pr(Y \geq j+1)\\ &\quad> \Pr(Y_1 = 1)\Pr(Y_{2,k} \geq j-\ell)\Pr(Y_{k+1,k+\ell} = \ell)\\ &\qquad\geq \left(\frac{p_t(1)(1-p_t(1))}{13}\right)\left(1-\frac{1}{n}\right)^{\ell}\\ &\qquad\geq \left(\frac{(1/\mu)(1-1/\mu)}{13}\right)\left(1-\frac{1}{n}\right)^{\ell} \geq \frac{1}{14e\mu}.\end{aligned}$$

The last inequality holds for $\mu \geq 14$, which in turn only requires n to be larger than some constant. Hence, in the case when $\gamma = 0$, we get

$$\begin{aligned}\Pr(Y \geq j+1) &\geq \min\left\{\frac{1}{14e\mu}, \frac{n-j}{en}\right\}\\ &\geq \frac{n-j}{14e\mu(n-j)+en} =: z_j.\end{aligned}$$

Clearly, there exists a z_* with $1/z_* \in \text{poly}(n)$ such that $\Pr(Y \geq j+1) \geq z_*$ for all $j \in [n-1]$. Condition (C2) is therefore satisfied.

Finally, we consider condition (C3) regarding the population size. The parameters δ and $\gamma_0 = \mu/\lambda$ are constants with respect to n, therefore the variables a, ε and c in condition (C3) are also constants, and $1/z_* \in \text{poly}(n)$. Hence, there must exist a constant $b > 0$ such that condition (C3) is satisfied when $\lambda \geq b\log(n)$.

To conclude, the expected optimisation time is

$$\begin{aligned}&\mathcal{O}\left(n\lambda\ln(\lambda) + \sum_{j=1}^{n-1}\frac{1}{z_j}\right)\\ &\qquad= \mathcal{O}\left(n\lambda\ln(\lambda) + 14e\mu n + \sum_{j=0}^{n-1}\frac{en}{n-j}\right)\\ &\qquad= \mathcal{O}(n\lambda\ln\lambda). \quad \square\end{aligned}$$

6. CONCLUSION

Runtime analysis has become one of the standard approaches to theoretical analysis of evolutionary algorithms. Runtime results are available for a large number of algorithms and problems. Runtime analysis provides valuable insight into the behaviour of an algorithm, as it shows how the efficiency of the algorithm depends on its parameters and the characteristics of the fitness landscape.

However, with a few exceptions, runtime analyses have so far largely been unavailable for estimation of distribution algorithms (EDAs). An analysis has seemed elusive, even for apparently simple settings such as UMDA on OneMax. Many of the techniques that were developed to analyse evolutionary algorithms such as the (1+1) EA, do not transfer to the more complex EDAs. Mathematical simplifications that have been employed in theoretical studies of EDAs, such as infinite population size or time going to infinite, do not yield meaningful results in the context of runtime.

This paper shows that the *level-based technique* which was recently introduced to analyse population-based evolutionary algorithms is directly applicable to runtime analysis of estimation of distribution algorithms. In particular, the technique applies to any estimation of distribution algorithm where the sampling distribution can be computed from the current population. We demonstrate this new approach by analysing the expected runtime of the univariate marginal distribution algorithm (UMDA) on standard benchmark functions. In particular, for the LeadingOnes problem, we show a significantly improved upper bound on the runtime. Furthermore, we analyse the expected runtime on OneMax, a problem that has been remained open for a very long time.

The significance of this work lies in the relatively ease with which these results were obtained. Previous runtime analyses of EDAs have been highly involved, even in simple settings, preventing studies of more interesting scenarios. We believe that the new technique could make it possible to analyse the runtime of more complex EDAs and more challenging fitness landscapes.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 618091 (SAGE).

7. REFERENCES

[1] Tianshi Chen, Per Kristian Lehre, Ke Tang, and Xin Yao. When is an estimation of distribution algorithm better than an evolutionary algorithm? In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009*, pages 1470–1477, 2009.

[2] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. On the analysis of average time complexity of estimation of distribution algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007*, pages 453–460, 2007.

[3] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. Rigorous time complexity analysis of univariate marginal distribution algorithm with margins. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009*, pages 2157–2164, 2009.

[4] Tianshi Chen, Ke Tang, Guoliang Chen, and Xin Yao. Analysis of computational time of simple estimation of distribution algorithms. *IEEE Trans. Evolutionary Computation*, 14(1):1–22, 2010.

[5] Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Level-based analysis of genetic algorithms and other search processes. In *Proceedings of Parallel Problem Solving from Nature - PPSN XIII*, pages 912–921, 2014.

[6] Duc-Cuong Dang and Per Kristian Lehre. Evolution under partial information. In *Genetic and Evolutionary Computation Conference, GECCO'14*, pages 1359–1366, 2014.

[7] Duc-Cuong Dang and Per Kristian Lehre. Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms. In *(To appear) Foundations of Genetic Algorithms XIII, FOGA'15*, 2015.

[8] Stefan Droste. A rigorous analysis of the compact genetic algorithm for linear functions. *Natural Computing*, 5(3):257–283, 2006.

[9] Uriel Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. In *Proc. of the 36th STOC*, pages 594–603, 2004.

[10] Tobias Friedrich, Timo Kötzing, Martin Krejca, and Andrew M. Sutton. The benefit of sex in noisy evolutionary search. *CoRR*, abs/1502.02793, 2015.

[11] Cristina González, Jose Antonio Lozano, and Pedro Larrañaga. Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 12:465–479, 2000.

[12] Georges R. Harik, Fernando G. Lobo, and David E. Goldberg. The compact genetic algorithm. *IEEE Trans. Evolutionary Computation*, 3(4):287–297, 1999.

[13] Markus Höhfeld and Günter Rudolph. Towards a theory of population-based incremental learning. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 1997*, pages 1–5, 1997.

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

[15] Pedro Larrañaga and Jose A. Lozano, editors. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, volume 2 of *Genetic Algorithms and Evolutionary Computation*. Springer, 2002.

[16] Per Kristian Lehre and Carsten Witt. Black-Box Search by Unbiased Variation. *Algorithmica*, 64(4):623–642, 2012.

[17] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 178–187. Springer Berlin Heidelberg, 1996.

[18] Heinz Mühlenbein. The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.

[19] Pietro Simone Oliveto, Jun He, and Xin Yao. Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4(3):281–293, 2007.

[20] Martin Pelikan, Kumara Sastry, and David E. Goldberg. Scalability of the bayesian optimization algorithm. *Int. J. Approx. Reasoning*, 31(3):221–258, 2002.

[21] Jonathan L. Shapiro. Drift and scaling in estimation of distribution algorithms. *Evolutionary Computation*, 13(1):99–123, 2005.

[22] Qingfu Zhang. On the convergence of a factorized distribution algorithm with truncation selection. *Complexity*, 9(4):17–23, 2004.

[23] Qingfu Zhang and Heinz Mühlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Trans. Evolutionary Computation*, 8(2):127–136, 2004.