# When Is an Estimation of Distribution Algorithm Better than an Evolutionary Algorithm?

Tianshi Chen, Per Kristian Lehre, Ke Tang, and Xin Yao

***Abstract*—Despite the wide-spread popularity of estimation of distribution algorithms (EDAs), there has been no theoretical proof that there exist optimisation problems where EDAs perform significantly better than traditional evolutionary algorithms. Here, it is proved rigorously that on a problem called SUBSTRING, a simple EDA called univariate marginal distribution algorithm (UMDA) is efficient, whereas the (1+1) EA is highly inefficient. Such studies are essential in gaining insight into fundamental research issues, i.e., what problem characteristics make an EDA or EA efficient, under what conditions an EDA is expected to outperform an EA, and what key factors are in an EDA that make it efficient or inefficient.**

## I. INTRODUCTION

Estimation of Distribution Algorithms (EDA) differ from traditional evolutionary algorithms (EA) in the way offspring individuals are produced from the selected parent individuals. Instead of applying genetic operators like mutation and crossover to the parents, EDAs estimate a probability distribution over the search space based on how the parent individuals are distributed in the search space, and then sample the offspring individuals from this distribution [1]. Estimating the probability distribution can be significantly more computationally expensive than applying the genetic operators. It is therefore important to study what can be gained from paying this additional computational cost. Several experimental studies indicate that estimating a distribution has positive effects, however it is still not completely understood why and how the estimated distribution benefit the evolutionary search. It has been suggested that in certain problem domains, for example in software test case generation [2], the estimated probability distribution could provide the user of the EDA with additional information about the problem structure. It has also been argued that the EDAs' potential in discovering problem structure, in particular problem variable dependencies, can make the search more effective and reduce the overall runtime [3]. However, these arguments have until now been based on non-rigorous methods.

To better understand the situations where estimating a probability distribution can significantly reduce the runtime compared with traditional evolutionary algorithms, this paper applies methods that have been developed over the last decade to analyse the runtime of randomised search heuristics [4]. Droste *et al* [5] presented a number of well known results about runtime of the (1+1) EA, and those on linear functions and LEADINGONES [5] have been frequently mentioned in the literature [4]. Some results have also been obtained for the more complex evolutionary algorithms that employ a population [6]. Additionally, results start to emerge on the runtime of evolutionary algorithms on classical combinatorial optimisation problem, for example the vertex cover problem [7], [8], [9].

The authors are listed in alphabetic order. T. Chen, K. Tang and X. Yao are with the Nature Inspired Computation and Applications Laboratory (NICAL), Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. P. K. Lehre (X. Yao) is (also) with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK. (Email: cetacy@mail.ustc.edu.cn, P.K.Lehre@cs.bham.ac.uk, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk)

The runtime of EDAs is a much less explored area. The runtime of a univariate EDA called compact genetic algorithm (cGA) was analysed rigorously by Droste in [10]. Under certain conditions, the univariate distribution updated by the cGA can become stuck, leading to an infinite runtime, hence the expression for expected runtimes are conditional on finite runtime. With this restriction, it was shown that the expected runtime of cGA is in general bounded from below by $\Omega(K\sqrt{n})$ for any function. Here, $K$ is a parameter of the algorithm, often set to $K = n$, that adjusts the learning rate of the algorithm. It was further shown that given a parameter setting of $K = n^{1/2+\varepsilon}$, ($\varepsilon > 0$ is a constant), the algorithm optimises the linear function ONEMAX within $O(K\sqrt{n})$ iterations, implying that ONEMAX is one of the simplest functions for the cGA. On the other hand, it was proved that there exists harder linear functions for the cGA, notably the BINVAL function, which raises the asymptotic runtime bound for the class of linear functions to $\Theta(Kn)$ [10].

In a series of papers, Chen *et al* [11], [12], [13] have analysed the runtime of another univariate EDA called the univariate marginal distribution algorithm (UMDA) on several non-linear problems. In contrast to the cGA, the UMDA employs a population of individuals, where the parent and offspring population sizes are specified by parameters $N$ and $M$, respectively. The initial analysis was made under a "no-random-error" assumption, showing that the expected runtime of UMDA on the LEADINGONES function is $O(Nn)$ [11]. Furthermore, it was proved that a variant of this problem called TRAPLEADINGONES is hard for the UMDA. However the "no-random-error" assumption does not hold in practice. Improved techniques were therefore later developed to lift this assumption, and proving that the runtime of $O(Nn)$ on the LEADINGONES function still holds [13]. This and the following results hold when the population sizes are sufficiently large, i.e. as in Theorem 2 in this paper.

As for the cGA, the probability distribution updated by the traditional UMDA can under certain circumstances get stuck at certain values. To alleviate for this problem, the univariate distribution learned by the UMDA can be restricted in some way. One such idea is the *relaxation by margins*, which restricts the estimated marginal probabilities in the univariate distribution to the interval $[1/M, 1-1/M]$. It has been proved that this modification can dramatically improve the runtime on some fitness functions [13]. In particular, introducing margins on the unimodal fitness function BVLEADINGONES improves the runtime from infinite (with overwhelming probability), to $O(Nn)$ (with overwhelming probability) [13]. Gonzáles has proved that using Laplace correction, which is a similar but earlier idea, one can obtain a general exponential upper bound on the expected runtime of several EDAs, including UMDA on any fitness function [14]. On the other hand, in recent work, it has been proved that the function TRAPLEADINGONES remains hard for the UMDA, even with relaxation by margins [12].

A comparison between the results above and the results that have been obtained for the (1+1) EA appears unfavourable for the cGA and the UMDA. It is well known that the expected runtime of the (1+1) EA on ONEMAX is $\Theta(n \log n)$, and that the same runtime bound also holds for the entire class of linear functions [5]. Hence, assuming the parameter settings $K = n^{1/2+\varepsilon}$ ($\varepsilon > 0$ constant) considered in [10], the (1+1) EA is asymptotically better than the cGA. Even on the fitness function ONEMAX, which belongs to the easiest functions for cGA, the cGA has asymptotically worse expected runtime than (1+1) EA. The expected runtime of (1+1) EA on LEADINGONES is $\Theta(n^2)$ [5]. Currently, there does not exist a lower bound on the runtime of UMDA on LEADINGONES that matches the upper bound of $O(Nn)$. However, for the super-quadratic population sizes considered above, even the duration of the first generation of UMDA is asymptotically longer than the whole runtime of (1+1) EA on LEADINGONES. It is currently an open problem whether the cGA and the UMDA perform more favourably on these functions for different parameter settings.

Our contribution is a rigorous runtime analysis of the (1+1) EA and the UMDA on a problem called SUBSTRING showing that the EDA is highly efficient on the problem whereas the EA fails completely. This contribution is significant, because there exists no previous theoretical comparison between the runtime of EAs and EDAs.

The rest of this paper is organised as follows. Section II defines the SUBSTRING problem. Section III introduces the analytical techniques that will be used. The runtime of the algorithms is analysed in Section IV. Finally, Section V concludes the paper.

## II. THE SUBSTRING PROBLEM

We hypothesise that the UMDA and the (1+1) EA behave differently on fitness functions where some variables do not influence fitness. The intuition is that the marginal probability distribution in UMDA only changes for variables that are subject to selection, while the distribution remains (approximately) constant for other variables. On the other hand, the mutation operator in (1+1) EA will drive variables that are not subject to selection towards a uniform distribution. Hence, if the subset of variables subject to selection differs within the search space, then the two algorithms might follow different search trajectories.

To make this idea more precise, we construct the SUBSTRING function. The formal definition of SUBSTRING is presented in Definition 1 at the top of the next page. The global optimum of SUBSTRING is $1^n$. The output of SUBSTRING for a given search point (except the global optimum), as illustrated in Figure 1, is essentially the position of the rightmost occurrence of the substring $1^{\alpha n}$, or the number of leading 1-bits if no such substring exists. As a consequence, for any search point (except the optimum), only at most $\alpha n$ out of $n$ variables determine the fitness of the search point and thus are subject to selection. A non-optimal search point can be improved by flipping the first 0-bit after the rightmost substring $1^{\alpha n}$, or the first 0-bit after the leading 1-bits if no substring $1^{\alpha n}$ exists. This improvement can be likened to "moving" the substring $1^{\alpha n}$ from the left to the right.

## III. ANALYTICAL TOOLS

Before introducing the theoretical results in this paper, we need to introduce some analytical tools which will be utilised in the proofs. First we introduce some tail probability techniques, among which Chernoff bounds might be the most famous techniques in the community of theoretical evolutionary computation:

*Lemma 1 (Chernoff bounds [15]):* Let $X_1, X_2, \dots, X_k \in \{0,1\}$ be $k$ independent random identically distributed variables taking the value from $\{0,1\}$,

$$\forall i \neq j : \mathbb{P}(X_i = 1) = \mathbb{P}(X_j = 1),$$

where $i, j \in \{1, \dots, k\}$. Let $X$ be the sum of those random variables, i.e., $X = \sum_{i=1}^{k} X_i$, then we have

- $\forall 0 < \delta < 1$:
$$\mathbb{P}\Big(X < (1-\delta)\mathbb{E}[X]\Big) < e^{-\mathbb{E}[X]\delta^2/2}.$$
- $\forall \delta \leq 2e - 1$:
$$\mathbb{P}\Big(X > (1+\delta)\mathbb{E}[X]\Big) < e^{-\mathbb{E}[X]\delta^2/4}.$$

In addition to Chernoff bounds, we borrow the following lemma from the field of statistics:

*Lemma 2 ([13], [16], [17]):* [1] Consider sampling without replacement from a finite population $\{X_1, \dots, X_N\} \in \{0,1\}^N$. Let $\{X_1, \dots, X_M\} \in \{0,1\}^M$ be a sample of size $M$ drawn randomly without replacement from the whole population, $X^{(M)}$ and $X^{(N)}$ be the sums of the random variables in the sample and population respectively, i.e.,

[1] The first inequality of the lemma comes from Corollary 1.1 of [17] (or a similar form can be found in [16]), and the second inequality comes from Eq. 3.3 of [17]. They have been used to cope with the time complexity analysis of EDAs in [13].

*Definition 1 (*SUBSTRING*):* For any constant $\alpha, 0 < \alpha < 1/3$, define

$$\text{SUBSTRING}(x) := \begin{cases} 2n & \text{if } x = 1^n; \\ \max_{1 \leq i \leq n} i \cdot \prod_{j=\max\{i-\alpha n, 1\}}^{i} x_j & \text{otherwise.} \end{cases}$$

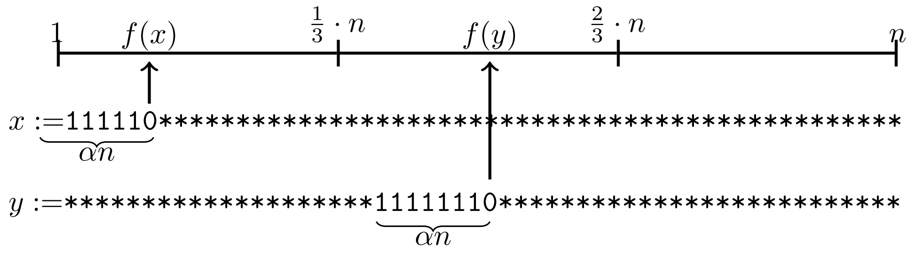
1
$f(x)$
$\frac{1}{3} \cdot n$
$f(y)$
$\frac{2}{3} \cdot n$
$n$
$x :=$111110*****************************************************
$\alpha n$
$y :=$******************1111110*************************
$\alpha n$

Fig. 1. SUBSTRING evaluated on two search points $x$ and $y$.

$X^{(M)} = \sum_{i=1}^{M} X_i$ and $X^{(N)} = \sum_{i=1}^{N} X_i$, then we have:

$$\mathbb{P}\left(X^{(M)} - \frac{MX^{(N)}}{N} \geq Mc\right) < e^{-2Mc^2},$$
$$\mathbb{P}\left(\left|X^{(M)} - \frac{MX^{(N)}}{N}\right| > Mc\right) < 2e^{-2Mc^2},$$

where $c \in [0, 1]$ is some constant.

Finally, we will use the drift theorem [18], [19], which is a general technique for proving exponential lower bounds on the first hitting-time in Markov processes and which is frequently applied in the analysis of evolutionary algorithms [18], [20]. Here, we will use a simplified version of the theorem developed by Oliveto and Witt [21]:

*Lemma 3 (Simplified Drift Theorem [21]):* Let $X_t, t \geq 0$, be the random variables describing a Markov process over the state space $S := \{0, 1, ..., N\}$, and denote

$$\Delta(i) := (X_{t+1} - X_t \mid X_t = i)$$

for $i \in S$ and $t \geq 0$. Suppose there exists an interval $[a, b]$ of the state space and three constants $\beta, \delta, r > 0$ such that for all $t \geq 0$

1) $\mathbf{E}[\Delta(i)] \geq \beta$ for $a < i < b$, and
2) $\mathbb{P}(\Delta(i) = -j) \leq 1/(1+\delta)^{j-r}$ for $i > a$ and $j \geq 1$,

then there is a constant $c^* > 0$ such that for

$$T^* := \min\{t \geq 0 : X_t \leq a \mid X_0 \geq b\}$$

it holds $\mathbb{P}\left(T^* \leq 2^{c^*(b-a)}\right) = 2^{-\Omega(b-a)}$.

## IV. THEORETICAL RESULTS

### A. Runtime of (1+1) Evolutionary Algorithm

We will start by analysing the behaviour of the (1+1) EA on SUBSTRING. We will first state the theoretical results. Then we will outline the proof ideas and explain informally why the SUBSTRING problem is hard for the (1+1) EA. Finally, we will provide the formal proofs. The (1+1) EA is defined below. Note that the mutation rate in this version of the algorithm has been generalised to $\chi/n$, where the parameter $\chi$ has to satisfy $\chi/n = o(1)$, e.g. $\chi > 0$ can be any positive constant.

**Algorithm 1** (1+1) Evolutionary Algorithm.

```
1: Sample $x$ uniformly at random from $\{0,1\}^n$.
2: repeat
3:     $x' \leftarrow x$. Flip each bit of $x'$ with probability $\chi/n$.
4:     if $f(x') \geq f(x)$ then
5:         $x \leftarrow x'$.
6:     end if
7: until termination condition met.
```

Lemma 4 will be used when proving Theorem 1.

*Lemma 4:* Let $A$ be any subset of $\{0,1\}^n$, and $f : \{0,1\}^n \rightarrow \mathbb{R}$ any pseudo-boolean function that for any bitstring $x \in A$ satisfies

$$\begin{aligned} &f(x_1 \cdots x_{i-1} \cdot x_i \cdot x_{i+1} \cdots x_n) \\ =&f(x_1 \cdots x_{i-1} \cdot \overline{x_i} \cdot x_{i+1} \cdots x_n). \end{aligned}$$

Let random variable $X_t \in \{0,1\}, t \geq 0$, denote the value of the $i^{th}$ bit in the current search point in iteration $t$ of (1+1) EA with mutation probability satisfying $\chi/n = o(1)$ and $\chi > 0$. If the current search point belongs to set $A$ for any iteration $0$ to $t$, then

$$\mathbf{E}[|X_t - X_0|] \geq \frac{1}{1+e^\chi} - \frac{1}{1+e^\chi} \cdot \left(1 - \frac{\chi(1+e^{-\chi})}{n}\right)^t.$$

*Proof:* In order for the $i^{th}$ bit to be flipped, it suffices that the mutation operator flips bit $i$, and no other bits, an event which happens with probability higher than $\chi/ne^\chi$. In order for the $i^{th}$ bit to be flipped, it is necessary that the mutation operator flips at least bit $i$, an event which happens with probability $\chi/n$.

Define $p_t := \mathbb{P}(|X_t - X_0| = 1)$. Clearly, $\mathbf{E}[|X_t - X_0|] = p_t$. We then have $p_0 = 0$, and $p_{t+1} \geq p_t(1 - \chi/n) + (1 - p_t)\chi/ne^\chi = p_t(1 - \chi(1+e^{-\chi})/n) + \chi/ne^\chi$ for any $t \geq 0$. The lemma holds for iteration $t = 1$ because $p_1 = p_0(1 -$

$\chi(1+e^{-\chi})/n)+\chi/ne^{\chi} = \chi/ne^{\chi}$. Assuming that the lemma also holds for $t = k$ for a $k \geq 1$, one can show that it must also hold for $t = k + 1$

$$p_{k+1} \geq p_k \cdot \left(1 - \frac{\chi(1+e^{-\chi})}{n}\right) + \frac{\chi}{ne^{\chi}}$$
$$\geq \frac{1}{1+e^{\chi}} - \frac{1}{1+e^{\chi}} \cdot \left(1 - \frac{\chi(1+e^{-\chi})}{n}\right)^{k+1}.$$

The lemma now follows by induction on iteration number $t$. ∎

The main theoretical result in this section is the following theorem.

*Theorem 1:* The probability that (1+1) EA with bitwise mutation probability $\chi/n$, for any constant $\chi > 0$, finds the global optimum of SubString in less than $2^{cn}$ iterations is $e^{-\Omega(n)}$, where $c > 0$ is a constant.

We now explain the ideas of proving the above theorem. Assume first the mutation rate parameter setting of $\chi = 1$, which corresponds to the traditional $1/n$ mutation rate in the (1+1) EA. We call the position of the right-most occurrence of the substring $1^{\alpha n}$ the *substring position* in a search point. In order to reach the optimum, it is necessary to increase the substring position to its maximum value, i.e. to obtain a search point with the suffix $1^{\alpha n}$. By the definition of the fitness function and the acceptance criterion of the (1+1) EA, the substring position increases monotonically. The substring position can be increased by flipping the left-most 0-bit after the substring position. Hence, the (1+1) EA will increase the substring position in a similar manner to how the (1+1) EA increases the fitness on LeadingOnes. However, in contrast to LeadingOnes, as the substring position increases, it is not necessary to maintain the initial leading 1-bits in the search point. The (1+1) EA will therefore tend to "forget" such acquired 1-bits. Lemma 4 shows how many such bits that are forgotten in expectation as a function of time $t$.

The proof idea is to show that the time the (1+1) EA needs to obtain a search point with maximal substring position is so long that it also has lost many of the initial 1-bits. When a search point with suffix $1^{\alpha n}$ has been found, many leading 1-bits have been lost and it becomes difficult for the (1+1) EA to obtain the globally optimal search point $1^n$ because the fitness function is essentially a needle in the haystack with respect to the first $n \cdot (1 - \alpha)$ bit-positions. One could hypothesise that this problem could be alleviated, either by reducing the mutation rate $\chi$ in order to lower the rate of forgetting, or by increasing the mutation rate $\chi$, in order to reduce the time to reach the optimum. However, as is shown in Theorem 1, lowering the mutation rate also increases the time to increase the substring position, and as shown in Lemma 4, increasing the mutation rate, also increases the rate of forgetting. Based on the above discussions, we provide the formal proof of Theorem 1.

*Proof of Theorem 1:* Define the *substring position* of a search point $x$ as the largest index $p$ such that $x_{p-\epsilon n} = x_{p-\epsilon n+1} = \cdots = x_p = 1$. Having a substring position of $n$ is a necessary, but not sufficient, condition for the search point to be optimal. In the following, the *prefix* of the current search point corresponds to the first $n/3$ bits, and the *suffix* corresponds to the last $n/3$ bits of the search point.

We divide any run of the (1+1) EA on SubString into three phases depending on the current substring position $p$, which by the definition of the fitness function is monotonically increasing. *Phase 1* starts with the initial iteration and lasts until the substring position is higher than $2n/3$. A failure occurs in Phase 1 if the initial search point has substring position higher than $2n/3$. *Phase 2* lasts as long as the substring position is at least $2n/3$ and less than $n$. A failure occurs in Phase 2 if the phase ends with a search point having less than $n/(12(1+e^{\chi}))$ 0-bits in the prefix. *Phase 3* begins with a search point with substring position $n$ and at least $n/(12(1+e^{\chi}))$ 0-bits in the prefix and lasts until the optimum has been found. A failure occurs in Phase 3 if the phase lasts less than $2^{cn}$ iterations for some $c$. The theorem follows if we can prove that the probability of failure in any of the three phases is exponentially small.

By union bound, the failure probability in Phase 1 is less than $n2^{-\alpha n} = e^{-\Omega(n)}$. To bound the failure probability in Phase 2, first note that by using the same arguments as in the proof of Theorem 17 in Droste *et al* [5], it can be shown that there exists a constant $c_1$ such that the probability that the duration of Phase 2 is shorter than $c_1 n^2$ iterations is $e^{-\Omega(n)}$. Given that the duration of Phase 2 is at least $c_1 n^2$ with overwhelming probability, we are now in position to bound the failure probability during Phase 2. The substring position is at least $2n/3$ during this phase, hence all the bits in the prefix remain independent for the rest of the run. For any index $i, 1 \leq i \leq n/3$, let random variable $Y_{i,t} \in \{0, 1\}$ be an indicator variable denoting whether the $i^{th}$ bit in iteration $t$ of Phase 2 is 0. In the most optimistic case, $Y_{i,0} = 0$. Hence, by Lemma 4, the expected value of $Y_{i,t}$, where $t \geq c_1 n^2$ is the duration of Phase 2 is

$$\mathbf{E}[Y_{i,t}] \geq \frac{1}{1+e^{\chi}} - \frac{1}{1+e^{\chi}} \cdot \left(1 - \frac{\chi(1+e^{-\chi})}{n}\right)^{c_1 n^2}$$
$$\geq \frac{1}{1+e^{\chi}} - \frac{1}{1+e^{\chi}} \cdot \exp(-\Omega(n)).$$

Hence, for sufficiently large $n$, $\mathbf{E}[Y_{i,t}] \geq 1/(3(1+e^{\chi}))$. Let random variable $Y := \sum_{i=1}^{n/3} Y_{i,t}$ denote the number of 0-bits when Phase 2 ends. The expectation of $Y$ is at least $n/(9(1+e^{\chi}))$, and by Chernoff bounds, the probability that there are less than $n/(12(1+e^{\chi})$, 0-bits in the prefix when Phase 2 ends is $e^{-\Omega(n)}$.

Finally, we bound the failure probability during Phase 3 using simplified drift analysis [21]. In order to obtain the optimal search point, it is necessary that the prefix consists of 1-bits only. Let the state $i \in \{0, ..., N\}$ be the number of 0-bits in the prefix, with $N := n/3$. Furthermore, define $a := 0$ and $b := n/(12(1+e^{\chi}))$. The remaining part of the analysis is now practically identical to the analysis of (1+1) EA on Needle in [21]. Assuming $a < i < b$, the expected drift in the process is $\mathbf{E}[\Delta(i)] = \chi((n/3 - i)/n - i/n) \geq \chi/6$, and condition 1 of the drift theorem holds with $\beta = \chi/6$. In order to decrease the number of 0-bits in the prefix by $j$, it is necessary to flip $j$ 0-bits simultaneously, an event which happens with probability

$$\begin{aligned}\binom{\frac{n}{12(1+e^{\chi})}}{j}\cdot(\chi/n)^j &\leq \frac{(n/e^{\chi})^j}{j!}\cdot(\chi/n)^j\\ &\leq 1/j! \leq 2^{-j+1},\end{aligned}$$

so condition 2 of the theorem holds with $\delta = r = 1$. Hence, the probability that the optimum has been found within $2^{cn}$ iterations, is bounded from above by $e^{-\Omega(n)}$, for some constant $c$. ■

Theorem 1 shows that the success probability of the (1+1) EA on SUBSTRING is exponentially low, for any mutation rate $\chi/n$, assuming $\chi > 0$ is a constant. This means that even by restarting the algorithm, the SUBSTRING problem remains hard for the (1+1) EA.

It is an open problem how a population-based EA will behave on SUBSTRING. However, assuming that the algorithm does not use crossover such that genetic material is not transmitted between individuals, one can conjecture that each individual in the population will suffer similar problems to the (1+1) EA.

We will now show that the UMDA finds the solution to SUBSTRING easily.

## *B. Runtime of UMDA with Truncation Selection*

The UMDA concerned in this paper is shown in Table 2, where we let $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ be an individual (search point), $\xi_t$ and $\xi_t^{(s)}$ be the populations before and after the selection at the $t^{th}$ generation ($t \in \mathbb{N}^+$) respectively, $p_{t,i}(1)$ ($p_{t,i}(0)$) be the estimated marginal probability of the $i^{th}$ bit of an individual to be 1 (0) at the $t^{th}$ generation, and the indicators $\delta(x_i|1)$ be defined as follows:

$$\delta(x_i|1) := \begin{cases} 1, & x_i = 1,\\ 0, & x_i = 0.\end{cases}$$

Let $\mathbf{P}_t(\mathbf{x}) := \Big(p_{t,1}(x_1), p_{t,2}(x_2), \ldots, p_{t,n}(x_n)\Big)$, be a vector made up of $n$ random variables $p_{t,1}(x_1), \ldots, p_{t,n}(x_n)$.

For UMDA on SUBSTRING, the bits left of the substring $1^{\alpha n}$ will always take the value of 1, because by the time UMDA has moved the substring one position to the right, the marginal probability corresponding to the leftmost 1-bit in the substring has converged to 1. On the basis of this intuitive idea, we are able to obtain the following formal result by theoretical analysis using Lemmas 1 and 2:

*Theorem 2:* Given any population sizes $N = \omega(n^{2+\beta}\log n)$, $M = \omega(n^{2+\beta}\log n)$ (where $\beta$ can be any positive constant) and $M = \gamma N$ ($\gamma \in (0,1)$ is some constant), the UMDA with truncation selection will spend no more than $\tau$ generations to find the global optimum of the SUBSTRING problem with a probability that is super-polynomially close to 0, where

$$\tau < \frac{n\Big(\ln\frac{eM}{N} - \ln(1-c)\Big)}{\ln(1-c) + \ln\left(\frac{N}{M}\right)} + 2n$$

and $c \in (\max\{0, 1-\frac{2M}{N}\}, 1-\frac{M}{N})$ is a constant. In other words, the number of fitness evaluations of the UMDA on SUBSTRING is $O(nN)$ with an overwhelming probability.

*Sketch of Proof:* The proof idea for this theorem is very similar to the one for Theorem 2 in our earlier paper [13]. For the sake of brevity, we only provide the sketch of the proof. It is essential to carry out two major steps following our approach described in [13]:

1) *Build an easy-to-analyse discrete dynamic system for the EDA. The idea is to de-randomise the EDA and build a deterministic dynamic system.*
2) *Analyse the deviations (errors) caused by de-randomisation. Note that EDAs are stochastic algorithms. Concretely, tail probability techniques, such as Chernoff bounds, can be used to bound the deviations.*

Recently, the above approach has also be used to study the time complexity of an improved version of UMDA on a deceptive bi-modal problem [12]. Before introducing the corresponding steps of the proof of this theorem, we have to provide some notations. Let the $i$-convergence time $T_i$ be defined as follows ($i \in \{1, \ldots, n\}$):

$$T_i := \min\{t; p_{t,i}(x_i^*) = 1\}.$$

Due to the stochastic nature of the UMDA, we know that $T_1, \ldots, T_n$ are all random variables. Moreover, we let $T_0 = 0$. Given the $t^{th}$ generation ($t \in \mathbb{N}^+$), we say that it belongs to the $i^{th}$ phase ($i \in \{1, \ldots, n\}$) if and only if $t$ satisfies $T_{i-1} < t \leq T_i$. To carry out a worst-case analysis, we define the deterministic updating rule for the $i^{th}$ phase as

$$\begin{aligned}\hat{\mathbf{P}}_{t+1}(\mathbf{x}^*) := \gamma_i(\hat{\mathbf{P}}_t(\mathbf{x}^*)) = \Big(&\hat{p}_{t,1}(x_1^*), \ldots, \hat{p}_{t,i-1}(x_{i-1}^*),\\ &[G\hat{p}_{t,i}(x_i^*)], R\hat{p}_{t,i+1}(x_{i+1}^*), \ldots, R\hat{p}_{t,n}(x_n^*)\Big), \quad (1)\end{aligned}$$

where $\hat{\mathbf{P}}_t(\mathbf{x}^*)$ represents the de-randomised probability vector of UMDA in the deterministic system, $\hat{p}_{t,1}(x_1^*), \ldots, \hat{p}_{t,n}(x_n^*)$ are its $n$ components, and $\mathbf{x}^* = (x_1^*, \ldots, x_n^*) = (1, \ldots, 1)$ is the global optimum of the SUBSTRING problem. In the above equation, we aim at dealing with three kinds of situations:

1) $j \in \{1, \ldots, i-1\}$ : In the deterministic system above, the de-randomised marginal probabilities $\hat{p}_{t,j}(x_j^*)$ have converged to 1, thus at the next generation they will not change.
2) $j = i$ : In the deterministic system above, the de-randomised marginal probability $\hat{p}_{t,i}(x_i^*)$ is converging, and we use the factor $G = (1-c)\frac{N}{M}$ to demonstrate the impact of selection pressure on this converging marginal probability[2], where $\frac{N}{M}$ represents the influence of the truncation selection operator which preserves the $M$ best individuals among the total $N$ individuals, and $c \in (\max\{0, 1-\frac{2M}{N}\}, 1-\frac{M}{N})$ is a constant parameter that controls the size of the

[2]"[ ]" in Eq. 1: given $a > 1$, $[a] = 1$; given $a \in (0,1)$, $[a] = a$.

**Algorithm 2** Univariate Marginal Distribution Algorithm with Truncation Selection.

```
1: for $i = 1$ to $n$ do
2:     $p_{0,i}(x_i) \leftarrow 0.5$.
3: end for
4: $\xi_1 \leftarrow N$ individuals are sampled according to the distribution $p_0(\mathbf{x}) = \prod_{i=1}^{n} p_{0,i}(x_i)$.
5: repeat
6:         $\xi_t^{(s)} \leftarrow$ The best $M$ individuals are selected from the $N$ individuals in $\xi_t$ $(N > M)$.
7:         $p_{t,i}(1) \leftarrow \sum_{\mathbf{x} \in \xi_t^{(s)}} c(x_i|1)/M, p_{t,i}(0) \leftarrow 1 - p_{t,i}(1) \quad (\forall i = 1, \dots, n)$.
8:         $\xi_{t+1} \leftarrow N$ individuals are sampled according to the distribution $p_t(\mathbf{x}) = \prod_{i=1}^{n} p_{t,i}(x_i)$.
9: until termination condition met.
```

deviation in the random sampling processes of the UMDA.

3) $j \in \{i+1, \dots, n\}$ : In our worst-case analysis, at the $i^{th}$ phase, we pessimistically consider that the $j^{th}$ marginal probability $p_{\cdot,j}(x_j^*)$ $(j \in \{i+1, \dots, n\})$ will not be affected by the selection pressure. Hence, we should take the genetic drift (the accumulation of random errors) into account. The genetic drift is possible to result in both increase and decrease of the marginal probability $p_{\cdot,j}(x_j^*)$, however, in our worst-case analysis, we only consider the case that $p_{\cdot,j}(x_j^*)$ is consistently reduced by genetic drift. To demonstrate the impact of genetic drift in the deterministic system, we utilise the factor $R = (1-\eta)(1-\eta')$, where $\eta < 1$ and $\eta' < 1$ are positive functions (of the problem size $n$) that controls the size of the deviation in Simple Random Sampling *without replacement*[3] when the truncation selection operator of the UMDA is dealing with different individuals with the same fitness (genetic drift).

On the other hand, since $\{\gamma_i\}_{i=1}^{n}$ de-randomises the whole optimisation process, $T_1, \dots, T_n$ in the above equation are no longer random variables. Consequently, we can define explicitly the deterministic system at the $i^{th}$ phase $(i = 1, \dots, n)$ as follows:

$$\forall \hat{T}_{i-1} < t \le \hat{T}_i : \quad \hat{\mathbf{P}}_t(\mathbf{x}^*) = \gamma_i^{t-\hat{T}_{i-1}}\left(\hat{\mathbf{P}}_{\hat{T}_{i-1}}(\mathbf{x}^*)\right),$$

where $\hat{T}_i$ is formally defined as follows:

$$\hat{T}_i := \min\{t; \hat{p}_{t,i}(x_i^*) = 1\}.$$

Given the definition of the deterministic system, it is not hard to obtain the following upper bound for $\hat{T}_i$ $(i \in \{1, \dots, n\})$:

$$\hat{T}_i \le \frac{i \ln \frac{eM}{N} - i \ln(1-c)}{\ln(1-c) + \ln\left(\frac{N}{M}\right)} + 2i.$$

So far we have finished the first step of the approach proposed in [13]. The rest part of the proof will concentrate on estimate the deviation between the deterministic system and the real optimisation process of the UMDA, and the goal is to show that $T_n \le \hat{T}_n$ holds with an overwhelming probability (a probability that is super-polynomially close to 1). A feasible idea leading to the goal is by proving that for any $i \in \{1, \dots, n\}$, $T_i \le \hat{T}_i$ holds with an overwhelming probability. The concrete implementation of the idea is using mathematical induction.

First we must prove that $\mathbb{P}\left(T_1 \le \hat{T}_1 \mid p_{0,1}(x_1^*) = \hat{p}_{0,1}(x_1^*)\right)$ is an overwhelming probability by linking the value of $T_1$ back to the values of random variables $p_{t,1}(x_1^*)$). In this case, we need to investigate the relation between $p_{t,1}(x_1^*)$ and $\hat{p}_{t,1}(x_1^*)$. More precisely, the probability that $p_{t,1}(x_1^*)$ is bounded from below by $\hat{p}_{t,1}(x_1^*)$ should be taken into account. This can be carried out by Chernoff bounds, since the random sampling processes of the UMDA (to generate new solutions) can be considered as repeated Bernoulli trials for a specific schemata. Noting that the outputs of Chernoff bounds are in the form of the reciprocal of an exponential function of the problem size $n$, we can expect to obtain an overwhelming probability for the event $p_{t,1}(x_1^*) \ge \hat{p}_{t,1}(x_1^*)$ in consideration, given that $N = \omega(n^{2+\beta} \log n)$ and $0 < t \le \hat{T}_1$. On the other hand, since $\hat{T}_1 = \Theta(1)$ is relatively small, given any positive integer $t$ that is no larger than $\hat{T}_1$ generations, the probability of the event $\forall t' \le t, t \in \mathbb{N}^+ : p_{t',1}(x_1^*) \ge \hat{p}_{t',1}(x_1^*)$ remains overwhelming (super-polynomially close to 1). As a consequence, it is not hard to reach the result that $\mathbb{P}\left(T_1 \le \hat{T}_1 \mid p_{0,1}(x_1^*) = \hat{p}_{0,1}(x_1^*)\right)$ is an overwhelming probability.

The rest part of the induction is similar to the case of the 1st phase. However, there are still two differences to concern. The first difference is that, when we are considering the $i^{th}$ phase, we cannot ignore the genetic drift for the $i^{th}$ marginal probability $p_{\cdot,i}(x_i^*)$ before the $i^{th}$ phase. The difficulty of coping with the genetic drift is that, we have to deal with the random errors brought by the selection processes: let us consider the case that the truncation selection operator is facing a number of *best but different* individuals (some different individuals are with the best fitness). The behaviour of the truncation selection in this case is actually Simple Random Sampling *without replacement*, and Lemma 2 can deal with this situation. By noting that $\hat{T}_1 < \dots < \hat{T}_n = O(n)$, we know that applying Lemma 2 at most $\hat{T}_n$ times

[3]The truncation selection operator cannot select an individual for more than one time.

will still lead to an overwhelming probability (given the conditions that the population sizes that $N = \omega(n^{2+\beta} \log n)$, $M = \omega(n^{2+\beta} \log n)$ and $M = \gamma N$), while the marginal probability $p_{\cdot,j}(x_j^*)$ $(j > i)$ will still be a constant at $(0, 1)$ with such overwhelming probability, no matter the value of the problem size $n$.

Another difference comes from the possibility of generating a substring $1^{\alpha n}$ during the genetic drift, which will stop the UMDA from performing under the "bound" of the deterministic system. Fortunately, the probability of the above event is exponentially close to 0, since we can apply the total probability theorem on the basis of the facts that $\alpha$ is a constant and the marginal probabilities under genetic drift are all constants at $(0, 1)$ within $O(n)$ generations with an overwhelming probability. ■

## V. CONCLUSION

In this paper, we construct the SUBSTRING problem on which a simple EDA called UMDA outperforms the (1+1) EA significantly. It is the first time that a comparison between an EA and an EDA has been carried out by means of a runtime analysis. Nonetheless, we are not trying to persuade readers that the UMDA is superior to the (1+1) EA. Instead, the investigation aims at showing a problem characteristic where UMDA can perform better, taking SUBSTRING as an example. As we have mentioned, for non-optimal search points of SUBSTRING, only a small subset of the variables influence the fitness, and the remaining variables are thus not subject to selection. However, for each variable, there is a region of the search space in which the variable influences the fitness. The marginal probability distribution in UMDA only changes for variables that are subject to selection, while remains constant for variables which have already converged (i.e., the corresponding marginal probability has reached 0 or 1), but the (1+1) EA will always flip the bits of search points and will thus "forget" the acquired information given no selection pressure. Furthermore, it is shown that on SUBSTRING, it is not helpful to lower the rate of forgetting by lowering the bitwise mutation rate of the (1+1) EA, because this also slows down the optimisation process and thus gives the algorithm more time to forget.

However, given the fact that the marginal probabilities of the UMDA can completely converge to 0 or 1, there is a natural question whether the promising performance of the UMDA on SUBSTRING only comes from this algorithmic feature. To answer this question, we can "relax" the marginal probabilities of the UMDA, and prevent the marginal probabilities from converging, e.g. by means of relaxation by margins [13] or Laplace correction. The corresponding investigation is left for future work.

In addition to the above issue, there are still a number of investigations to be carried out in the future. First, the analysis of the (1+1) EA on SUBSTRING should be generalised to population-based EAs, which will lead to fairer comparisons between EAs and UMDA, since UMDA employs population while (1+1) EA does not. Second, we need to extend the analysis of UMDAs to more complex EDAs where the dependencies among variables are taken into account.

## ACKNOWLEDGEMENTS

The authors would like to thank Ramón Sagarna for useful comments. This work was supported by EPSRC under grant no. EP/D052785/1, National Natural Science Foundation of China under grant no. 60533020, and the Fund for Foreign Scholars in University Research and Teaching Programs (China) under grant no. B07033.

## REFERENCES

[1] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Boston: Kluwer Academic Publishers, 2002.

[2] R. Sagarna, A. Arcuri, and X. Yao, "Estimation of distribution algorithms for testing object oriented software," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, 2007, pp. 438–444.

[3] M. Pelikan, K. Sastry, and D. E. Goldberg, "Scalability of the bayesian optimization algorithm," *International Journal of Approximate Reasoning*, vol. 31, no. 3, pp. 221–258, November 2002.

[4] P. S. Oliveto, J. He, and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, no. 1, pp. 100–106, 2007.

[5] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) Evolutionary Algorithm," *Theoretical Computer Science*, vol. 276, no. 1-2, pp. 51–81, 2002.

[6] J. He and X. Yao, "From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 5, pp. 495–511, 2002.

[7] J. He, X. Yao, and J. Li, "A comparative study of three evolutionary algorithms incorporating different amount of domain knowledge for node covering problems," *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 35, no. 2, pp. 266–271, 2005.

[8] P. S. Oliveto, J. He, and X. Yao, "Evolutionary algorithms and the vertex cover problem," in *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC'07)*, 2007, pp. 1870–1877.

[9] ——, "Analysis of population-based evolutionary algorithms for the vertex cover problem," in *Proceedings of IEEE World Congress on Computational Intelligence (WCCI'08), Hong Kong, June 1-6, 2008*, 2008, pp. 1563–1570.

[10] S. Droste, "A rigorous analysis of the compact genetic algorithm for linear functions," *Natural Computing*, vol. 5, no. 3, pp. 257–283, 2006.

[11] T. Chen, K. Tang, G. Chen, and X. Yao, "On the analysis of average time complexity of estimation of distribution algorithms," in *Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC'07)*, September 2007, pp. 453–460.

[12] ——, "Rigorous time complexity analysis of univariate marginal distribution algorithm with margins," in *Proceedings of 2009 IEEE Congress on Evolutionary Computation (CEC'09)*, May 2009.

[13] ——, "Analysis of computational time of simple estimation of distribution algorithms," submitted to *IEEE Trans. on Evolutionary Computation* on 26/11/2007.

[14] C. Gonzáles, "Contributions on theoretical aspects of estimation of distribution algorithms," Ph.D. dissertation, University of the Basque Country, November 2005.

[15] R. Motwani and P. Raghavan, *Randomized Algorithms.* Cambridge University Press, 1995.

[16] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963. [Online]. Available: http://www.jstor.org/stable/2282952

[17] R. J. Serfling, "Probability inequalities for the sum in sampling without replacement," *Annals of Statistics*, vol. 2, no. 1, pp. 39–48, 1974.

[18] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial Intelligence*, vol. 127, no. 1, pp. 57–85, March 2001.

[19] T. Chen, J. He, G. Sun, G. Chen, and X. Yao, “A new approach to analyzing average time complexity of population-based evolutionary algorithms on unimodal problems,” *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, accepted in December 2008.

[20] J. He and X. Yao, “A study of drift analysis for estimating computation time of evolutionary algorithms,” *Natural Computing*, vol. 3, no. 1, pp. 21–35, 2004.

[21] P. S. Oliveto and C. Witt, “Simplified drift analysis for proving lower bounds in evolutionary computation,” in *Proceedings of Parallel Problem Solving from Nature (PPSN’X)*, ser. LNCS, no. 5199, 2008, pp. 82–91.