Dynamic Evolutionary Optimisation: An Analysis of Frequency and Magnitude of Change[*]

Philipp Rohlfshagen
School of Computer Science
The University of Birmingham
Birmingham, United Kingdom
pzr@cs.bham.ac.uk

Per Kristian Lehre
School of Computer Science
The University of Birmingham
Birmingham, United Kingdom
pkl@cs.bham.ac.uk

Xin Yao
School of Computer Science
The University of Birmingham
Birmingham, United Kingdom
xin@cs.bham.ac.uk

ABSTRACT

In this paper, we rigorously analyse how the magnitude and frequency of change may affect the performance of the algorithm (1+1) EA_{dyn} on a set of artificially designed pseudo-Boolean functions, given a simple but well-defined dynamic framework. We demonstrate some counter-intuitive scenarios that allow us to gain a better understanding of how the dynamics of a function may affect the runtime of an algorithm. In particular, we present the function MAGNITUDE, where the time it takes for the (1+1) EA_{dyn} to relocate the global optimum is less than $n^2 \log n$ (i.e., efficient) with overwhelming probability if the magnitude of change is large. For small changes of magnitude, on the other hand, the expected time to relocate the global optimum is $e^{\Omega(n)}$ (i.e., highly inefficient). Similarly, the expected runtime of the (1+1) EA_{dyn} on the function BALANCE is $O(n^2)$ (efficient) for a high frequencies of change and $n^{\Omega(\sqrt{n})}$ (highly inefficient) for low frequencies of change. These results contribute towards a better understanding of dynamic optimisation problems in general and show how traditional analytical methods may be applied in the dynamic case.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; G.3 [**Probability and Statistics**]: Probabilistic Algorithms

General Terms

Algorithms, Theory

Keywords

Evolutionary algorithms, dynamic evolutionary computation, runtime analysis

[*]This work was supported by EPSRC grants no. EP/E058884/1 and EP/D052785/1.

1. INTRODUCTION

The field of *dynamic evolutionary optimisation* is concerned with the application of evolutionary algorithms (EAs) to the class of problems that change over time. The time-variant feature of *dynamic optimisation problems* (DOPs) poses many new challenges to the design of new algorithms, which, in turn, tend to be more complex than their counterparts used in the stationary domain. This has direct implications on their theoretical treatment and only very few results have been obtained so far on the expected runtimes of such algorithms. However, the complexity of the algorithms is not the only problem as there is also a distinct lack of generally accepted formal frameworks, making it difficult to unambiguously describe certain attributes of DOPs. Finally, it is not entirely clear how to analyse algorithms in the dynamic domain as concepts such as *expected first hitting time* or *success probability* may not apply, at least not without minor modifications. In this paper, we will discuss these issues and present a rigorous analysis of the (1+1) EA, adapted to the dynamic domain, on a set of simple dynamic functions. In particular, we look at the two most prominent features of DOPs, namely the *magnitude* and *frequency* of change.

There are numerous different types of DOPs but here we only consider the simplest case as it is by far the most commonly considered one in the literature. A DOP may be viewed as a time-variant series of *instances* $I(\cdot)$ of the same *problem* Π,

$$I(\pi = 0) \longrightarrow I(\pi = 1) \longrightarrow \ldots \longrightarrow I(\pi = m)$$

where the dynamics T determine how each instance differs from the next such that $I(\pi+1) = T(I(\pi), t)$. The time $t = 0, 1, 2, \ldots$ is discrete and advances with every call to the objective function, $f : X \times \mathbb{N}_0 \to \mathbb{R}$. The problem changes every τ generations[1] (i.e., the frequency of change is $1/\tau$). We denote τ as the *update period*, each of which is indexed by π:

$$\pi \leftarrow \begin{cases} \pi + 1 & \text{if } t \bmod \tau = 0 \\ \pi & \text{otherwise} \end{cases}$$

The field of dynamic evolutionary computation has gained significant momentum in recent years and a wide range of novel EAs have been proposed. The benchmarking of these algorithms is often carried out on readily available benchmark generators such as MOVING PEAKS due to Branke [2], DF1 due to Morrison [10] and XOR due to Yang ([17]; see section 2). However, the relative lack of theoretical results makes it difficult to fully assess the strengths and weaknesses of the individual algorithms. In other words, without a firm theoretical foundation, it is difficult to assess what these benchmarks actually test.

[1]In the more general case, τ is also a function of t (i.e., $\tau(t)$) although this is rarely considered in the literature.

This problem is somewhat worsened by the fact that most efforts are concerned exclusively with the development of new EAs and often ignore the inherent attributes and dynamics of the underlying problem. Subsequently, the majority of insights regarding the properties of evolutionary dynamic optimisation originate from the empirical observation of the behaviour of individual algorithms. In this paper, we attempt to address this issue and concentrate on two counter-intuitive cases that contradict commonly held beliefs. In doing so, we not only show that care has to be taken in making general claims about certain attributes of the dynamic domain, but we also attempt to illustrate how existing analytical techniques may be applied in the dynamic case.

The remainder of this paper is structured as follows: First, we fully describe the dynamic framework used in this paper in section 2, followed by a review of previous theoretical work in section 3. In section 4 we discuss the application of standard analytical tools and quantitive performance measures in the dynamic case. The second part of the paper presents two rigorous proofs for two counter-intuive dynamic scenarios: Section 5 deals with large and small magnitudes of change whereas section 6 is concerned with high and low frequencies of change. Finally, the paper in concluded in section 7 where some prospects for future work are discussed as well.

2. A DYNAMIC FRAMEWORK

The XOR benchmark [17, 18] imposes dynamics on any stationary pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ by means of a bit-wise *exclusive-or* operation that is applied to each search point $x \in \{0,1\}^n$ prior to each function evaluation. The dynamic equivalent of any stationary function is simply $f(x(t) \oplus m(\pi))$ where $\oplus$ is the *xor* operator. The vector $m(\pi) \in \{0,1\}^n$, which initially is equivalent to 0^n, is a binary mask, generated as $m(\pi) := m(\pi-1) \oplus p(\pi)$ where $p(\pi) \in \{0,1\}^n$ is a randomly created template that contains exactly $\lfloor \rho n \rfloor$ ones. In other words, for each period π, the mask m is altered using a randomly created template p that encapsulates the magnitude of change $\rho \in (0,1]$ (i.e., ρn is the actual number of bits inverted). The period index $\pi = \lceil t/\tau \rceil$ is determined by the duration $\tau > 0$ between changes. The parameters τ and ρ are usually kept constant throughout the execution of the algorithm. Finally, it should be noted that the majority of dynamics modelled with XOR are random but it is also possible to model both cyclical [19] and noisy cyclical [20] changes. In these cases, the mask m is defined non-randomly at each period to ensure the desired cycle is generated.

XOR was analysed formally by Tinós and Yang [16]: If we assume that the transformation of each encoding $x(t)$ by $m(\pi)$ yields a vector $z(t) = x(t) \oplus m(\pi)$, then it is possible to rewrite this expression as $z^n(t) = A(\pi)x^n(t)$ where $x \in \{0,1\}^n$ is normalised to $x^n(t) \in \{-1,1\}^n$ and where $A(\pi)$ is a linear transformation:

$$A(\pi) = \begin{bmatrix} A_1(\pi) & 0 & \dots & 0 \\ 0 & A_2(\pi) & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & A_n(\pi) \end{bmatrix}$$

where

$$A_i(\pi) = \begin{cases} 1 & \text{if } m_i(\pi) = 0 \\ -1 & \text{if } m_i(\pi) = 1 \end{cases}$$

for $i = 1, 2, \dots, n$. It follows that XOR does not actually alter the underlying function but instead *rotates* each search point x prior to each function evaluation. This approach clearly differs from more realistic dynamic problems where the actual fitness landscape varies structurally over time. However, XOR circumvents some of the numerous problems associated with defining precise dynamics in the combinatorial domain (see [14]) and thus provides a framework, which, albeit simple, defines the dynamics of the problem unambiguously. This property makes it possible to obtain accurate results and motivates the use of XOR in this paper.

We define two specifically designed functions in order to investigate how the performance of the (1+1) EA$_{\text{dyn}}$ is affected by the two most prominent characteristics of dynamic functions, namely the magnitude and frequency and change. These functions are pseudo-Boolean and are made dynamic using the XOR framework.

Algorithm 1 (1+1) EA$_{\text{dyn}}$ for time-variant problems.

```
Set $t := 0$ and choose $x(t) \in \{0,1\}^n$ uniformly at random.

while termination criterion not met do
    $x'(t) := x(t)$
    Flip each bit of $x'(t)$ with probability $1/n$.
    if $f(x'(t), t) \geq f(x(t), t)$ then
        $x(t+1) := x'(t)$.
    else
        $x(t+1) := x(t)$.
    end if
    $t := t + 1$.
end while
```

3. PREVIOUS THEORETICAL WORK

The (1+1) EA is the simplest EA that has the ability to perform global search by means of a mutation operator that uniformly inverts bits of x with some probability p_m. Here we assume that $p_m = 1/n$, where n is the length of x. The (1+1) EA$_{\text{dyn}}$ [3] is a simple adaptation of the (1+1) EA to the dynamic domain (see algorithm 1) that differs from the (1+1) EA only in the aspect that the objective function is called twice during each iteration to prevent the use of outdated fitness values. The (1+1) EA$_{\text{dyn}}$ has been analysed on several dynamic variants of the ONEMAX function, extending the work carried out previously on the stationary ONEMAX (e.g., [1, 12, 6]). The dynamic variants of the ONEMAX differ in their transitions from one period to the next, a property which may drastically affect the runtime of the algorithm. The publications that are discussed next did not make use of XOR although the framework is essentially identical. However, it should be noted that the runtime analysis is simplified if the dynamics are viewed as an additional mutation operator that acts directly on x (depending on the magnitude and frequency of change). This principle is used in the following discussion.

The first analysis of the (1+1) EA$_{\text{dyn}}$ on the dynamic ONEMAX is due to Stanhope and Daida [15] who considered dynamics where every τ function evaluations, solutions are mutated by m bits (like we do in section 5). Stanhope presented the transition probabilities of the (1+1) EA$_{\text{dyn}}$, successfully validated by a comparison to Monte-Carlo generated fitness distributions. Amongst other things, Stanhope showed that even small perturbations in the fitness function could have a significantly negative impact on the performance of the (1+1) EA$_{\text{dyn}}$. Droste [3, 4] considered two slightly different variants of the dynamic ONEMAX. In the first case, the search point is mutated by exactly one bit after each function evaluation and with probability p'. In the second and more general case, each bit of x may be inverted independently with probability p''. Droste showed that in the first case, the expected first hitting time of the (1+1) EA$_{\text{dyn}}$ is polynomial if and only if $p' = O(\log n/n)$. In the second case, Droste shows that the first hitting time changes from polynomial to super-polynomial at the critical growth rate of $p'' = \Theta(\log n/n^2)$.

4. ANALYSIS OF DYNAMIC FUNCTIONS

The analysis of dynamic functions can be very challenging as previous publications have demonstrated. Part of this challenge is, of course, due to the dynamics of the function which have to be considered in addition to the dynamics of the algorithm. However, another important aspect is the quantity by which the quality of an algorithm is to be judged. In the stationary case, the objective of an optimisation algorithm is to find the global optimum in as few steps (number of calls to the objective functions) as possible for any number of inputs n. In the dynamic case, on the other hand, the goal is to not only locate the global optimum but also to *track* said optimum as the search space changes over time [2]. In his work, Droste considers the expected first hitting time of the (1+1) EA_{dyn} in the continuously changing ONEMAX problem [4]. The expected first hitting time in this case is more accurately referred to as *expected temporal first hitting time* as we are interested in the time the *current* global optimum is first found (which, in turn, may be lost as soon as a change occurs). Droste mentions additional measures that may be taken into account such as the degree to which a found optimum is lost or the average distance to the nearest optimum over time [4, p 56].

There are numerous additional measures that could prove useful such as the time spent at the global optimum during each period or the highest frequency under which an algorithm can guarantee to locate the global optimum in each period with some probability p. Another interesting concept is the *expected n^{th} hitting time*: The first hitting time usually assumes a uniformly random distribution from which the initial search points are selected. In the dynamic case, however, once a change takes place, the algorithm has already spent τ steps optimising the function and is thus at a non-random point (e.g., the previous global optimum) when faced with a new instance of the problem. It is therefore reasonable to expect that the first hitting time may differ from subsequent ones and clearly, this depends on the magnitude and frequency of change. It is possible that the hitting times converge over time and it could be useful to obtain the expectation of the runtime over all periods for an algorithm on a given problem.

In this paper, we consider two different functions and for each proof we make use of two different quantitive measures. The first case, concerned with the magnitude of change, assumes that the frequency of change is sufficiently low to guarantee that the algorithm reaches either the local or global optimum during each period (the function is bi-modal) with high probability. We are subsequently interested in the time required to relocate the global optimum once it has been lost due to update of the function. We call this measure the *second hitting time*. In the case of the second function, which is concerned with the frequency of change, we analyse the first hitting time of the algorithm. This is similar to the work by Droste [3, 4], but in our case, once the global optimum has been found, it is no longer lost (i.e., the global optimum in this case is stationary). Although not estimated explicitly in this paper, it is also natural to consider the duration with which the algorithm resides at the optimum, a measure we call the *séjour time*. We now define these measures formally (we assume, without loss of generality, that we maximise the function).

Definition 1 (DYNAMIC RUNTIME). *Given a search space X and a dynamic fitness function $f : X \times \mathbb{N}_0 \to \mathbb{R}$, let $x(t)$, $t \geq 0$, be the current search point at iteration t of optimisation algorithm A on dynamic fitness function f. Then the* hitting times T_j *and the* séjour times S_i *of algorithm A on function f are defined as*

$$T_i := \min_t \{t \geq 0 \mid \forall y \in X, f(x(Q_i+t), Q_i+t) \geq f(y, Q_i+t)\},$$

$$S_i := \min_t \{t \geq 0 \mid \exists y \in X, f(x(Z_i+t), Z_i+t) < f(y, Z_i+t)\},$$

where $i \geq 1$ and

$$Q_k := \begin{cases} 0 & \textit{if } k = 1, \\ \sum_{l=1}^{k-1}(T_l + S_l) & \textit{otherwise} \end{cases}$$

$$Z_k := \begin{cases} T_1 & \textit{if } k = 1, \\ T_1 + \sum_{l=1}^{k-1}(S_l + T_{l+1}) & \textit{otherwise} \end{cases}$$

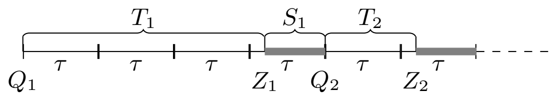


Figure 1: First hitting time T_1, séjour time S_1, and second hitting time T_2, where gray lines correspond to time intervals where the algorithm resides at the optimum.

Figure 1 illustrates these definitions. As mentioned previously, the objective of dynamic optimisation, at least in the cases considered here, is to *locate* and *track* the global optimum over time. To be considered efficient in the dynamic domain, it is necessary that the algorithm locates the optimum within reasonable (i.e., polynomial) time. Furthermore, the second and subsequent hitting times should not be larger than the first. Otherwise, a restart strategy should be favored over continous tracking of the optimum. In our case, the second hitting time is representative of the time to relocate the optimum after a change has occured. However, in the general case, it may be necessary to also consider subsequent hitting times in order to properly evaluate the effectiveness of the algorithm.

5. MAGNITUDE OF CHANGE

In dynamic evolutionary computation, the magnitude of change is generally regarded as the *relatedness* of two successive instances, $I(\pi)$ and $I(\pi+1)$. It is a common assumption that smaller magnitudes of change are easier to adapt to, primarily by "transferring knowledge from the past" [9, p 311] which may help to accelerate the rate of convergence after a change has occurred. However, there are two important aspects regarding these concepts that are often ignored. First, is not entirely clear how the magnitude of change may be measured. The most common approach is to use the genotypic distance between successive global optima but numerous other scenarios are possible. Fortunately, in XOR, this quantity is defined unambiguously by the parameter $\rho \in (0, 1]$ and simply refers to the number of bits a search point is rotated by. In our experiments, we denote the magnitude of change as θ which is an integer $0 < \theta \leq n$.

The second aspect is the concept of *knowledge* itself. Again, it is not entirely clear what information constitutes towards *knowledge from the past*; in most cases, such knowledge is restricted to the location of local and possibly global optima visited by the algorithm up to the present point in time. In the case of XOR, once the algorithm has located a local or global optimum, the magnitude of change determines exactly how far away a point will be rotated. It therefore seems reasonable to assume that smaller magnitudes of change are easier to recover from than larger ones and it is straightforward to show examples where this is indeed the case. However,

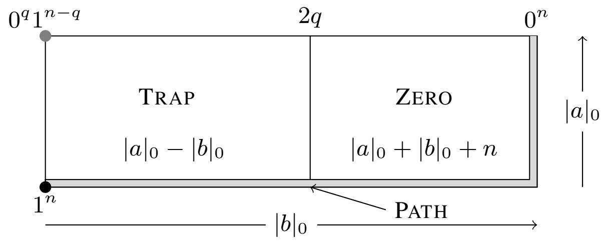


Figure 2: A view of MAGNITUDE in terms of the number of 0-bits in the prefix a and suffix b of $x = ab$.

the ease with which it is possible to specify such functions should not detract from other cases and here we show a function where small magnitudes of change are, in fact, much more difficult to adapt to than larger ones.

Definition 2 (MAGNITUDE). *Let $q := \epsilon n$ for any constant $0 < \epsilon \leq 1/3$. For any bitstring $x = ab$ of length n, define the function* MAGNITUDE $(x) := f(x_1 \cdots x_q, x_{q+1} \cdots x_n)$*, where*

$$f(a,b) := \begin{cases} 2n + i & \text{if } x = 1^i 0^{n-i} \in \text{PATH} \\ |a|_0 + |b|_0 + n & \text{if } x \in \text{ZERO} \\ |a|_0 - |b|_0 & \text{if } x \in \text{TRAP} \end{cases}$$

where the search space X is divided into three disjoint sets such that X = PATH $\cup$ ZERO $\cup$ TRAP: The set PATH $:= \{x \in \{0,1\}^n \mid x = 1^i 0^{n-i}, 0 < i \leq n\}$ *contains $n-1$ points which lead directly to 1^n. The set* ZERO $:= \{ab \in \{0,1\}^n \mid \ell(a) = q,\ 2q \leq |b|_0 \leq \ell(b),\ x \notin$ PATH$\}$ *contains all points with at least $2q$ 0-bits in the suffix that are not on the path. Finally, the set* TRAP $:= \{ab \in \{0,1\}^n \mid \ell(a) = q,\ 0 \leq |b|_0 < 2q,\ x \notin$ PATH$\}$ *is the set of all points with less than $2q$ 0-bits in the suffix that are not on the path.*

This stationary function is subsequently made dynamic using the XOR framework to yield a dynamic MAGNITUDE function. For the sake of clarity, we will continue to call this function simply MAGNITUDE and take it to refer to the dynamic version of the function. In this function, the local optimum, LOCAL $:= 0^q 1^{n-q}$, attracts almost all points with less than $2q$ 0-bits in the suffix (*trap* region). Almost all other points in X lead to the point 0^n where a path of $n-1$ points leads directly to GLOBAL $:= 1^n$. Figure 2 illustrates these concepts.

The proof idea for the runtime of the (1+1) EA$_{\text{dyn}}$ follows directly from the function definition and is based on two concepts: The behaviour of the algorithm during each update period (i.e., in time of stagnation) and the impact of the dynamics on the algorithm, given the algorithm is either at LOCAL or GLOBAL. We assume that the time between changes is sufficiently long for the algorithm to reach one of the two optima with high probability. Then, depending on the magnitude of change, different behaviours emerge. The initial search point may be in TRAP or ZERO. The probability to be on PATH is exceedingly small. If the algorithm starts in TRAP, it will be led away from the other regions of the search space towards the point LOCAL. Subsequently, if a small change occurs, the rotated search point will still be in the region TRAP and is hence attracted again to the local optimum. If, on the other hand, the initial search point is not in TRAP, the algorithm is led to the beginning of the path which leads directly to GLOBAL. The situation at GLOBAL is similar to the one at LOCAL. If the magnitude of change is small, the search point will be rotated into the TRAP region. If the magnitude of change is large, on the other hand, the search point will *jump* across the trap into ZERO or PATH. Similarly, if the algorithm is at LOCAL and a large change takes place, the search point is rotated beyond the boundary of the TRAP region.

In the following, we consider a magnitude of change to be *small* when $1 \leq \theta \leq q - cn$ for any constant c, $0 < c < \epsilon$, and a magnitude of change to be *large* when $3q \leq \theta \leq n$.

Proposition 1. *Assume a small magnitude of change, $1 \leq \theta \leq q - cn$, and an update period of $\tau = n^2 \log n$ and that at some point during period π, the search point x is at* LOCAL*. Then the probability that (1+1) EA$_{dyn}$ escapes* TRAP *before the end of period $\pi + 1$ is exponentially small.*

PROOF. Let x be any search point during period $\pi+1$. We make use of the Hamming distance between two points in the search space, denoted by $d(x,y)$. If the search point x has not yet escaped TRAP, then $d(x, \text{LOCAL}) \leq \theta \leq q - cn$, as the initial distance to LOCAL in period $\pi+1$ is θ, and the distance to LOCAL in TRAP decreases monotonically with the fitness. For any point z in PATH, the Hamming distance to LOCAL is lower bounded by $d(\text{LOCAL}, z) \geq q$. The triangle inequality implies that $d(z,x) + d(x, \text{LOCAL}) \geq d(z, \text{LOCAL})$, which implies $d(z,x) \geq q - (q - cn) = nc$. Analogously, by noting that the Hamming distance from any point z in ZERO to LOCAL is lower bounded by $d(\text{LOCAL}, z) \geq 2q$, one obtains $d(z,x) \geq q + nc$. Consequently, in order to escape TRAP in any given iteration, it is necessary to flip at least nc bits simultaneously. The probability that this happens in at most 2τ steps is by union bound less than $2\tau \cdot \binom{n}{nc} \cdot n^{-cn} = e^{-\Omega(n)}$. $\square$

Proposition 2. *For any constant $c > 0$, if (1+1) EA$_{dyn}$ remains in* TRAP *for a period of at least $\tau \geq cn^2 \log n$ iterations, then the algorithm will find* LOCAL *before the end of that period with probability $1 - e^{-\Omega(n)}$.*

PROOF. Following the well know arguments for ONEMAX [5], LOCAL is found starting from any point in TRAP within expected time $c'n \log n$ iterations, for some constant c'. Hence, by Markov's inequality, the probability that LOCAL is not found within any phase of $2c'n \log n$ iterations is less than $1/2$. Hence, the probability that LOCAL is not found within $\tau / 2c'n \log n = \Omega(n)$ such phases is $e^{-\Omega(n)}$. $\square$

Proposition 3. *If (1+1) EA$_{dyn}$ visits* GLOBAL *some time during period π, where the update period is $\tau = n^2 \log n$, and the magnitude of change is $1 \leq \theta \leq q - cn$, then there is a constant probability that the algorithm reaches* LOCAL *in period $\pi + 1$.*

PROOF. We first prove that there is an exponentially small probability that the search point enters ZERO during period $\pi + 1$. All points on PATH have higher fitness than any point in ZERO, so if the search point enters PATH, then the search point remains outside ZERO for the rest of the period. The first search point of period $\pi + 1$ has less than $q - nc$ 0-bits in the suffix, so the first search point is outside ZERO. If the search point is outside both ZERO and PATH, then it must be in TRAP. Inside TRAP, the fitness of any consecutive search point $x = ab$ is $|a|_0 - |b|_0 \geq q - nc$, so the number of 0-bits in the suffix is bounded from above by $|b|_0 \leq |a|_0 + q - nc \leq 2q - nc$. To reach ZERO, it is therefore necessary to flip at least nc 1-bits in the suffix simultaneously . The probability of not reaching ZERO within τ iterations is therefore at least $1 - \tau \cdot \binom{n}{nc} \cdot n^{-nc} > 1 - \tau/(nc)! = 1 - e^{-\Omega(n)}$.

We then show that there is a constant probability that the search point remains in TRAP during period $\pi + 1$. With probability at least q/n, the first search point during period $\pi + 1$ has at least one

0-bit in the prefix, and the search point is outside PATH. In the following, we assume that this bit is not flipped back to a 1-bit during $k \cdot (n-1)$ iterations, where k is a constant that will be specified later. The probability of this event is at least $(1-1/n)^{k\cdot(n-1)} \geq 1/e^k$. As long as the fitness is less than $q/4$ in TRAP, the probability of increasing the fitness in any iteration is at least $3q/4en$. The fitness in the initial iteration is at least $-|b|_0 > -q$, hence the expected time until the fitness is at least $q/4$ is no more than $(5q/4) \cdot (4en/3q) = 5en/3$. By Markov's inequality, the probability that the algorithm has not reached a fitness of at least $q/4$ in $k \cdot (n-1)$ iterations, where $k := 5e$ is less than $1/2$ for $n \geq 3$. Hence, the unconditional probability that the current search point in iteration $k\cdot(n-1)$ is in TRAP with fitness at least $q/4$ is at least $(1-e^{-\Omega(n)})/2e^k = \Omega(1)$. If the search point is in TRAP with fitness at least $|a|_0 - |b|_0 \geq q/4$, then the number of 0-bits in the prefix is at least $|a|_0 \geq q/4$. In order to reach PATH in this situation, it is necessary to flip at least $q/4$ 0-bits in the prefix, an event that happens with probability $\binom{n}{q/4} \cdot n^{-q/4} < e^{-\Omega(n)}$ in any given step. The probability that the algorithm escapes TRAP before the end of the period is then by union bound less than $\tau \cdot e^{-\Omega(n)} = e^{-\Omega(n)}$.

Conditional on the event that the current search point remains in TRAP during period $\pi+1$, Proposition 2 implies that the algorithm will find LOCAL with probability $1-e^{-\Omega(n)}$ before the end of the period. The probability that all of these events occur, and hence that LOCAL is found before the end of period $\pi+1$ is bounded below by a constant. □

Proposition 4. *If period π starts with the current search point outside* TRAP *and the update period is $\tau \geq n^2 \log n$, then the expected time until (1+1) EA$_{dyn}$ reaches* GLOBAL *is $O(n^2)$. Furthermore, the probability that the algorithm does not find* GLOBAL *within $n^2 \log n$ steps is exponentially small.*

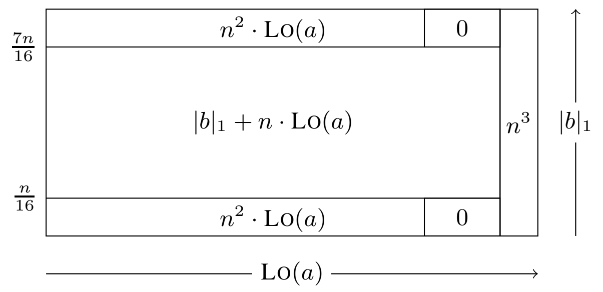
PROOF. This proof can make use of well established results as the regions ZERO and PATH correspond together precisely to the function SPI [8]. First, it is easy to show that the algorithm can not enter TRAP as the lowest fitness of any point outside TRAP is $n+2q$ and the highest fitness of any point in TRAP is, by definition, q. The period is divided into two phases, each of length $\tau/2$ iterations. Phase 1 is considered successful if PATH is reached before the end of the phase. Phase 2 is considered successful if GLOBAL is reached before the end of the phase.

In Phase 1, if the current search point is in ZERO, then the probability to get closer to 0^n, given a distance of i, is at least $((n-i)/n)(1-1/n)^{n-1}$. Substituting $1/e < (1-1/n)^{n-1}$, the expected time to reach 0^n or PATH, starting from any position in ZERO, is

$$E[T] \leq \sum_{i=0}^{n-1} \frac{en}{n-i} = en \sum_{i=1}^{n} \frac{1}{i} < en(\ln(n)+1).$$

By Markov's inequality, the probability that 0^n or PATH is not found within $2en(\ln(n)+1)$ iterations is less than $1/2$. Dividing the first phase of $\tau/2$ iterations into sub-phases, each of length $2en(\ln(n)+1)$, the probability that 0^n or PATH is not found within $\tau/4en(\ln(n)+1) = \Omega(n)$ sub-phases, i.e. that Phase 1 is unsuccessful, is $e^{-\Omega(n)}$.

In Phase 2, we assume that the current search point has reached PATH. The probability of progressing one further step along the path is bounded from below by $(1/n)(1-1/n)^{n-1} \geq 1/(en)$, similarly to that of LEADINGONES [5], that is optimised in expected time $E[T] \leq \sum_{i=0}^{n-1} en = en^2$. It follows that the hitting time is $O(n^2)$. Following arguments in [5], it can be shown that there exists a constant c' such that the probability that the optimum is not found within $c'n^2$ iterations is exponentially small. Hence for sufficiently large n, the probability that GLOBAL is not found within $\tau/2 \geq (n^2 \log n)/2$ iterations, i.e. that Phase 2 is unsuccessful, is exponentially small. □



$\frac{7n}{16}$
$n^2 \cdot \mathrm{LO}(a)$
0
$|b|_1 + n \cdot \mathrm{LO}(a)$
n^3
$|b|_1$
$\frac{n}{16}$
$n^2 \cdot \mathrm{LO}(a)$
0
$\mathrm{LO}(a)$

Figure 3: Fitness of a search point ab on fitness function BALANCE **for different values of leading 1-bits in the prefix a and 1-bits in the suffix b.**

Theorem 1. *The second hitting time of (1+1) EA$_{dyn}$ on function* MAGNITUDE *with update period $\tau \geq n^2 \log n$ and magnitude of change $1 \leq \theta \leq q - cn$, is $\mathbf{E}[T_2] = e^{\Omega(n)}$. If the update period is $\tau \geq n^2 \log n$ and the magnitude of change is $3q \leq \theta \leq n$, then the second hitting time is $T_2 < n^2 \log n$ with probability $1 - e^{-\Omega(n)}$.*

PROOF. Assume first that $1 \leq \theta \leq q - cn$ (*small* magnitude of change). By Proposition 1, the probability of escaping TRAP during a given period that starts with the current search point at LOCAL is $e^{-\Omega(n)}$. Furthermore, by Proposition 2, the probability of re-locating LOCAL before the end of the period is $1 - e^{-\Omega(n)}$. Hence, the expected number of iterations needed to escape TRAP is lower bounded by $\mathbf{E}[T \mid \mathrm{LOCAL}] \geq \tau e^{\Omega(n)}$. Assume that the current search point is at GLOBAL. By Proposition 3, the probability of reaching LOCAL starting from GLOBAL is at least $\mathbf{Pr}[\mathrm{LOCAL}] = \Omega(1)$. Hence, the unconditional expected time to re-locate GLOBAL, is at least $\mathbf{E}[T_2] \geq \mathbf{Pr}[\mathrm{LOCAL}] \cdot \mathbf{E}[T \mid \mathrm{LOCAL}] = e^{\Omega(n)}$.

Now we assume that $3q \leq \theta \leq n$ (*large* magnitude of change) and that the current search point is at GLOBAL. A large change will invert at least $3q$ 1-bits in x, which ensures the search point can not be located within TRAP and will either enter ZERO or PATH. The time it takes to reach GLOBAL is $O(n^2)$ as shown in Proposition 4. Finally, it follows from Proposition 4 that the probability to find GLOBAL within τ steps is exponentially large. □

6. FREQUENCY OF CHANGE

In this section, we will study how the frequency of change may affect the difficulty of a DOP. The frequency of change determines how often the problem changes and it seems intuitive to assume that a high frequency of change makes a problem more difficult for an algorithm to solve as less time is available at each period to reach the new global optimum. However, even if the global optimum is stationary (but the remainder of the search space is subject to consistent variations), as it is the case for the function considered in this section, a high frequency of change introduces considerably more uncertainty than a low frequency of change and could therefore still be expected to be more challenging. Here we will show that this assumption is not entirely correct. In particular, we will describe a dynamic optimisation problem called BALANCE which is hard at low frequencies, and easy at high frequencies.

Definition 3 (BALANCE). *For any bitstring x of length n, define* BALANCE $(x) := f(x_1 \cdots x_{n/2}, x_{n/2+1} \cdots x_n)$*, where*

$$f(a,b) := \begin{cases} n^3 & \textit{if} \quad \textsc{Lo}(a) = n/2, \textit{ else} \\ |b|_1 + n \cdot \textsc{Lo}(a) & \textit{if} \quad \frac{n}{16} < |b|_1 < \frac{7n}{16}, \textit{ else} \\ n^2 \cdot \textsc{Lo}(a) & \textit{if} \quad |a|_0 > \sqrt{n}, \textit{ else} \\ 0 & \textit{otherwise,} \end{cases}$$

where $\textsc{Lo}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$.

As shown in Figure 3, for each search point $x = ab$, we associate the *prefix* a of length $n/2$ with the horizontal axis, and the the *suffix* b of length $n/2$ with the vertical axis. The position of a search point x along the horizontal axis is given by the number of leading 1-bits in the prefix, and the position along the vertical axis is given by the number of 1-bits in the suffix. For the majority of search points, the dominating term in the fitness function is the number of leading 1-bits in the prefix. A globally optimal search point is obtained when the number of leading 1-bits in the prefix reaches the maximal value of $n/2$, i.e. the rightmost region in Figure 3. However, before reaching a globally optimal search point, the algorithm may reach one of the two trap regions that correspond to all non-optimal search points that have more than $7n/16$, or less than $n/16$ 1-bits in the suffix, i.e. the upper and lower regions in Figure 3. Only globally optimal search points have higher fitness than the search points in the trap region, which are separated from the globally optimal search points by two regions of fitness 0. The shortest distance from within the trap regions to a global optimum is $\sqrt{n}$.

We will consider cyclical dynamics that differ from the dynamics in the previous section. Following the framework outlined in Section 2, the mask m is defined as a function of the period index π as

$$m(\pi) := \begin{cases} 0^{n/2} \cdot 0^{n/2} & \text{if} \quad \pi \bmod 2 = 0, \text{ and} \\ 0^{n/2} \cdot 1^{n/2} & \text{otherwise.} \end{cases}$$

Hence, only the suffix of the point x is affected, and the magnitude of change is equivalent to $n/2$. The dynamical component of the fitness function corresponds to the vertical position. The fitness increases along the vertical dimension when the period index π is even, and the fitness decreases along the vertical dimension when the period index π is odd.

The proof idea is to show that the algorithm will balance along the centre of the vertical axis when the frequency of change is high, while the algorithm is likely to fall into one of the trap regions when the frequency of change is sufficiently low. This can be proved by analysing the horizontal and vertical *drift*. Informally, the drift of a search point is the distance the search point moves per iteration. The horizontal drift corresponds to the change in number of leading 1-bits in the prefix, and the vertical drift corresponds to the change in number of 1-bits in the suffix. As long as the trap region has not been reached, the position along the vertical axis can be changed by flipping any of at least $n/16$ bits, and no other bits. In contrast, in order to reduce the distance to the optimum along the horizontal axis, it is necessary to flip the single left-most 0-bit, an event that happens with much lower probability. Therefore, the vertical drift is much larger than the horizontal drift. If the frequency of change is sufficiently low, then the current search point will have enough time to reach one of the trap regions before the optimum is found. On the other hand, if the frequency of change is sufficiently high, then the search point will not have time to reach the trap region during one period. In the following period, the vertical drift will be in the opposite direction, and the vertical displacement of the search point is off-set. These informal ideas can be turned into a rigorous analysis using the simplified drift theorem.

Theorem 2 (SIMPLIFIED DRIFT THEOREM [13]). *Let X_t, $t \geq 0$, be the random variables describing a Markov process over the state space $S := \{0, 1, ..., N\}$, and denote $\Delta_t(i) := (X_{t+1} - X_t \mid X_t = i)$ 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. $\mathbf{E}[\Delta_t(i)] \geq \beta$ *for $a < i < b$, and*
2. $\mathbf{Pr}[\Delta_t(i) = -j] \leq 1/(1+\delta)^{j-r}$ *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

$$\mathbf{Pr}\left[T^* \leq 2^{c^*(b-a)}\right] = 2^{-\Omega(b-a)}.$$

Proposition 5. *Given a period of $\tau = 2$, the probability that within 2^{cn} iterations, the current search point of (1+1) EA$_{dyn}$ on* BALANCE *has a suffix b with $|b|_1 < n/16$ or $|b|_1 > 7n/16$ is exponentially small $e^{-\Omega(n)}$, where c is a constant.*

PROOF. Theorem 2 will be used to bound the probability that within 2^{cn} iterations, the suffix has less than $n/16$ number of 1-bits. The probability that the number of 1-bits in the suffix exceeds $7n/16$ can be proved analogously. We consider the interval $[n/16, n/8]$ over the state space $S := \{0, 1, ..., n/2\}$, and for each $t \geq 0$ define X_t to be the number of 1-bits in the suffix of the current search point in iteration $2t$ of the algorithm. Hence $X_{t+1} = X_t + \Delta_{2t} + \Delta_{2t+1}$, where Δ_{2t} and Δ_{2t+1} is the drift in iteration $2t$ and $2t+1$ of the algorithm. By Chernoff bounds [11], the probability that the number of 1-bits in the suffix of the initial search point is less than $n/8$ is $e^{-\Omega(n)}$. Hence, the conditional event $X_0 \geq n/8$ in the definition of random variable T^* holds with overwhelming probability.

In even time-steps $2t$ of the algorithm, we optimistically assume that the number of 1-bits does not increase. The expected drift in iteration $2t$ is therefore at least

$$\begin{aligned} \mathbf{E}[\Delta_{2t}] &\geq -\sum_{j=0}^{n/8} j \cdot \binom{n/8}{j} \cdot n^{-j} \\ &\geq -\sum_{j=0}^{\infty} j \cdot \frac{(n/8)^j}{j! n^j} \geq -\frac{e^{1/8}}{8}. \end{aligned}$$

In odd time-steps $2t+1$ of the algorithm, we distinguish between the case when $\textsc{Lo}(a) < \textsc{Lo}(a')$, and the case when $\textsc{Lo}(a) = \textsc{Lo}(a')$, where a and a' correspond to the prefixes of the current and mutated search point. Denote the event $\textsc{Lo}(a) < \textsc{Lo}(a')$ by $\mathcal{L}$. When event $\mathcal{L}$ occurs, the mutated search point is always accepted. Hence, if the number of 1-bits in the suffix of the current search point is $i < n/8$, the expected drift is

$$\mathbf{E}[\Delta_{2t+1} \mid \mathcal{L}] \geq \frac{n/2 - i}{n} - \frac{i}{n} = \frac{n/2 - 2i}{n} \geq 1/4.$$

In the second case, the mutated search point will not be accepted if the number of 1-bits decreases. To increase the number of 1-bits, it suffices to flip one of the at least $7n/8$ 0-bits in the suffix, and no other bits. Hence, the expected drift in this case is bounded by $\mathbf{E}\left[\Delta_{2t+1} \mid \overline{\mathcal{L}}\right] \geq 7/(e8) > 1/4$.

The expected drift during both iterations is then bounded from below by $\mathbf{E}[\Delta] \geq 1/4 - e^{1/8}/8 \geq 1/10$, so the first condition of Theorem 2 holds with $\beta := 1/10$.

Finally, to estimate the probability of reducing the number of 1-bits by j, one can optimistically assume that during two iterations, the EA will never flip the same bit position twice. Hence, the probability of flipping j 1-bits during two steps is not higher than the probability of flipping j bits out of $2n$ bits. Hence,

$$\begin{aligned}\mathbf{Pr}[\Delta = -j] &\leq \binom{2n}{j} \cdot \frac{1}{n^j} \leq \frac{2^j}{j!} \cdot \frac{n^j}{n^j} \\ &\leq \frac{2^j}{(2 \cdot 2)^{j-3} \cdot 3!} \leq \frac{1}{2^{j-6}},\end{aligned}$$

and the second condition of Theorem 2 holds with $\delta := 1$ and $r := 6$, which concludes the proof. □

Proposition 6. *Given a period of $\tau > 40n$, with $n > 100$, the probability that within $40n$ iterations, the current search point of (1+1) EA$_{dyn}$ on* BALANCE *has a suffix b with $|b|_1 < n/16$ or $|b|_1 > 7n/16$ is at least $1/4$.*

PROOF. We obtain the bound on the probability by applying Markov's inequality to the time T it takes to obtain a search point with less than $n/16$ 1-bits during an odd time cycle, as the reciprocal can be proved analogously. The expectation of T is bounded using the polynomial drift theorem [7], where the current state of the Markov chain is defined as $\max\{0, i - n/16\}$, where i is the number of 1-bits in the suffix of the current search point.

The expected drift conditional on the event $\mathcal{L}$ that the number of leading 1-bits in the prefix increases can be bounded analogously as in the proof of Proposition 5. Given $i > n/16$ number of 1-bits, the negative drift is bounded by $\mathbf{E}[\Delta \mid \mathcal{L}] > -((n/2 - i)/n - i/n) > -3/8$.

The expected drift conditional on the complementary event $\overline{\mathcal{L}}$ that the number of leading 1-bits in the prefix is constant, is at least $\mathbf{E}\left[\Delta \mid \overline{\mathcal{L}}\right] > (n/16)/(en) > 1/48$, because the number of 1-bits can be decreased by mutating one 1-bit and no other bits, and the number of 1-bits cannot increase.

The unconditional expected drift can be bounded by noting that the probability of event $\mathcal{L}$ is no more than $1/n$, giving $\mathbf{E}[\Delta] > (1 - 1/n)/48 - 3/8n$, which for n larger than 100 is greater than $1/80$. By the polynomial drift theorem, the expected time to remove $B < 7n/16 - n/16 = 3n/8$ 1-bits, is $\mathbf{E}[T] = B/\mathbf{E}[\Delta] < (3n/8)/(1/80) = 30n$ iterations. Hence, by Markov's inequality, the probability that it takes longer than $40n$ iterations to acquire a search point with less than $n/16$ 1-bits, is less than $3/4$. □

Theorem 3. *The expected first hitting time of (1+1) EA$_{dyn}$ on* BALANCE *with update time τ is*

$$\mathbf{E}[T] = \begin{cases} n^{\Omega(\sqrt{n})} & \text{if } \tau > 40n, \text{ and} \\ O(n^2) & \text{if } \tau = 2. \end{cases}$$

PROOF. We first lower bound the expected runtime when $\tau > 40n$. We only consider the runs where at least one of the search points within the first $40n$ iterations has a suffix b where $|b|_1 < n/16$ or $|b|_1 > 7n/16$. By Proposition 6, this event happens with probability $1/4$. Using ideas similar to those in the proof of Theorem 17 in [5], with probability $1 - o(1)$, the number of 0-bits in the prefix in the first $40n$ iterations is at least $n/8 > \sqrt{n}$. Since we are proving a lower bound, we optimistically assume that if either of these two events do not happen, then the optimum will be found in 0 iterations. If both of these events happen, then by the definition of the fitness function, the prefix of the current search point will remain on the form $|a|_0 > \sqrt{n}$ as long as the optimum has not been found. Hence, in order to reach the optimum, it is necessary to flip at least $\sqrt{n}$ bits simultaneously. The expected time until this event happens is at least $n^{\sqrt{n}}$, hence the unconditional runtime when $\tau > 40n$ is at least $(1/4) \cdot (1 - o(1)) \cdot n^{\sqrt{n}} = n^{\Omega(\sqrt{n})}$.

We now upper bound the expected runtime when $\tau = 2$. Following the ideas in the proof of Theorem 17 in [5], there exists a constant c such that the probability that (1+1) EA$_{\text{dyn}}$ has found the optimum of BALANCE within cn^2 iterations, conditional on the event that the suffix of the current search point satisfies $\frac{n}{16} < |y|_1 < \frac{7n}{16}$, is lower bounded by $1 - e^{-\Omega(n)}$. Hence, by Proposition 5, the unconditional probability that the optimum has been found within cn^2 iterations is also $1 - e^{-\Omega(n)}$. If the optimum has not been found within cn^2 iterations, then (1+1) EA$_{\text{dyn}}$ will find a search point with $\text{LO}(x) = \sqrt{n} - 1$ within $O(n^2)$ generations, again following Theorem 17 in [5]. From this point, the optimum can be found by flipping $\sqrt{n}$ bits simultaneously. The expected time until this event happens is less than $en^{\sqrt{n}}$. Hence, the unconditional expected runtime of (1+1) EA$_{\text{dyn}}$ with $\tau = 2$ is $(1 - e^{-\Omega(n)}) \cdot cn^2 + e^{-\Omega(n)} \cdot en^{\sqrt{n}} = O(n^2)$. □

7. CONCLUSIONS AND FUTURE WORK

The number of contributions towards the field of *evolutionary dynamic optimisation* has risen dramatically in recent years and a wealth of novel evolutionary algorithms (EAs) have been suggested that attempt to track the global optimum of some dynamic function over time. However, theoretical results on the expected runtimes of these algorithms are almost non-existent and almost all findings are based exclusively on empirical data. This imbalance may lead to incorrect assumptions about the dynamic domain and theoretical results are required to confirm the empirical evidence. In this paper, we attempt to take a step in that direction. The contributions of this paper are threefold: First, we discuss quantitative measures that may be used to measure the quality of an algorithm in the dynamic domain. Second, we present two counter-intuitive scenarios that contradict commonly held beliefs and third, we show how existing analytical methods may be applied in the dynamic case.

We presented two rigorous proofs regarding the magnitude and frequency of change. It has been shown how the expected second hitting time of (1+1) EA$_{\text{dyn}}$ on the function MAGNITUDE is $\mathbf{E}[T_2] = e^{\Omega(n)}$ when the magnitude of change affects less than $q - cn$ bits (*small* magnitude of change), where q is linear in n and c is some constant. On the other hand, if the magnitude of change affects at least $3q$ bits (*large* magnitude of change), the second hitting time is $T_2 < n^2 \log n$ with probability $1 - e^{-\Omega(n)}$. In other words, the algorithm is efficient if the magnitude of change is large and highly inefficient if the magnitude of change is small. A similar counter-intuitive example is given with regard to the frequency of change. Here it is shown how the expected runtime of the (1+1) EA$_{\text{dyn}}$ on the function BALANCE is $\mathbf{E}[T_1] = O(n^2)$ given a very short update periods of $\tau = 2$. If the update period is $\tau > 40n$, on the other hand, the expected runtime is $\mathbf{E}[T_1] = n^{\Omega(\sqrt{n})}$. Similarly to before, the algorithm is efficient if the frequency of change is very high and inefficient if the frequency of change is sufficiently low.

The two functions considered in this paper represent interesting cases, not only because they illustrate counter-intuitive scenarios, but also because they highlight certain properties of the dynamic domain that are often ignored. In the case of MAGNITUDE, we have seen when it is beneficial to consider a *dynamic* approach: If the magnitude of change is large, it is beneficial to re-use the search point uncovered in the previous period. If the magnitude of change is small, on the other hand, a random-restart would be much more efficient because the algorithm may start outside the trap that exists in this function. In the second case, the function BALANCE demonstrates that dynamics, which do not affect the position of the global optimum, may still have a significant impact on the expected first hitting time of an algorithm. Such dynamics are frequently ignored in the literature as almost all attention is paid to the dynamic behaviour of the global optimum.

The functions and the dynamic framework considered in this paper are very simple and may not be representative of the majority of DOPs, especially those found in industrial settings. However, the XOR framework has been used frequently in the literature for benchmarking new algorithms, often using simple underlying functions such as the ONEMAX (e.g., [17, 18]). In the near future, we would like to consider the expected runtimes of population based EAs, first on functions similar to those considered here and eventually on more complicated ones. Furthermore, additional work is required to formulate more extensively the different quantitive measures that may be used to judge the quality of an algorithm.

8. REFERENCES

[1] T. Bäck. Optimal mutation rates in genetic search. In *Proceedings of the 5th international conference on genetic algorithms*, pages 2–8, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[2] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2002.

[3] S. Droste. Analysis of the (1+1) EA for a dynamically changing objective function. Technical Report CI-113/01, Universität Dortmund, 2001.

[4] S. Droste. Analysis of the (1+1) EA for a dynamically changing onemax-variant. In *Congress on Evolutionary Computation*, pages 55–60. IEEE Press, 2002.

[5] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) Evolutionary Algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[6] S. Droste, Jansen T., and Wegener I. A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.

[7] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

[8] T. Jansen and I. Wegener. Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2001.

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

[10] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer, Berlin, 2004.

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

[12] H. Mühlenbein. How genetic algorithms really work - i. mutation and hillclimbing. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 15–26. Elsevier, 1992.

[13] P. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. In *Proceedings of Parallel Problem Solving from Nature (PPSN'X)*, number 5199 in LNCS, pages 82–91, 2008.

[14] P. Rohlfshagen and X. Yao. Attributes of dynamic combinatorial optimisation. In *Proceedings of the 7th Int. Conference on Simulated Evolution and Learning*, number 5361 in LNCS, pages 442–451. Springer, 2008.

[15] S. A. Stanhope and J. M. Daida. (1+1) genetic algorithm fitness dynamics in a changing environments. In *Congress on Evolutionary Computation*, volume 3, pages 1851–1858. IEEE, 1999.

[16] R. Tinos and S. Yang. Continuous dynamic problem generators for evolutionary algorithms. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pages 236–243, 2007.

[17] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithms. In R. Sarker, R. Reynolds, H. Abbass, K.-C. Tan, R. McKay, D. Essam, and T. Gedeon, editors, *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, volume 3, pages 2246–2253, 2003.

[18] S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, volume 2, pages 1115–1122. ACM Press, 2005.

[19] S. Yang. Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2560–2567, 2005.

[20] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *IEEE Transactions on Evolutionary Computation*, 12(5):542–561, Oct. 2008.