Analysis of Randomised Search Heuristics for Dynamic Optimisation

Jansen, Thomas; Zarges, Christine

Published in:
Evolutionary Computation

DOI:
10.1162/EVCO_a_00164

Publication date:
2015

Citation for published version (APA):
Jansen, T., & Zarges, C. (2015). Analysis of Randomised Search Heuristics for Dynamic Optimisation. *Evolutionary Computation*, *23*(4), 513-541. https://doi.org/10.1162/EVCO_a_00164

Analysis of Randomised Search Heuristics for Dynamic Optimisation*

Thomas Jansen t.jansen@aber.ac.uk
Dep. of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, UK

Christine Zarges c.zarges@cs.bham.ac.uk
School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

Abstract

Dynamic optimisation is an area of application where randomised search heuristics like evolutionary algorithms and artificial immune systems are often successful. The theoretical foundation of this important topic suffers from a lack of a generally accepted analytical framework as well as a lack of widely accepted example problems. This article tackles both problems by discussing necessary conditions for useful and practically relevant theoretical analysis as well as introducing a concrete family of dynamic example problems that draws inspiration from a well-known static example problem and exhibits a bi-stable dynamic. After setting the stage this way the framework is made concrete by presenting the results of thorough theoretical and statistical analysis for mutation-based evolutionary algorithms and artificial immune systems.

Keywords

Dynamic optimisation problems, evolutionary algorithms, artificial immune systems, theory, fixed budget computations

1 Introduction

Optimisation problems are ubiquitous and not all optimisation problems have the property that they stay fixed while they are solved by some optimisation algorithm. Some change over time and if the change is sufficiently fast in comparison to the optimisation process the changing nature of the problem has to be taken into account. In these cases we speak of dynamic optimisation problems and are often confronted with the situation that no good problem-specific algorithms are available for solving them. In practice, in such situations often heuristic optimisers are used. There are many different randomised search heuristics that can be applied in this context, among them evolutionary algorithms and artificial immune systems.

Evolutionary algorithms have been successfully applied in dynamic optimisation as witnessed by books that are devoted to precisely this topic (Branke, 2002; Weicker, 2003; Morrison, 2004; Yang and Yao, 2013). The theoretical analysis, however, is lagging far behind. While it is common that the theoretical analysis of randomised search heuristics follows after their successful application, the last two decades have witnessed an immense development in the theory of randomised search heuristics for static optimisation (Neumann and Witt, 2010; Auger and Doerr, 2011; Jansen, 2013). This article contributes to the endeavour to carry over this success in the analysis of static optimisation to dynamic optimisation.

* A preliminary version of this work has been presented at the *Genetic and Evolutionary Computation Conference (GECCO)* in 2014 (Jansen and Zarges, 2014a).

1.1 Our Contribution

There are five main contributions of this article, three of them rather general and the other two rather concrete. The first general contribution is pointing out that the perspective of theoretical analysis of dynamic optimisation should change to adopt the fixed budget computations perspective, a paradigm shift that recently occurred in the analysis of static optimisation (Jansen and Zarges, 2012; Doerr et al., 2013; Jansen and Zarges, 2014b,c; Nallaperuma et al., 2014; Lengler and Spooner, 2015). The second general contribution is clearly pointing out that the rate of change of the dynamic optimisation problem and the speed of the execution platform that runs the (heuristic) optimiser are two different things that should not be confused. We discuss in Subsection 1.2 why this is important. The third general contribution is the presentation of an example function that, we hope, is sufficiently simple to attract attention in the further analytical study of randomised search heuristics but is sufficiently interesting to capture important properties of real dynamic optimisation problems. It is a bi-stable function that exhibits phases of stability and rapid change. Its definition is motivated by characteristics of practical problems in, e. g., pharmaceutical design (see (Tifenbach, 2013, page 126) for a discussion of this aspect). The first concrete contribution is the analysis of a class of mutation-based evolutionary algorithms and artificial immune systems on this bi-stable dynamic optimisation problem. This demonstrates that the analytical perspective of fixed budget computations and our new example problem both provide feasible settings for theoretical analysis. We choose to study not only evolutionary algorithms but also artificial immune systems because there is reason to believe that in situations of rapid change artificial immune systems may have an advantage over evolutionary algorithms (Jansen and Zarges, 2014c). The second contribution is the first in-depth analysis of a variant of a well-known artificial immune system that was suggested by earlier theoretical analysis of artificial immune systems in static optimisation (Jansen et al., 2011).

1.2 State of the Art in the Theoretical Analysis of Dynamic Optimisation

While dynamic optimisation is an important area of application for many randomised search heuristics the theoretical analysis lags behind even more than it does for static optimisation. Bu and Zheng (2010) and Nguyen et al. (2012) both point this out when discussing the state of the art. Both articles provide an overview of performance measures for dynamic optimisation, Nguyen et al. (2012) also provide an extensive overview of benchmark problems. While these benchmarks have value for empirical studies, they have not proven particularly useful and popular in theoretical studies. The same holds for the different performance measures that are discussed in both overview articles. Complex performance measures tend to elude theoretical analysis and, as a consequence, quite simplistic performance measures dominate in theoretical analyses of randomised search heuristics in dynamic optimisation. Alternatively, only very limited aspects of the algorithm are analysed theoretically and the major parts of the analysis are based on experiments (Stanhope and Daida, 1999).

When considering theoretical analyses we see that they are either based on very simple example functions, which are based on the most popular static example functions, or they consider very specific (and sometimes complicated) functions that are designed with a very specific purpose in mind. Typical instances of theoretical analysis that are based on extremely simple static benchmark functions include work by Droste (2002, 2003) who analysis a dynamic variant of ONEMAX, where the fitness of a search point equals the number of bits this string has in common with a target string.

He concentrates on the first hitting time of the optimum and the (1+1) evolutionary algorithm (EA), a very simple mutation-based evolutionary algorithm that has a population of size 1 and creates only one offspring in each generation. For such algorithms that perform only a very small number of function evaluations per generation (2 in this case) it is reasonable to assume that the fitness function does not change during a generation. Other examples of this kind of research include work by Rohlfshagen et al. (2009) and Kötzing and Molter (2012). In both articles very specific example functions are designed in order to prove a specific point. In the case of Kötzing and Molter (2012) the example function is derived from ONEMAX and the very specific change that is defined is used to make the difference in dealing with the speed of change between ant colony optimisers and evolutionary algorithms explicit. In the case of Rohlfshagen et al. (2009) the custom-design dynamic fitness function is designed to prove the point that sometimes dynamic functions can be easier to optimise than static variants, contrary to common belief and intuition. Another research direction aims at performing an analysis of the random process as Markov chain in the same way Vose and others established this for static functions (Vose, 1998). This has been done by Tinos and Yang (2010, 2013) but with very limited tangible theoretical results. Tinos and Yang (2014) follow a similar approach, consider a wide range of classes of dynamic optimisation problems and present a benchmark problem generator. In all three papers the most significant results are empirical, gained from experiments with example functions.

When the number of function evaluations per generation is larger than just 2 it becomes dubious if one can simply assume that the dynamic fitness function does not change during a generation. To the best of our knowledge Branke and Wang (2003) are the only ones to consider the scenario of change during a generation and provide a detailed analysis of a simple $(1, 2)$ evolution strategy, an algorithm that also performs only 2 function evaluations per generation, for this case. Other articles, among them work by Jansen and Schellbach (2005), Kötzing et al. (2015), Lissovoi and Witt (2013), Lissovoi and Witt (2014), Oliveto and Zarges (2013), and Oliveto and Zarges (2014), larger population size or offspring population size are taken into account but it is (sometimes implicitly) assumed that the fitness function does not change within a generation. This assumption becomes critical when the effects of the choice of the population size and of the offspring population size are studied. Increasing the size and consequently the number of function evaluations in a generation effectively means slowing down the rate of change in the dynamic objective function. It then becomes unclear if improved performance is actually due to the increased (offspring) population size or the slower rate of change.

What most articles also have in common is that they concentrate on the expected first hitting time of the global optimum (or similar measures). This is motivated by the fact that the analysis of the expected optimisation time for static optimisation is the most used and successful performance measure. The step from the analysis of the expected optimisation time to the analysis of the expected solution quality that is performed when using the fixed budget perspective has not yet been made in theoretical analyses of dynamic optimisation. It is worth noting that in empirical studies it is much more common to concentrate on the average solution quality (Bu and Zheng, 2010; Nguyen et al., 2012).

1.3 Organisation of the Article

Using the perspective of fixed budget computations alone is not sufficient to guarantee that the results of theoretical analyses are meaningful. It is also required that the con-

sidered algorithms are practically relevant, the range of parameters considered makes sense in practical settings, and that the considered dynamic problem is either relevant itself or exhibits properties that are believed to be relevant. We consider the aspect of the interplay between properties of the problem, parameter settings, and algorithmic properties in the next section and point out how this can be taken into account in analysis so that it is no longer overlooked as has happened in the past. In Section 3 we introduce and carefully motivate the bi-stable dynamic example problem. Section 4 introduces the classes of evolutionary algorithms and artificial immune systems we consider. For the artificial immune systems we consider a relatively new variant that in some sense hybridises artificial immune systems and evolutionary algorithms and exhibits improved performance in a number of circumstances. Section 5 contains our analysis where we present results for evolutionary algorithms and artificial immune systems for our benchmark problem in a wide variety of settings. We present theoretical results as well as thorough statistical studies of the results of experiments. This helps us to gain a deeper understanding and identify open problems. We summarise and show directions for future research in Section 6.

2 Analysing Randomised Search Heuristics on Dynamic Problems

A dynamic optimisation problem is one that changes over time. Formally, we model this by saying that the quality of a point in the search space, $x \in S$, is given by the function value $f(x,t)$ at time step $t \in \mathbb{N}_0$. Thus, a static optimisation problem can be described as a degenerated dynamic one where $f(x,t) = f(x,t')$ holds for all $t, t' \in \mathbb{N}_0$. It is important to note that the time steps are an important property of the problem and are as such independent of the means to solve this problem, in particular independent of the heuristic optimisation method we plan to use.

This independence has important consequences. It implies that the speed of change of the dynamic optimisation problem (expressed as discrete time steps $t \in \mathbb{N}_0$) is not related to the speed at which 'our' heuristic optimisation method is executed. E. g., if we employ a larger population in an evolutionary algorithm we should assume that the speed of change of the optimisation problem appears to be faster when measured in the number of generations. The reason is that the speed of change of the optimisation problem is unchanged but that each generation of the evolutionary algorithm now takes longer due to the larger population size. As we have pointed out in Section 1.2 this has often been overlooked in the past.

In the analysis of randomised search heuristics one usually considers one evaluation of the objective function f to be an atomic event. Usually, the performance of a randomised search heuristic is measured in the number of such function evaluations. This is true for the perspective of run time or optimisation time analysis (Jansen, 2013) as well as for the perspective of fixed budget computations where the computational budget is measured as the number of function evaluations (Jansen and Zarges, 2012).

We adopt this point of view and assume that evaluating the objective function once, i. e., computing $f(x,t)$ for any $x \in S$ and any $t \in \mathbb{N}_0$ can be carried out in one time step of the dynamic objective function f. We argue that this makes sense because the computation of the function value $f(x,t)$ is obviously connected to properties of the function. If one allowed the objective function to change faster it becomes difficult to see how a function is supposed to be optimisable because in the extreme case it can change arbitrarily during the time it takes to evaluate a single function value.

Clearly it may be possible to compute $f(x,t)$ in a time that is much faster than that required by a complete time step of the dynamic objective function. We characterise the

speed of the computation platform that executes the heuristic optimiser and computes function values by the number of function evaluations it can make in one time step of the dynamic objective function.

Definition 1. *Let $\sigma \in \mathbb{N}$ be the number of function evaluations that can be be carried out between time steps t and $t+1$. This number is independent of t and characterises the* speed *of the execution platform.*

Having defined the speed of an execution platform σ we can now investigate two different kinds of questions. On the one hand, we can investigate how the speed of the execution platform influences the performance of a fixed optimiser. We expect the performance to increase with increasing speed of the execution platform. Finding out how this happens can help to understand when it makes sense to invest in better hardware when being confronted with a dynamic optimisation problem. On the other hand, we can investigate how different optimisers or the same optimiser with different parameter settings perform on the same dynamic problem when executed on the same execution platform, i. e., on an execution platform with a fixed speed σ.

When considering optimisers that work in rounds (or generations), like evolutionary algorithms and artificial immune systems do, it is important to see if the dynamic optimisation problem can change within one such round. As we have pointed out in Section 1.2 almost all previous work assumes that this does not happen (with the exception of the analysis by Branke and Wang (2003)). In this article we also work under the assumption that no change happens within one round. This means that σ needs to be sufficiently large so that one round can be executed completely (or, the other way around, that the optimiser needs to be parameterised so that it performs at most σ function evaluations per round). We make the assumption that optimisers are unaware of the time steps and do not know when the dynamic optimisation problem may change. The execution platform makes sure that one round of an optimiser is carried out in one time step of the dynamic optimisation problem so that the values of search points do not change during one round. If the optimiser makes ρ function evaluations in each round, then the execution platform performs $\lfloor \sigma/\rho \rfloor$ rounds in one time step of the dynamic optimisation problem.

As mentioned in Section 1 we adopt the fixed budget perspective for our analysis. We believe it makes more sense to have a statement about the performance of an optimiser tackling a dynamic optimisation problem at any point of time instead of concentrating on one specific aspect like the first hitting time of a (potentially moving) global optimum. Instead of considering the function values, as it is common in fixed budget analyses (Jansen and Zarges, 2012), we consider the difference in function value to the optimal function value. This slight change of perspective allows for a somewhat more natural formulation of performance. Note that the following definition formalises the notion of efficiency roughly and only makes sense for objective functions with certain properties. It makes the implicit assumption that it is very easy to find search points with a distance of $\Theta(n)$ in function value to the optimal value and becomes increasingly difficult to improve over that.

Definition 2. *For $g, t \in \mathbb{N}_0$ let $f(x,t)$ denote the function value in time step t of the dynamic optimisation problem and let g denote the round (or generation) of an optimiser for f. Let $\rho(g) \in \mathbb{N}$ denote the number of function evaluations that the optimiser makes in round g, let $x_1^{(g)}$, $x_2^{(g)}$, ..., $x_\rho^{(g)}$ denote the search points that it evaluates in this round. Let OPT$(t) = \max\{f(x,t) \mid x \in S\}$ denote the optimal function value in time step t. We define the distance in function value to the optimal value in round g as $M(g) :=$*

$\min\left\{OPT(t) - f\left(x_i^{(g)}\right) \mid i \in \{1,2,\ldots,\rho\}\right\}$.

We say that the optimiser has perfect *performance in generation g if $M(g) = 0$ holds. We say that its performance is* good *in generation g if it does not perform perfectly but if $M(g) = O(1)$ holds. We say that its performance is* mediocre *in generation g if its performance is neither perfect nor good but if $M(g) = o(n)$ holds. We say that its performance is* bad *in generation g if $\mathrm{E}(M(g)) = \Theta(n)$ holds.*

Note that Definition 2 makes use of the distance $M(g)$ in most cases but considers $\mathrm{E}(M(g))$ in the case of bad performance. Since we consider randomised search heuristics (namely, evolutionary algorithms and artificial immune systems) $M(g)$ is a random variable. Therefore, we will make statements about performance that is better than bad by giving bounds on the probability for such a performance.

3 A Dynamic Example Problem

The example problem we introduce is inspired by the example problem ONEMAX, the most commonly studied example problem when analysing the performance of randomised search heuristics in static optimisation. Like ONEMAX, it has a very simple structure that facilitates analysis and understanding while having properties that are natural in some sense. It is a bi-stable function, i. e., it oscillates between two different global optima where it is stable for some time. In phases of change the change is rapid.

The function is a pseudo-Boolean function, i. e., it operates on bit strings. At any point in time t the function value is given as the number of bits where a bit string agrees with the current global optimum. We see that the function values are always integers between 0 and n and that n is the global maximum. This is precisely the same as a generalised ONEMAX where the unique global optimum is some fixed bit string. For our example function the two stable global optima are o and its bit-wise complement, $\overline{o}$, where the bit string $o \in \{0,1\}^n$ is a parameter of the function.

The length of the stable phases is also a parameter of the function, called $\tau \in \mathbb{N}$. The search point o is the unique global optimum for a duration of τ steps. After this stable phase the global optimum moves gradually in a random but orderly fashion towards its bit-wise complement, $\overline{o}$. Once $\overline{o}$ is the unique global optimum the function is stable again for τ time steps. After this stable phase the global optimum moves gradually back to o, structurally using the same path but avoiding repetition of any intermediate points. In the non-stable phases where the optimum moves we have it move by changing exactly one bit in one time step. We see that this implies that the example function has a period of length $\Pi = 2(\tau + n - 1)$, i. e., $f(x,t) = f(x, t + k \cdot \Pi)$ for all $k, t \in \mathbb{N}_0$ and all $x \in \{0,1\}^n$. We will use binary masks to define the transition formally in Definition 3 similarly to the way Yang and Yao (2005) have introduced binary masks to define dynamic problem generators.

In the transition where the global optimum moves from o to $\overline{o}$ the bits change their values in an order so that the bits equal to o always form one contiguous block. One could also consider a somewhat simpler variant of this function where the next bit to change its value is selected uniformly at random among the bits that have not yet changed their value. While for mutation-based evolutionary algorithms such a change in the dynamic problem is unimportant it can have consequences for other algorithms. Those algorithms potentially affected include evolutionary algorithms with k-point crossover and some artificial immune systems, notably the B cell algorithm. Since we consider the latter we consider the variant where unchanged bits form a contiguous block.

We define the example function that we call $\mathrm{BSO}_{o,\tau}$ (short for **B**i-**S**table **O**ptimisation problem) formally and precisely in the following. For this we make use of the well-known notation for the concatenation of letters. For a letter $b \in \{0,1\}$ and a length $i \in \mathbb{N}_0$ we define b^i as the concatenation of i copies of b, e. g., $1^4 = 1111$. For $i = 0$ we obtain the empty word, i. e., $b^0 = \varepsilon$. As usual, $b^1 = b$ holds. We allow the concatenation of such expressions, e. g., $1^2 0^3 1^0 0^1 1^4 = 1100001111$. For two bits $u, v \in \{0,1\}$ let $u \oplus v$ denote the exclusive OR of u and v, i. e., $u \oplus v = 0$ if $u = v$ and $u \oplus v = 1$ if $u \neq v$. For two bit strings $x, y \in \{0,1\}^n$ of equal length n let $x \oplus y$ denote the bit-wise exclusive OR of x and y, e. g., $0110 \oplus 1100 = 1010$. Finally, let $\mathrm{H}(x,y)$ denote the Hamming distance of x and y, i. e., $\mathrm{H}(x,y) = \sum_{i=0}^{n-1} |x[i] - y[i]|$. Note that $x[i]$ denotes the bit at position i in $x \in \{0,1\}^n$ and that the leftmost position is $x[0]$.

Definition 3. *For $n \in \mathbb{N}$, $\tau \in \mathbb{N}$ and $o \in \{0,1\}^n$ we define the bi-stable optimisation problem* $\mathrm{BSO}_{o,\tau}\colon \{0,1\}^n \times \mathbb{N}_0 \to \mathbb{N}_0$. *We define the cycle length $\Pi := 2(\tau + n - 1)$. For $t \in \mathbb{N}_0$ let $t' = t \bmod \Pi$ denote the time index in the current period. Let $x^*(t) \in \{0,1\}^n$ be the unique global optimum at time step t. We define $x^*(t)$ with the help of transition masks which we will define later.*

$$x^*(t) := \begin{cases} o & \text{if } t' \in [0, \tau - 1], \\ o \oplus M^{\lfloor t/\Pi \rfloor}_{t'-\tau} & \text{if } t' \in [\tau, \tau + n - 2], \\ \overline{o} & \text{if } t' \in [\tau + n - 1, 2\tau + n - 2], \\ \overline{o} \oplus M^{\lfloor t/\Pi \rfloor}_{t'-(2\tau+n-1)} & \text{if } t' \in [2\tau + n - 1, \Pi - 1], \end{cases}$$

Given $x^*(t)$, we define $\mathrm{BSO}_{o,\tau}(x,t) := n - \mathrm{H}(x, x^*(t))$.

To define the transition from o to $\overline{o}$ we use transition masks $M^c_s \in \{0,1\}^n$ (for $s \in \{0,1,\ldots,n-2\}$) where $c \in \mathbb{N}_0$ denotes the number of the period (so that the random transition is potentially different in each period). Note that with $c = \lfloor t/\Pi \rfloor$ we have $t = c \cdot \Pi + t'$. When we define $M^c_s \in \mathcal{M}$ for some set $\mathcal{M}$ this means that M^c_s is selected uniformly at random from $\mathcal{M}$.

We define $M^c_0 \in \{0^i 10^{n-i-1} \mid i \in \{0,1,\ldots,n-1\}\}$. For $s \in \{1,2,\ldots,n-2\}$ and $c \in \mathbb{N}_0$ we define

$$M^c_s \in \begin{cases} \{0^{i-1}1^{s+1}0^{n-i-s}, 0^i 1^{s+1} 0^{n-i-s-1}\} & \text{if } M^c_{s-1} \in \{0^i 1^s 0^{n-i-s} \mid i \in [1, n-s-1]\}, \\ \{1^{i+1}0^{n-s-1}1^{s-i}, 1^i 0^{n-s-1} 1^{s-i+1}\} & \text{if } M^c_{s-1} \in \{1^i 0^{n-s} 1^{s-i} \mid i \in [0,s]\}. \end{cases}$$

The definition of the transition masks M^c_s ensures that the unique global optimum moves from o to $\overline{o}$ in a very specific way. The bits where o and $o \oplus M^c_s$ differ are always in one contiguous block if one allows for blocks that are 'wrapping around', i. e., not ending at the right end of the bit string but being considered as continuing at the beginning. This can help artificial immune systems which make use of contiguous hypermutations (Kelsey and Timmis, 2003) since those mutations always flip some contiguous blocks of bits and have a much better chance of performing such a mutation than standard bit mutations that are used in evolutionary algorithms.

To further clarify the definition of $\mathrm{BSO}_{o,\tau}$ we present a small example for $n = 4$, $o = 0110$ and $\tau = 6$. We consider the location of the unique global optimum for each step of the first period, i. e., for each $t \in \{0,1,\ldots,\Pi-1\}$ with $\Pi = 2(\tau + n - 1) = 18$ in our example. To do this we need to fix the random transition masks and decide $M^0_0 = 1000$, $M^0_1 = 1100$, $M^0_2 = 1101$, $M^0_3 = 1111$. Note that the first mask is selected

uniformly at random from the n masks with exactly one 1-bit and the other masks are selected uniformly at random from the two possible masks that extend the current block of 1-bits either to the left and right. The final mask, M_3^0, was not actually random since there was only one choice left. We depict the sequence of unique global optima in Figure 1.

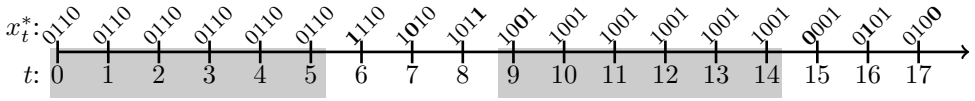


Figure 1: Example: Sequence of unique global optima for $\text{BSO}_{\mathbf{0110,6}}$ with 'random' transition masks $\boldsymbol{M_0^0 = 1000}$, $\boldsymbol{M_1^0 = 1100}$, $\boldsymbol{M_2^0 = 1101}$, $\boldsymbol{M_3^0 = 1111}$. The stable phases are marked in gray. Bits in the global optima differing from the predecessor are printed in bold.

Using the same visualisation as in Figure 1 we depict an abstract version of the way the unique global optimum moves in Figure 2, this time for two complete periods. It also shows the change from the transition masks from M_i^0 to M_i^1 from the first to the second period. The random choice of the transition masks indicates that it is much more useful to remember the unique global optima from the stable phases, o and its bit-wise complement $\overline{o}$, than it is to remember the intermediate points.

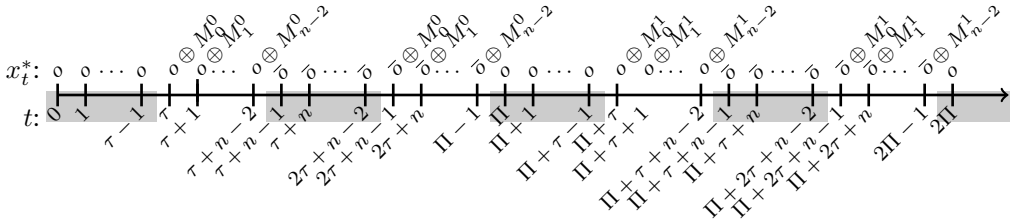


Figure 2: Sequence of unique global optima for $\text{BSO}_{o,\tau}$ for bit string length n, first cycle and first global optimum of second cycle. The stable phases are marked in gray.

4 Evolutionary Algorithms and Artificial Immune Systems

We consider one evolutionary algorithm and two variants of an artificial immune system. The evolutionary algorithm is known as (μ+λ) evolutionary algorithm (EA), uses a population of size μ, uniform selection for reproduction, generates λ independently identically distributed offspring by means of standard bit mutations (Algorithm 1) with mutation probability $1/n$ (and no crossover) and employs plus-selection to select the population for the next generation (see, e. g., Jansen (2013)). A formal description is given as Algorithm 2.

Algorithm 1: Standard Bit Mutation (SBM)

Input: $x \in \{0,1\}^n$
for $k \in \{1,\dots,n\}$
  With probability $1/n$, invert the bit $x[k]$;

Algorithm 2: (μ+λ) evolutionary algorithm (EA)

Input: μ, λ

1 Let $g := 0$ and choose $x_1^{(g)}, \ldots, x_\mu^{(g)} \in \{0,1\}^n$ independently uniformly at random (u. a. r.);
2 **repeat**
3 Set $g := g + 1$;
4 **for** $i \in \{1, 2, \ldots, \lambda\}$ `// Variation`
5 Select $y_i \in \{x_1^{(g-1)}, \ldots, x_\mu^{(g-1)}\}$ u. a. r.;
6 Perform SBM(y_i) `// see Algorithm 1;`
7 **Selection:** Sort $x_1^{(g-1)}, \ldots, x_\mu^{(g-1)}, y_1, \ldots, y_\lambda$ descending according to fitness, break ties by sorting $y_1, \ldots, y_\lambda$ in front of $x_1^{(g-1)}, \ldots, x_\mu^{(g-1)}$, break remaining ties u. a. r. Set $x_1^{(g)}, \ldots, x_\mu^{(g)}$ to the first μ elements from this sorted sequence.
8 **until** *some termination condition is met*;

The artificial immune system is the B-cell algorithm as introduced by Kelsey and Timmis (2003). It uses a population of size μ, generates λ clones for each member of the population, and applies somatic contiguous hypermutations (Algorithm 3) to all and additionally standard bit mutation (Algorithm 1) to one of them. It applies plus-selection between each member of the population and its clones. A formal description is given as Algorithm 4.

Algorithm 3: Somatic Contiguous Hypermutation (CHM)

Input: $x \in \{0,1\}^n$

1 Select $p \in \{0, 1, \ldots, n-1\}$ uniformly at random;
2 Select $l \in \{0, 1, \ldots, n\}$ uniformly at random;
3 **for** $k \in \{0, 1, \ldots, l-1\}$
4 Invert the bit $x[1 + ((p + k) \bmod n)]$;

We additionally consider a variant of the B-cell algorithm suggested by Jansen et al. (2011). The only difference is that somatic contiguous hypermutations are applied to the individual undergoing standard bit mutation with probability p_{CHM}. Thus, with probability $1 - p_{\text{CHM}}$ one of the offspring is subject to standard bit mutations, only. To obtain a formal description, we replace lines 6–11 in Algorithm 4 by Algorithm 5. Usually we want the probability p_{CHM} to be some positive constant strictly less than 1. For experiments we will use $p_{\text{CHM}} = 1/2$ and invite the reader to think of p_{CHM} as this value in all contexts. For theoretical results we will be more general and mention for what range of probabilities the statements hold.

We summarise the three algorithms considered in this work:

Definition 4. *In the following, we refer to Algorithm 2 simply as* EA. *We call Algorithm 4* BCA *and denote Algorithm 4 using Algorithm 5* BCA*.

For the algorithms we assume that whenever a fitness value $f(x)$ is computed this is automatically translated to the appropriated evaluation $\text{BSO}_{o,\tau}(x,t)$ with parameters o and τ and correct current time step t without the need for the algorithm to be aware of the values of o, τ and t. We keep track of 'time' inside the algorithms by means of a generation counter g.

Algorithm 4: The B-Cell Algorithm (BCA)

Input: μ, λ

Let $g := 0$ and choose $x_1^{(g)}, \ldots, x_\mu^{(g)} \in \{0,1\}^n$ independently uniformly at random (u. a. r.);
repeat
  Set $g := g + 1$;
  for $i \in \{1, \ldots, \mu\}$ `// Clonal expansion`
    for $j \in \{1, \ldots, \lambda\}$ Set $y_{i,j} := x_i^{(g-1)}$;
  for $i \in \{1, \ldots, \mu\}$ `// Standard bit mutation`
    Select $j \in \{1, \ldots, \lambda\}$ u. a. r.;
    Perform SBM($y_{i,j}$) `// see Algorithm 1 ;`
  for $i \in \{1, 2, \ldots, \mu\}$ `// Hypermutation`
    for $j \in \{1, 2, \ldots, \lambda\}$
      Perform CHM($y_{i,j}$) `// see Algorithm 3 ;`
  for $i \in \{1, 2, \ldots, \mu\}$ `// Selection`
    if $f(x_i^{(g-1)}) \leq \max\{f(y_{i,1}), \ldots, f(y_{i,\lambda})\}$ **then**
      Set $x_i^{(g)} := y_{i,j}$ where $f(y_{i,j}) = \max\{f(y_{i,1}), \ldots, f(y_{i,\lambda})\}$, break ties u. a. r.;
    else
      Set $x_i^{(g)} := x_i^{(g-1)}$
until *some termination criterion is met*;

As discussed in Section 2, we assume that the execution platform ensures that values of search points do not change during one round, i. e., one generation can be carried out within one time step of the dynamic optimisation problem. For the algorithms considered here this means that we need to have $\mu(\lambda + 1) \leq \sigma$ for the BCA and BCA* and $\mu + \lambda \leq \sigma$ for the EA as these are the numbers of function evaluations per generation for the two algorithms.

5 Analysis of Evolutionary Algorithms and Artificial Immune Systems for the Dynamic Bi-Stable Example Problem

5.1 Heuristic Dynamic Optimisation with a Slow Execution Platform

We start our analyses by considering slow execution platforms with $\sigma = \Theta(1)$ and compare the three algorithms introduced in Section 4 for three different lengths of the

Algorithm 5: Modified BCA Mutation

Input: $\mu, \lambda, p_{\text{CHM}}$

for $i \in \{1, \ldots, \mu\}$
  Select $g \in \{1, \ldots, \lambda\}$ u. a. r. `// Standard bit mutation;`
  Perform SBM($y_{i,g}$) `// see Algorithm 1 ;`
  With probability p_{CHM}, perform CHM($y_{i,g}$) `// Algorithm 3 ;`
  for $j \in \{1, 2, \ldots, \lambda\}$, $j \neq g$ `// Hypermutation`
    Perform CHM($y_{i,j}$) `// see Algorithm 3 ;`

stable interval in this setting: $\tau = n$, $\tau = n^{3/2}$, and $\tau = n^3$. Note that $\sigma = \Theta(1)$ implies $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ since we assume that one generation can be carried out within one time step of the dynamic optimisation problem.

5.1.1 Short Stable Intervals

We show that all three algorithms fail to catch up with the global optimum if the stable phase is short, i. e., $\tau = n$. We start with an analysis of the EA and afterwards transfer the theoretical results to the two variants of the BCA.

Theorem 5. *Let $\tau = n$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always bad in the first $n^{O(1)}$ steps with probability converging to 1.*

Proof. Since $\mu = \Theta(1)$, all initial search points have Hamming distance $\Theta(n)$ to the optimum with probability exponentially close to 1. We first consider the situation during a stable phase of length τ, i. e., while the optimum does not move. Let d denote the current Hamming distance to the optimum. The expected decrease in distance in one mutation is bounded above by $(d/n)(1-1/n)^{n-d}$. For $d < n$ we have that $(1-1/n)^{n-d} < c < 1$ for some appropriately chosen constant $c \in (0,1)$. We see that the expected decrease in distance in $n\lambda$ mutations, i. e., n generations, is bounded above by $(d/n)c \cdot n\lambda = cd\lambda < d\lambda$. Application of Chernoff bounds yields that the probability to decrease the distance by at least $c(1+\varepsilon)d\lambda$ is bounded above by $e^{-\Omega(d\lambda)}$ for all positive constant ε. We can choose $\varepsilon = (1-c)/(2c\lambda)$ which guarantees that $\varepsilon > 0$ and $c(1+\varepsilon)d < d$ both hold. For $d = \Theta(n)$ the probability is exponentially small and the claim follows.

Outside the stable phases the optimum is moving. If the optimum is moving away from the current search point it is even more unlikely to decrease the Hamming distance to $o(n)$. For the case where it moves towards the current search point assume that the optimum moves from o to $\overline{o}$ (the other case is symmetric). We consider all points with equal distance from the search point o. Due to symmetry reasons all these points have equal probability to become the current search point. The moving global optimum will hit exactly one of these points. Since the current search point has linear Hamming distance to o there are exponentially many such search points and it is exponentially unlikely that the global optimum decreases the Hamming distance to the current search point below $c'n$ for some sufficiently small constant $c' > 0$. □

Theorem 6. *Let $\tau = n$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) is always bad in the first $n^{O(1)}$ steps with probability converging to 1.*

Proof. Again, after initialisation all search points have Hamming distance $\Theta(n)$ to the optimum with probability exponentially close to 1. For the BCA we observe that the probability for any specific mutation is $\Theta\big(1/n^2\big)$. This implies that the probability to decrease the Hamming distance to the global optimum by any constant is significantly smaller than for the EA. Since the initial search point is selected uniformly at random the probability that the Hamming distance can be decreased by $\omega(\log n)$ is superpolynomially small. When optimising ONEMAX this does not change significantly (Jansen and Zarges, 2011). This implies that the BCA performs at most as well as the EA. □

Theorem 7. *Let $\tau = n$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$ and $p_{CHM} \in [0,1]$. The BCA* (Definition 4) is always bad in the first $n^{O(1)}$ steps with probability converging to 1.*

Proof. In the proof of Theorem 6 we have seen that the probability to decrease the Hamming distance by $\omega(\log n)$ by means of contiguous hypermutations is superpolynomially small. Additionally applying one standard bit mutation to one of the offspring does not change this. It is not important if we apply only standard bit mutation or standard bit mutation together with contiguous hypermutations. Therefore, the value of p_{CHM} is not important and the results follows in the same way. □

The above theorems prove that all three algorithms perform badly in the considered setting and fail to catch up with the global optimum. However, the theoretical results obtained are rather abstract and only provide a coarse picture of the situation at hand. We therefore consider the results of experiments to provide a more concrete and clear picture. All experiments comprise of 100 independent runs of the three considered algorithms for $n = 100$, $\sigma = 2$, and $\lambda = \mu = 1$. For the BCA* we set $p_{\text{CHM}} = 1/2$. We display average fitness function values over these runs in Figure 3a and indicate start and end of stable intervals by vertical lines. Note that the x-axis displays time steps as defined by the dynamic problem, not generations or function evaluations. In one time step up to σ function evaluations are made (potentially 'wasting' function evaluations if there are not enough function evaluations left in the current time step to accommodate a complete generation). The algorithms we consider perform either $\lambda + \mu$ function evaluations (the EA) or $\mu(\lambda + 1)$ function evaluations (the BCAs) per generation. Since we have $\sigma = 2$ and $\lambda + \mu = \mu(\lambda + 1) = 2$ here we do have that time steps and generations coincide in this case.

We see that all three algorithms are bad as predicted by Theorems 5, 6, and 7 and are not able to reduce the distance to 20 or below (corresponding to fitness 80 or above). We see that the EA gets clearly closer to the global optimum in the stable phases than the BCA and has worse fitness when the optimum moves rapidly. Note that our theoretical results are too coarse to reveal these differences. The BCA* seems to combine the advantages of the other two algorithms and thus, performs best with respect to the observed average function values.

In order to investigate the significance of these experimental results we have performed Wilcoxon signed rank tests (Lehmann, 2006) for each pair of algorithms and each iteration. Due to the large number of tests for each pair, we perform Holm-Bonferroni correction (Holm, 1979) and depict the resulting p-values in Figure 4a along with the standard confidence level of 0.05.

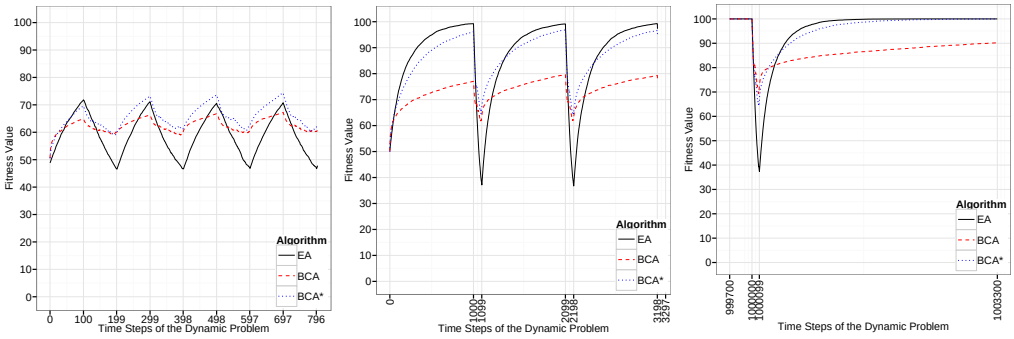
Here and in the following the diagrams showing the results of the Wilcoxon signed rank tests confirm that, roughly speaking, differences in functions values which are clearly visible tend to be statistically significant. In time steps where the function values of the potential solutions are very similar or even intersect there are no statistically significant differences, of course. The interested reader can see the details in those plots.

5.1.2 Long Stable Intervals

We now consider a longer stable period, i. e., $\tau = n^{3/2}$. We see that this length is still not sufficiently long for the BCA, however, both the EA and the BCA* are now able to catch up with the global optimum.

Theorem 8. *Let $\tau = n^{3/2}$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The BCA (Definition 4) is always bad in the first $n^{O(1)}$ steps with probability converging to 1.*

Proof. The statement follows from the proof of Theorem 6. There we have argued that the expected decrease in distance for the BCA is $O\big(\log(n)/n^2\big)$ per generation. Thus,



(a) $\tau = 100$ (Th. 5, 6, 7) (b) $\tau = 1000$ (Th. 8, 9, 10) (c) $\tau = 10^6$ (Cor. 11, 13; Th. 12)

Figure 3: Visualisation of experiments showing average fitness values over time over 100 independent runs for all three algorithms on $\text{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 2$, $\mu = \lambda = 1$, $p_{\text{CHM}} = 0.5$, and different values of τ. Vertical lines indicate start and end of stable intervals.

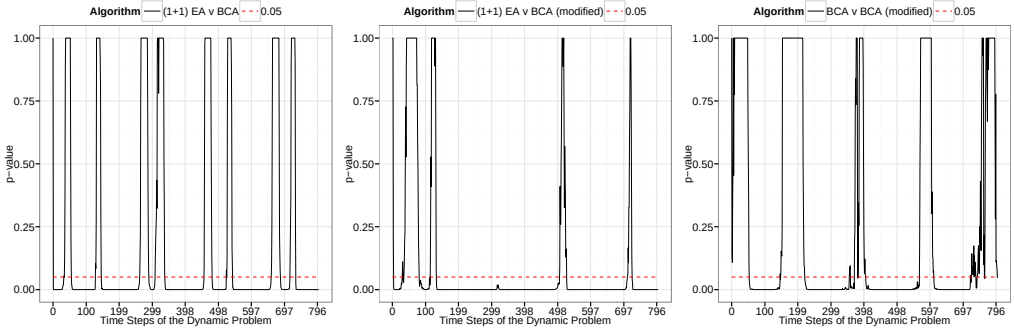
the expected decrease in $O(n^{3/2})$ generations is bounded above by $O(\log(n)/n^{1/2})$ and the result follows since in the phases where the optimum moves it is increased with probability converging to 1 as shown in the proof of Theorem 6. □

Theorem 9. *Let $\tau = n^{3/2}$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) becomes perfect after $O(n \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes bad again in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

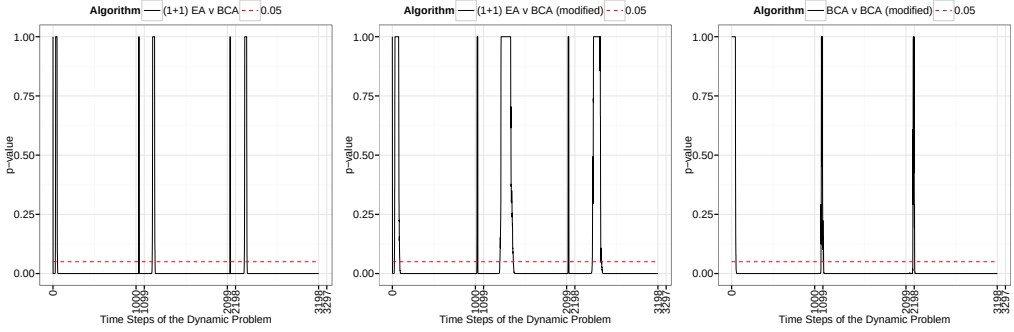
Proof. The statement about the repeating behaviour is a direct consequence of the first statement when applying the union bound. We see that the EA becomes perfect after $O(n \log n)$ steps and remains perfect as long as the optimum does not change as we did in the proof of Theorem 5. Now consider the subsequent n steps where the optimum changes by 1 bit in each of the steps. If the Hamming distance to the global optimum and the current search point is d the probability that the EA is able to decrease the Hamming distance is bounded above by d/n. We see that in n steps a Hamming distance $d = \Theta(n)$ is reached with probability exponentially close to 1. After the global optimum has reached either o or $\overline{o}$ we are in a simulation very similar to the one after initialisation and we can repeat the argument. □

Theorem 10. *Let $\tau = n^{3/2}$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda + 1) \leq \sigma$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) becomes perfect after $O(n \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes bad again in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

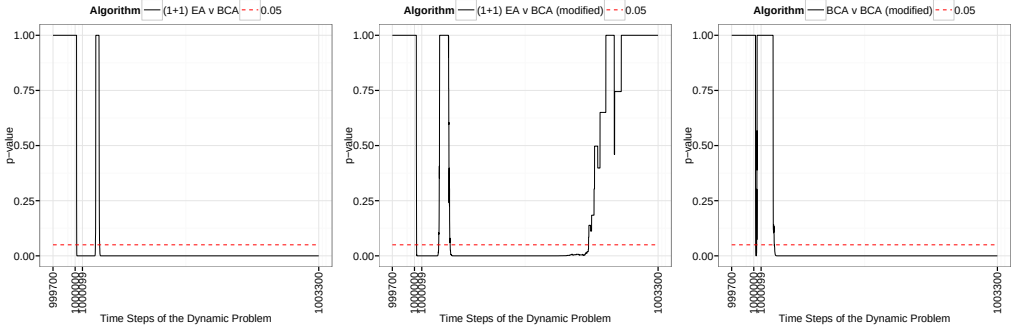
Proof. We consider only the offspring which are created with application of standard bit mutation. With probability $1 - p_{\text{CHM}} = \Omega(1)$ such an offspring is not also subject to a contiguous hypermutations. For those offspring the probability distribution is identical to the offspring in the EA. Since all values involved (the speed of the execution platform σ, the number of individuals μ and offspring λ) are constants the introduction of the factor $1 - p_{\text{CHM}} = \Theta(1)$ does not change anything significantly. Therefore, the statements about becoming and remaining perfect follow from the proof of Theorem 9.



(a) $\tau = 100$ (Theorems 5, 6, 7)



(b) $\tau = 1000$ (Theorems 8, 9, 10)



(c) $\tau = 10^6$ (Corollaries 11, 13; Theorem 12)

Figure 4: Visualisation of experiments showing p-values of the Wilcoxon tests after Holm-Bonferroni correction over 100 independent runs for all three algorithms on $\mathrm{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 2$, $\mu = \lambda = 1$, $p_{\mathrm{CHM}} = 0.5$ and different values of τ (cf Theorems 5, 6, and 7). Vertical lines indicate start and end of stable intervals.

We have already seen in the proof of Theorem 8 that contiguous hypermutations do not help in avoiding becoming bad. Therefore, the complete statement follows. □

We again perform experiments to get a more complete picture and depict the results in Figure 3b and 4b in the same way as in the previous subsection. We see that the BCA is still not able to keep up with the global optimum. While the EA reaches the global optimum just before the end of the stable phase, the BCA* is not quite as successful. On first sight this seems to be a contradiction to Theorem 10. However, we stress that all our results are asymptotic and thus, $O(n \log n)$ can still be smaller than $\tau = n^{3/2}$ for $n = 100$. We remark that for $n = 1000$ we obtain the results predicted by Theorem 10. We omit a visualisation of these results due to space restrictions.

5.1.3 Very Long Stable Intervals

Finally, we consider very long stable intervals, i. e., $\tau = n^3$. It does not come as a surprise that in settings where an algorithm was already good for shorter stable intervals this continues to be the case here. Moreover, the longer stable phases allows the BCA to catch up with the global optimum, too, whereas the stable phase of length $n^{3/2}$ was too short for this. All results here are quite direct consequence from results in the earlier subsections.

Corollary 11. *Let $\tau = n^3$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) becomes perfect after $O(n \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes bad again in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. Follows directly from Theorem 9 since a longer stable interval cannot decrease the performance of the algorithm. □

Theorem 12. *Let $\tau = n^3$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda + 1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O\big(n^2 \log n\big)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. We call this a phase. In the next $p(n) = n^{O(1)}$ phases this behaviour is repeated in $p(n) - o(p(n))$ phases with probability converging to 1.*

Proof. Theorem 8 in (Jansen and Zarges, 2011) proves an upper bound of $O\big(n^2 \log n\big)$ for the BCA on ONEMAX. This implies that the BCA becomes perfect after $O\big(n^2 \log n\big)$ steps and remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase. That it becomes mediocre or bad in the next n steps, when the optimum changes rapidly, follows from the proof of Theorem 6. Since all this happens with probability very close to 1 the statement about the repetition in a polynomial number of phases follows. □

Corollary 13. *Let $\tau = n^3$, $\sigma = \Theta(1)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda + 1) \leq \sigma$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) becomes perfect after $O(n \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes bad again in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. Follows directly from Theorem 10 since a longer stable interval cannot decrease the performance of the algorithm. □

Again we perform experiments to get a more complete picture and depict the results for the most interesting time steps (around the unstable interval) in Figure 3c and 4c. The experiments match the predictions in our theorems.

5.2 Heuristic Dynamic Optimisation with a Fast Execution Platform

In this subsection we consider the situation when we have a faster execution platform. We consider the situation when the execution platform is able to make $\sigma = \Theta(n^{3/2})$ function evaluations in one time step of the dynamic optimisation problem. This allows us to consider the algorithms with larger number of search points and study the effects of this. The comparison is fair and meaningful since all algorithms have the same computational budget each time the dynamic problem has the chance to change. In this paper we restrict our attention to larger offspring population size λ and leave the number of search points μ an algorithm uses as bases for its search restricted to $\mu = \Theta(1)$. Studying the effects of larger populations is beyond the scope of this article. We conjecture that having larger values for μ makes only sense with either more complex dynamic optimisation problems or when one additionally employs mechanisms to make the population maintain some level of diversity (or both). See work by Oliveto and Zarges (2014) for an example of such a study.

5.2.1 Short Stable Intervals

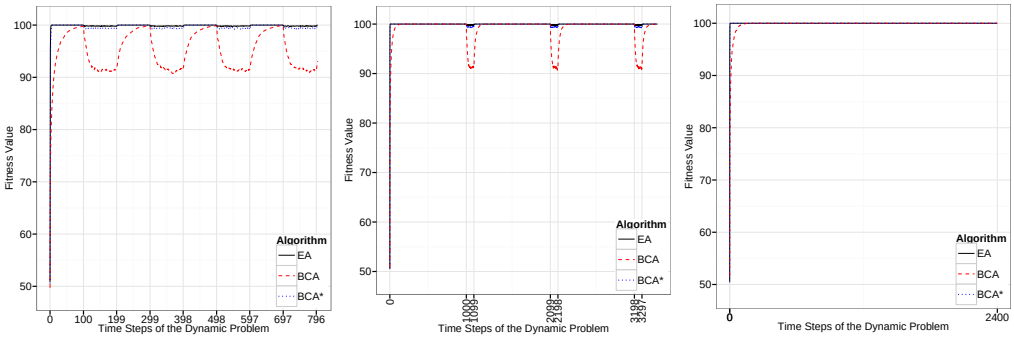
We begin with the consideration of very small offspring populations, i. e., $\lambda = \Theta(1)$. This means we invest the increased speed of the execution platform into having more generations per time step.

Theorem 14. *Let $\tau = n$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. It is well known that the expected optimisation time of the EA with $\mu = \lambda = 1$ on ONEMAX is $\Theta(n \log n)$ and that the probability not to be finished in $O(n^{3/2})$ generations is exponentially small (Jansen, 2013). This implies that the EA performs perfectly after one step with overwhelming probability. After this it will remain perfect while the optimum does not move. When the optimum moves it changes by 1 bit in each step. In a single step the EA generates $\lfloor \sigma/2 \rfloor$ offspring by means of standard bit mutation. The probability that none of these equals the new optimum is bounded above by $\left(1 - (1/n)(1-1/n)^{n-1}\right)^{\Theta(n^{3/2})} = e^{-\Omega(n^{1/2})}$. Thus, the probability to be always perfect for the next $n^{O(1)}$ steps is $1 - n^{O(1)} \cdot e^{-\Omega(n^{1/2})} = 1 - e^{-\Omega(n^{1/2})}$. □

Theorem 15. *Let $\tau = n$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O(n^{1/2} \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes bad again in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. In each step the BCA performs $\Theta(\sigma) = \Theta(n^{3/2})$ generations. In the beginning we have a stable phase and $\text{BSO}_{o,\tau}$ is ONEMAX-like for τ steps. It is known that the BCA optimises this function on average and with probability very close to 1 in $O(n^2 \log n)$ generations (Jansen and Zarges, 2011). Thus, it reaches the optimum in a stable phase within the first $O(n^{1/2} \log n)$ steps with probability converging to 1. The second part of the statement follows from the proof of Theorem 6. When the optimum starts moving the probability to decrease the Hamming distance by 1 in 1 generation is $O(1/n^2)$. Thus, the BCA remains perfect only with probability $O(1/n^{1/2})$. Remember that also for larger Hamming distances the expected decrease in distance is bounded by $O(\log(n)/n^2)$. Thus, with probability converging to 1, the Hamming distance becomes linear in n steps. □



(a) $\tau = 100$ (Th. 14, 15, and 16) (b) $\tau = 1000$ (Th. 21, 22, 23) (c) $\tau = 10^6$ (Th. 27, 28, 29)

Figure 5: Visualisation of experiments showing average fitness values over time over 100 independent runs for all three algorithms on $\text{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 1001$, $\mu = \lambda = 1$, $p_{\text{CHM}} = 0.5$, and different values of τ. Vertical lines indicate start and end of stable intervals.

Theorem 16. *Let $\tau = n$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$, $\lambda = \Theta(1)$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA* (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. The proof makes use of the same idea as the proof of Theorem 10. In each generation we consider only the offspring that is created by means of standard bit mutation and that with probability $1 - p_{\text{CHM}} = \Theta(1)$ is not also subject to a contiguous hypermutation. Again, the introduction of this constant factor does not change anything significantly and the result is a direct consequence of Theorem 14. □

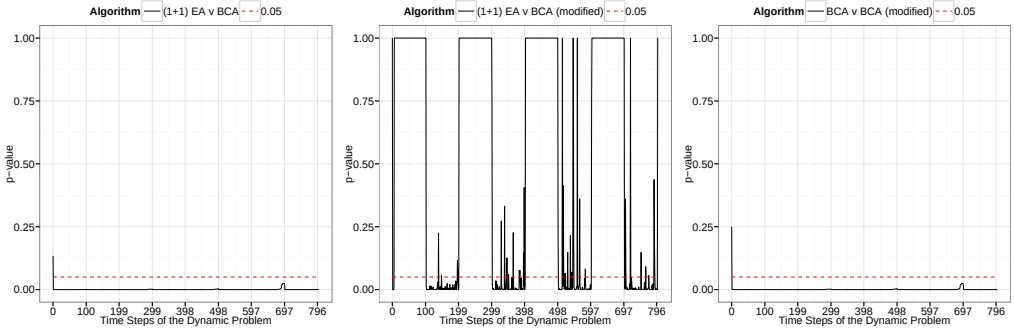
We again perform experiments and present their results in Figure 5a and 6a.

The alternative to sticking with small offspring population sizes and having as many generations per time step as possible is to increase the offspring population size. Of course, the execution platform needs to be fast enough to execute at least one complete iteration of the three considered algorithms. The number of function evaluations for the EA equals $\mu + \lambda = \lambda + 1$ (since we have $\mu = 1$) and for the two BCA variants it equals $\mu \cdot (\lambda + 1) = \lambda + 1$ (since we have $\mu = 1$). This implies that $\lambda \leq \sigma - 1$ needs to hold. We restrict our attention to the extreme case where $\lambda = \Theta(\sigma)$, i. e., to the case where the offspring population size of so large that the number of generations per time step is bounded above by a constant.

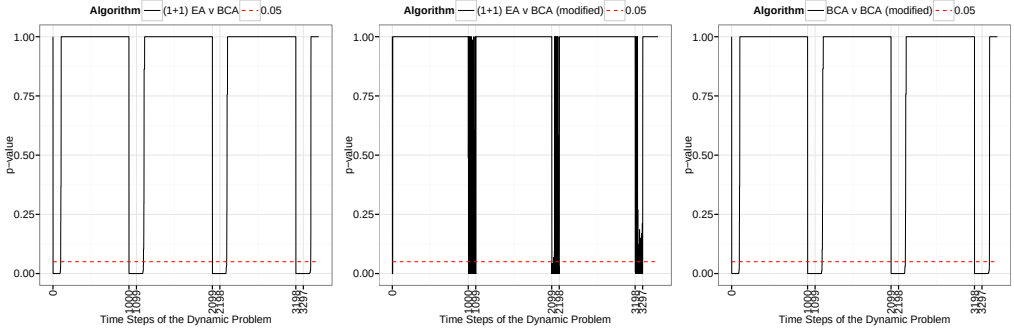
Theorem 17. *Let $\tau = n$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. We consider the situation directly after initialisation and denote with d the current Hamming distance to the optimum. Clearly, $d \leq n$ holds. The probability to decrease the Hamming distance by at least 1 in a single generation of the EA with $\lambda = \Theta(n^{3/2})$ is at least

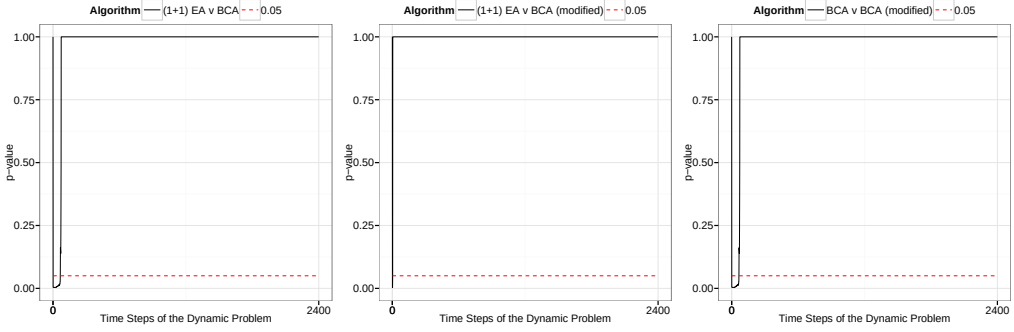
$$1 - \left(1 - \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1}\right)^{\Theta(n^{3/2})} = 1 - e^{-\Omega(\sqrt{n})}$$



(a) $\tau = 100$ (Theorems 14, 15, 16)



(b) $\tau = 1000$ (Theorems 21, 22, 23)



(c) $\tau = 10^6$ (Theorems 27, 28, 29)

Figure 6: Visualisation of experiments showing p-values of the Wilcoxon tests after Holm-Bonferroni correction over 100 independent runs for all three algorithms on $\text{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 2$, $\mu = \lambda = 1$, $p_{\text{CHM}} = 0.5$ and different values of τ. Vertical lines indicate start and end of stable intervals.

and the probability to see up to $d \leq n$ such decreases in d subsequent generations is at least $1 - d \cdot e^{-\Omega(\sqrt{n})} = 1 - e^{-\Omega(\sqrt{n})}$. Thus, after at most n generations, the EA is perfect. Since the EA executes n generations within the first $\tau = n$ steps, i. e., before the optimum starts moving, the first claim follows.

Outside the stable phases the optimum is moving, however, in each step only a single bit of the current optimum changes. Thus, again with probability $1 - e^{-\Omega(\sqrt{n})}$, the EA is able to catch up with this change and the probability to be always perfect for the next $n^{O(1)}$ steps is $1 - n^{O(1)} \cdot e^{-\Omega(\sqrt{n})} = 1 - e^{-\Omega(\sqrt{n})}$. □

For the BCA things are rather tight for this kind of setting. We start with proving a statement about the expected number of iterations the BCA with $\lambda = \Theta(n^{3/2})$ needs to optimise ONEMAX. We use this result to establish a result for the BCA in a setting that is a bit weaker than what we normally consider.

Lemma 18. *The BCA with $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ (Definition 4) finds the optimum of* ONEMAX *in expected $\mathrm{E}(T) = O(n)$ iterations.*

Proof. Let d denote the Hamming distance of the current search point to the global optimum. Then, a single contiguous hypermutation decreases the Hamming distance with probability $d/(n \cdot (n+1))$. The probability that at least one of the $\lambda = \Theta(n^{3/2})$ offspring decreases the Hamming distance is $1 - (1 - d/(n(n+1)))^{\Theta(n^{3/2})} \geq 1 - e^{-\Omega(d/\sqrt{n})}$. Using standard fitness-level argument and $\exp(x)/(\exp(x) - 1) \leq 1 + 1/x$ we derive an upper bound on $\mathrm{E}(T)$:

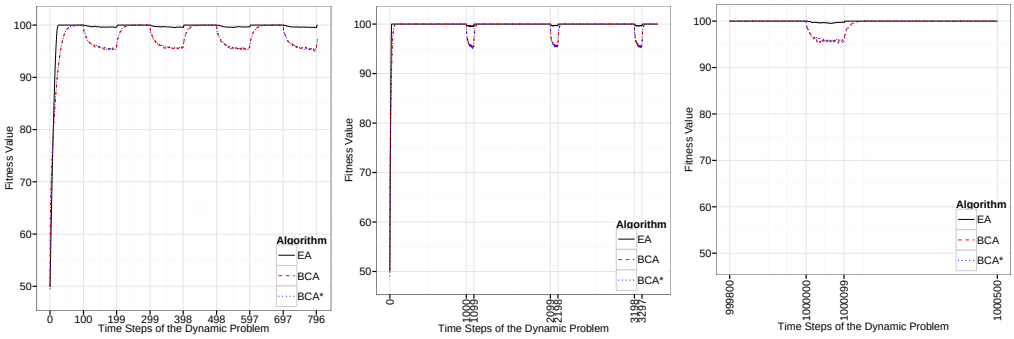
$$\begin{aligned}\mathrm{E}(T) \leq \sum_{d=1}^{n} \frac{1}{1 - \exp(-d/\sqrt{n})} &= \sum_{d=1}^{n} \frac{\exp(d/\sqrt{n})}{\exp(d/\sqrt{n}) - 1} \\ &\leq \sum_{d=1}^{n} \left(1 + \frac{1}{d/\sqrt{n}}\right) = n + \sqrt{n} \cdot \sum_{d=1}^{n} \frac{1}{d} = n + 2\sqrt{n} H_n\end{aligned}$$

where H_n is the n-th Harmonic number. With $H_n = O(\ln n)$ we get $\mathrm{E}(T) = O(n + \sqrt{n}\ln(n) + \sqrt{n}) = O(n)$. □

Theorem 19. *Let $\tau = c \cdot n$ ($c > 0$ constant), $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu(\lambda + 1) \leq \sigma$. If $c = O(1)$ is sufficiently large, the BCA (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. In the next $n^{O(1)}$ steps this behaviour is repeated with probability converging to 1.*

Proof. After initialisation the Hamming distance to the global optimum is $\Theta(n)$ with probability exponentially close to 1. We know that BCA becomes perfect in $O(n)$ iterations if the optimum does not move. If the stable phase has length $\tau = cn$ big enough this will happen and the BCA remains perfect for the remaining steps of the stable phase.

At the end of the stable phase the optimum starts to move by 1 bit per step. As long as the BCA is good the Hamming distance between the optimum and the current search point of the BCA is $O(1)$ and for a single offspring the probability to exactly hit the optimum is $O(1/n^2)$. Thus, for $\lambda = \Theta(n^{3/2})$ the probability to find the optimum is bounded by $O(1/n^{1/2})$. We conclude that the BCA becomes mediocre or worse. □



(a) $\tau = 100$ (Th. 17, 19, and 20) (b) $\tau = 1000$ (Th. 24, 25, 26) (c) $\tau = 10^6$ (Th. 30, 31, 32)

Figure 7: Visualisation of experiments showing average fitness values over time over 100 independent runs for all three algorithms on $\text{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 1001$, $\mu = 1$, $\lambda = 1000$, $p_{\text{CHM}} = 0.5$, and different values of τ. Vertical lines indicate start and end of stable intervals.

When using small offspring population size, $\lambda = \Theta(1)$, we have seen that the BCA* had a performance that is comparable to that of the EA and much better than that of the BCA. The reason is that on a ONEMAX-like function standard bit mutations are much more efficient in reducing an already small Hamming distance to the global optimum further than contiguous hypermutations. Since the BCA* uses only standard bit mutation for one of the λ offspring with probability $1 - p_{\text{CHM}}$ the expected fraction of offspring that have equal probability distribution in the EA and the BCA* equals $(1 - p_{\text{CHM}})/\lambda$. With $\lambda = \Theta(1)$ and $1 - p_{\text{CHM}} = \Theta(1)$ this is a constant fraction of the offspring population. Since we only perform an asymptotic analysis it is not surprising that the performance of the EA and the BCA* are comparable. With $\lambda = \Theta\big(n^{3/2}\big)$ the fraction shrinks to $\Theta\big(1/n^{3/2}\big)$ and we can expect to see significant differences.

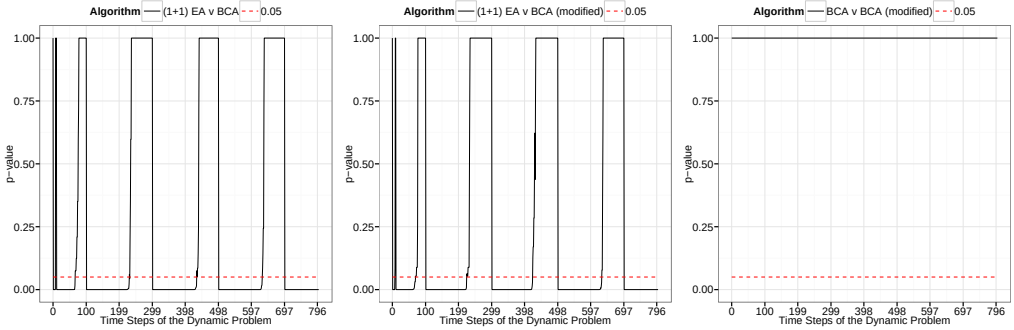
Theorem 20. *Let $\tau = c \cdot n$ ($c > 0$ constant), $\sigma = \Theta\big(n^{3/2}\big)$, $\mu = \Theta(1)$ and $\lambda = \Theta\big(n^{3/2}\big)$ with $\mu(\lambda + 1) \leq \sigma$. If $c = O(1)$ is sufficiently large, the BCA* (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. In the next $n^{O(1)}$ steps this behaviour is repeated with probability converging to 1.*

Proof. If we ignore the one offspring in each generation that is created by standard bit mutation the result follows from Theorem 19. Now we consider this offspring. We see that the probability that this offspring is able to follow the moving optimum in a single step is $O(1/n)$. However, there is only one such offspring and therefore the probability that any of the λ offspring (including the one that is created by standard bit mutation) locates the optimum is bounded by $O\big(1/n^{1/2}\big) + O(1/n) = O\big(1/n^{1/2}\big)$. Therefore the result follows from Theorem 19. □

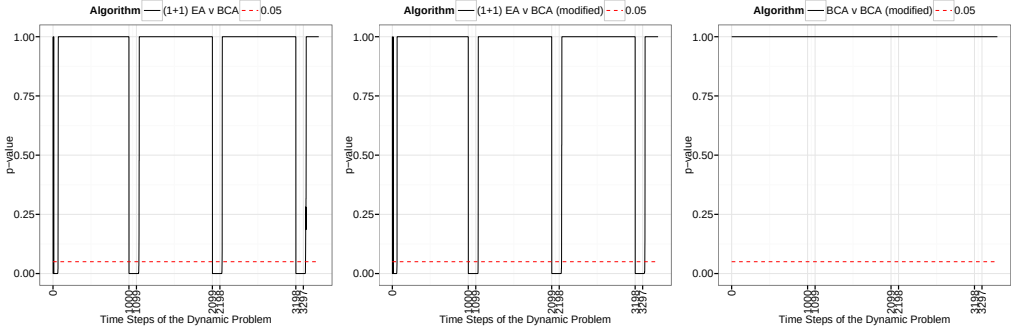
We again perform experiments and present their results in Figure 7a and 8a. We see that already $\tau = n$ (i. e., $c = 1$) is sufficiently large for the BCA and the BCA* to locate the optimum during the stable phases.

5.2.2 Long Stable Intervals

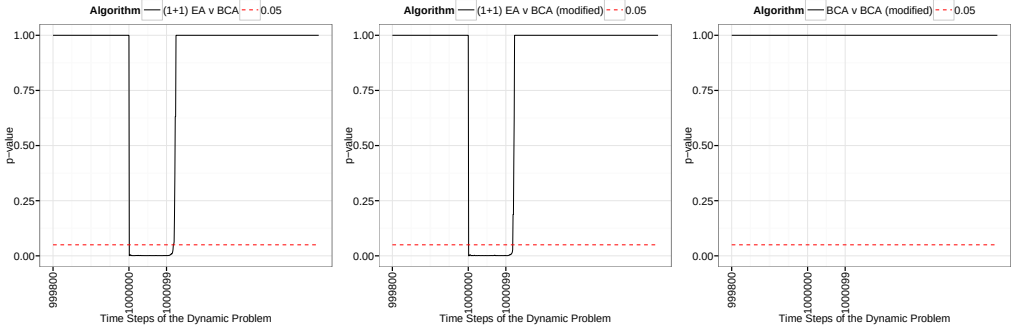
We now consider the situation when the length of the stable interval is considerably longer, $\tau = n^{3/2}$, giving the algorithms a much better chance to catch up with the



(a) $\tau = 100$ (Theorems 17, 19, 20)



(b) $\tau = 1000$ (Theorems 24, 25, 26)



(c) $\tau = 10^6$ (Theorems 30, 31, 32)

Figure 8: Visualisation of experiments showing p-values of the Wilcoxon tests after Holm-Bonferroni correction over 100 independent runs for all three algorithms on $\text{BSO}_{o,\tau}$ for $n = 100$, $\sigma = 2$, $\mu = 1$, $\lambda = 1000$, $p_{\text{CHM}} = 0.5$ and different values of τ. Vertical lines indicate start and end of stable intervals.

global optimum. Clearly, algorithm performance can only improve in comparison to shorter stable intervals. We start our investigation with small offspring population size where the performance was already quite good with much shorter stable phases.

Corollary 21. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first step for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. The statement is a direct consequence from the proof of Theorem 14 since making the stable phase longer by a polynomial number of steps cannot adversely affect the performance of the algorithm. □

Theorem 22. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O(n^{1/2} \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. We call this a phase. In the next $p(n) = n^{O(1)}$ phases this behaviour is repeated in $p(n) - o(p(n))$ phases with probability converging to 1.*

Proof. The proof is similar to the proof of Theorem 9. The bound of $O(n^{1/2} \log n)$ for the time needed to become perfect follows from the expected optimisation time $O(n^2 \log n)$ of the BCA on ONEMAX (Jansen and Zarges, 2011) and the fact that the BCA performs $\Theta(n^{3/2})$ generations per time step. The probability to reduce the Hamming distance by d is bounded by $O(d/n^2)$. The probability to see such an event in n steps before the Hamming distance is $\Omega(n)$ is bounded above by $n \cdot o(n/n^2) = o(1)$. The expected number of phases where we see this behaviour when considering $p(n)$ phases is bounded below by $(1 - o(n))\, p(n)$. Application of Chernoff bounds yields the result. □

Corollary 23. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$, $\lambda = \Theta(1)$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA* (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. The statement is a direct consequence of Theorem 16 since making the stable phase longer by a polynomial number of steps cannot adversely affect the performance of the algorithm. □

We again perform experiments and present their results in Figure 5b and 6b.

We now consider the case of using larger offspring populations. Like above we concentrate only on the extreme case, $\lambda = \Theta(n^{3/2})$.

Corollary 24. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 17 since a longer stable interval cannot decrease the performance of the algorithm. □

For the BCA with $\lambda = \Theta(n^{3/2})$ the behaviour changes considerably.

Theorem 25. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. The first part of the theorem follows direct from Lemma 18 since the first $\tau = n^{3/2}$ steps after initialisation are stable and the underlying problem corresponds to ONEMAX. Once the BCA is perfect in a stable phase, it remains stable until the end of the stable phase.

After the stable phase the optimum is moving and in each step the Hamming distance of the new global optimum to the global optimum in the stable phase is increased by 1. Let d denote the Hamming distance after d such steps. As in the proof of Lemma 18 the probability to decrease the Hamming distance by at least 1 is $1 - \exp(-\Omega(d/\sqrt{n}))$. For $d = \omega(\sqrt{n})$ this is $1 - o(1)$ and thus, with probability converging to 1, d will not increase beyond $o(n)$ between two stable phases.

We repeat the above argumentation to conclude the theorem. □

Theorem 26. *Let $\tau = n^{3/2}$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$, $\lambda = \Theta(n^{3/2})$ (with $\mu(\lambda+1) \leq \sigma$) and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. If we ignore the offspring that is created using standard bit mutation the result is a direct consequence of Theorem 25. Taking this one offspring per generation into account can only make things better. First we observe that it cannot speed up the expected number of generations needed to become perfect because this number is already $O(n)$ and standard bit mutations need $\Theta(n \log n)$ steps to achieve this. Since in the phase of change the optimum moves in each step by 1 bit and we only have $\Theta(\sigma/\lambda) = \Theta(1)$ generations per step this single standard bit mutation is insufficient to keep up with it. □

We again perform experiments and present their results in Figure 7b and 8b.

5.2.3 Very Long Stable Intervals

Here, we consider the same settings as before but now with very long stable intervals, concrete with $\tau = n^3$. We have already seen that longer stable phases imply better performance. As above we begin with very small offspring population sizes.

Corollary 27. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first step for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Corollary 21 since a longer stable interval cannot decrease the performance of the algorithm. □

Corollary 28. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O(n^{1/2} \log n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre or bad in the next n steps with probability converging to 1. We call this a phase. In the next $p(n) = n^{O(1)}$ phases this behaviour is repeated in $p(n) - o(p(n))$ phases with probability converging to 1.*

Proof. This is a direct consequence of Theorem 22 (longer stable interval). □

Corollary 29. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$, $\lambda = \Theta(1)$ (with $\mu(\lambda+1) \leq \sigma$) and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Corollary 23 (longer stable interval). □

Here, we also consider what happens if we invest into a larger offspring population size at the expense of the number of generations. As before we restrict our analysis to the extreme case $\lambda = \Theta(n^{3/2})$.

Corollary 30. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Corollary 24 (longer stable interval). □

Corollary 31. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 25 (longer stable interval). □

Corollary 32. *Let $\tau = n^3$, $\sigma = \Theta(n^{3/2})$, $\mu = \Theta(1)$, $\lambda = \Theta(n^{3/2})$ (with $\mu(\lambda + 1) \leq \sigma$) and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) becomes perfect after $O(n)$ steps, remains perfect for the remaining $\Theta(\tau)$ steps of the stable phase and becomes mediocre in the next n steps with probability converging to 1. This behaviour is repeated in the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 26 (longer stable interval). □

5.3 Heuristic Dynamic Optimisation with a Very Fast Execution Platform

Now we consider an even faster execution platform by considering $\sigma = \Theta(n^3)$ function evaluations in one time step of the dynamic optimisation problem. We consider the same lengths of the stable interval and the same offspring population sizes as in the previous section. We additionally analyse the extreme case $\lambda = \Theta(n^3)$. The case $\lambda = \Theta(n^{3/2})$ is now an intermediate case.

5.3.1 Short Stable Intervals

We start with the case $\tau = n$ and gradually increase the offspring population size. Most of the theoretical results follow directly from previous theorems since increasing the speed of the execution platform does not decrease the performance of the considered algorithms.

Corollary 33. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 14 since increasing the speed σ cannot decrease the performance of the algorithm. □

Theorem 34. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(1)$ with $\mu(\lambda + 1) \leq \sigma$. The BCA (Definition 4) is always perfect after the first step for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. For the BCA's performance after the first step and during the stable phase it suffices to remember that the expected optimisation time of the BCA with $\mu = \lambda = 1$ on ONEMAX is $O(n^2 \log n)$ and that the probability not to be finished in $O(n^3)$ generations is exponentially small (Jansen and Zarges, 2011). Analogously to the proof of Theorem 14 the probability that no offspring equals the new optimum is bounded above by $\left(1 - (1/(n^2+n))(1-1/n)^n\right)^{\Theta(n^3)} = e^{-\Omega(n)}$. Thus, the probability to be always perfect for the next $n^{O(1)}$ steps is $1 - n^{O(1)} \cdot e^{-\Omega(n)} = 1 - e^{-\Omega(n)}$. □

Corollary 35. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$, $\lambda = \Theta(1)$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$ with $\mu(\lambda+1) \leq \sigma$. The BCA* (Definition 4) is always perfect after the first step for $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 16 since increasing the speed σ cannot decrease the performance of the algorithm. □

Next, we consider $\lambda = \Theta(n^{3/2})$.

Corollary 36. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This is a direct consequence of Theorem 17 since increasing the speed σ cannot decrease the performance of the algorithm. □

Theorem 37. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^{3/2})$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) is always perfect after the first $O(1)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. The probability to decrease the Hamming distance by at least 1 in 1 generation is bounded below by $1-(1-1/(n(n+1)))^{\Theta(n^{3/2})} = 1 - e^{-\Theta(1/\sqrt{n})} = \Theta(1/\sqrt{n})$ (compare Lemma 18). In one step the BCA performs $\Theta(\sigma/\lambda) = \Theta(n^{3/2})$ generations. Therefore, the expected progress in 1 step is $\Theta(n)$ and this also holds with probability close to 1.

Once the optimum starts moving the probability not to catch up in one step is $\Theta\left((1-1/\sqrt{n})^{n^{3/2}}\right) = e^{-\Theta(n)}$. □

Corollary 38. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$, $\lambda = \Theta(n^{3/2})$ with $\mu(\lambda+1) \leq \sigma$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) is always perfect after the first $O(1)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This follows from Theorem 37 since only one of the $\lambda = \Theta(n^{3/2})$ offspring is subject to standard bit mutation. This is not a significant change. □

Finally, we examine the extreme case $\lambda = \Theta(n^3)$.

Corollary 39. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^3)$ with $\mu + \lambda \leq \sigma$. The EA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. The result is a direct consequence of Theorem 17 since increasing the speed σ (and the offspring population size λ accordingly) cannot decrease the performance of the algorithm. □

Theorem 40. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^3)$ with $\mu(\lambda+1) \leq \sigma$. The BCA (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. For the BCA we follow the argumentation of Theorem 17 for the EA and observe that the probability to decrease the Hamming distance by at least 1 in a single generation of the BCA is at least

$$1 - \left(1 - \frac{1}{n \cdot (n+1)}\right)^{\Theta(n^3)} \geq 1 - e^{-\Omega(n)}.$$

Consequently, the BCA is perfect after $O(n)$ steps and stays perfect for the next $n^{O(1)}$ steps with probability $1 - n^{O(1)} \cdot e^{-\Omega(n)} = 1 - e^{-\Omega(n)}$. □

Corollary 41. *Let $\tau = n$, $\sigma = \Theta(n^3)$, $\mu = \Theta(1)$ and $\lambda = \Theta(n^3)$ with $\mu(\lambda+1) \leq \sigma$ and $0 < p_{CHM} < 1$ with $p_{CHM} = \Theta(1)$. The BCA* (Definition 4) is always perfect after the first $O(n)$ steps for the next $n^{O(1)}$ steps with probability converging to 1.*

Proof. This follows from Theorem 40 since the one offspring subject to standard bit mutation is not relevant for the proof. □

We do not perform any experiments for the above cases since these do not add much to our theoretical analyses. All algorithms are perfect and remain perfect after some initial optimisation phase. Thus, all plots are very similar to the one in Figure 5c.

5.3.2 Long and Very Long Stable Intervals

We have seen in Section 5.3.1 that on the very fast execution platform ($\sigma = \Theta(n^3)$) all three considered algorithms are always perfect after some initial optimisation steps. If we increase the length of the stable interval to any larger value (in particular to $n^{3/2}$ and n^3) nothing changes significantly. For each statement from Section 5.3.1 there is a corresponding equal statement for the longer stable phases.

6 Conclusions

Dynamic optimisation by means of randomised search heuristics like evolutionary algorithms and artificial immune systems is an important topic because it is practically relevant. This makes it important to build a firm theoretical foundation. We have contributed to this foundation by pointing out the importance of recognising the speed of the execution platform running the randomised search heuristics as an important parameter in its own right. We have also contributed by defining a bi-stable dynamic example problem that shares important properties with ONEMAX and real dynamic optimisation problems. Using the perspective of fixed budget computations we have proven a large number of concrete theoretical results for mutation-based evolutionary algorithms and two variants of the B-cell algorithm on our new example problem. We summarise these theoretical results in Table 1.

We hope that the introduction of the speed σ will be picked up in future theoretical studies of dynamic optimisation. We also hope that our example problem proves to be a useful benchmark for other randomised search heuristics that are accessible to theoretical studies.

Clearly, all our theoretical results are relatively rough and could be refined in future research. The specific way the optimum moves in the non-stable phases of our example problem tends to be helpful for the contiguous hypermutations used in the

length of stable intervals / speed of exec. platform	**short: $\tau = n$**	**long: $\tau = n^{3/2}$**	**very long: $\tau = n^3$**
slow $\sigma = \Theta(1)$	EA (1): always bad (Thm. 5) BCA (1): always bad (Thm. 6) BCA* (1): always bad (Thm. 7)	EA (1): repeating perfect-bad (Thm. 9) BCA (1): always bad (Thm. 8) BCA* (1): repeating perfect-bad (Thm. 10)	EA (1): repeating perfect-bad (Cor. 11) BCA (1): repeating perfect-mediocre or bad (Thm. 12) BCA* (1): repeating perfect-bad (Cor. 13)
fast $\sigma = \Theta\left(n^{3/2}\right)$	EA (1): always perfect (Thm. 14) EA ($n^{3/2}$): always perfect (Thm. 17) BCA (1): repeating perfect-bad (Thm. 15) BCA ($n^{3/2}$): repeating perfect-mediocre or bad (Thm. 19) BCA* (1): always perfect (Thm. 16) BCA* ($n^{3/2}$): repeating perfect-mediocre or bad (Thm. 20)	EA (1): always perfect (Cor. 21) EA ($n^{3/2}$): always perfect (Cor. 24) BCA (1): repeating perfect-mediocre or bad (Thm. 22) BCA ($n^{3/2}$): repeating perfect-mediocre or bad (Thm. 25) BCA* (1): always perfect (Cor. 23) BCA* ($n^{3/2}$): repeating perfect-mediocre or bad (Thm. 26)	EA (1): always perfect (Cor. 27) EA ($n^{3/2}$): always perfect (Cor. 30) BCA (1): repeating perfect-mediocre or bad (Cor. 28) BCA ($n^{3/2}$): repeating perfect-mediocre or bad (Cor. 31) BCA* (1): always perfect (Cor. 29) BCA* ($n^{3/2}$): repeating perfect-mediocre or bad (Cor. 32)
very fast $\sigma = \Theta(n^3)$	EA (1): always perfect (Cor. 33) EA ($n^{3/2}$): always perfect (Cor. 36) EA (n^3): always perfect (Cor. 39) BCA (1): always perfect (Cor. 34) BCA ($n^{3/2}$): always perfect (Thm. 37) BCA (n^3): always perfect (Thm. 40) BCA* (1): always perfect (Cor. 35) BCA* ($n^{3/2}$): always perfect (Cor. 38) BCA* (n^3): always perfect (Cor. 41)	EA (1): always perfect EA ($n^{3/2}$): always perfect EA (n^3): always perfect BCA (1): always perfect BCA ($n^{3/2}$): always perfect BCA (n^3): always perfect BCA* (1): always perfect BCA* ($n^{3/2}$): always perfect BCA* (n^3): always perfect	EA (1): always perfect EA ($n^{3/2}$): always perfect EA (n^3): always perfect BCA (1): always perfect BCA ($n^{3/2}$):always perfect BCA (n^3): always perfect BCA* (1): always perfect BCA* ($n^{3/2}$): always perfect BCA* (n^3): always perfect

Table 1: Overview of all theoretical results; for the precise formulations see the Theorems in the main text. The format is A (v) where A is an algorithm used with the parameter values $\mu = \Theta(1)$ and $\lambda = \Theta(v)$.

two B-cell algorithm variants. Jansen and Zarges (2014c) demonstrate that in comparison to standard bit mutations used in evolutionary algorithms the somatic contiguous hypermutations employed by the B cell algorithm have advantages on ONEMAX when the global optimum is far away. This motivated us to investigate if this advantage transfers into a tangible advantage in dynamic optimisation for our example problem. In our analytical framework no such clear advantage can be observed in our theoretical results. Looking at the empirical results it appears that the worst function values the algorithms exhibit are always worst for the EA. This perspective of guaranteed worst case solution quality is exactly captured by the perspective of approximation where one considers the worst case ratio of the quality of an optimal solution and the solution delivered by an algorithm. It would be interesting to reconsider the situation with an emphasis of the approximation performance of the randomised search heuristics under consideration. From a practical point of view it would be important to know which kind of search heuristic tends to promise better approximations in such a scenario.

It also remains an open problem to identify example problems, preferably not too artificial ones, where artificial immune systems excel from the analytical perspective adopted in this article.

References

Auger, A. and Doerr, B., editors (2011). *Theory of Randomized Search Heuristics*. World Scientific Review.

Branke, J. (2002). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers.

Branke, J. and Wang, W. (2003). Theoretical analysis of simple evolution strategies in quickly changing environments. In *Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 537–548.

Bu, Z. and Zheng, B. (2010). Perspectives in dynamic optimization evolutionary algorithm. In *Proceedings of the 5th International Conference on Advances in Computation and Intelligence*, pages 338–348.

Doerr, B., Jansen, T., Witt, C., and Zarges, C. (2013). A method to derive fixed budget results from expected optimisation times. In *Genetic and Evolutionary Computation Conference (GECCO 2013)*, pages 1581–1588.

Droste, S. (2002). Analysis of the (1+1) EA for a dynamically changing onemax-variant. In *Proc. of CEC'02*, pages 55–60. IEEE Press.

Droste, S. (2003). Analysis of the (1+1) EA for a dynamically bitwise changing onemax. In *Genetic and Evolutionary Computation Conference (GECCO 2003)*, LNCS 2723, pages 909–921. Springer.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Jansen, T. (2013). *Analyzing Evolutionary Algorithms. The Computer Science Perspective*. Springer.

Jansen, T., Oliveto, P. S., and Zarges, C. (2011). On the analysis of the immune-inspired b-cell algorithm for the vertex cover problem. In *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*, pages 17–131.

Jansen, T. and Schellbach, U. (2005). Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in the lattice. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, pages 841–848. ACM Press.

Jansen, T. and Zarges, C. (2011). Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theor. Comput. Sci.*, 412(6):517–533.

Jansen, T. and Zarges, C. (2012). Fixed budget computations: A different perspective on run time analysis. In *Genetic and Evolutionary Computation Conference (GECCO 2012)*, pages 1325–1332.

Jansen, T. and Zarges, C. (2014a). Evolutionary algorithms and artificial immune systems on a bi-stable dynamic optimisation problem. In *Genetic and Evolutionary Computation Conference (GECCO 2014)*, pages 975–982. ACM.

Jansen, T. and Zarges, C. (2014b). Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, 545:39–58.

Jansen, T. and Zarges, C. (2014c). Reevaluating immune-inspired hypermutations using the fixed budget perspective. *IEEE Transactions on Evolutionary Computation*, 18:674–688.

Kelsey, J. and Timmis, J. (2003). Immune inspired somatic contiguous hypermutations for function optimisation. In *Genetic and Evolutionary Computation Conference (GECCO 2003)*, pages 207–218.

Kötzing, T., Lissovoi, A., and Witt, C. (2015). (1+1) EA on generalized dynamic onemax. In *Foundations of Genetic Algorithms (FOGA 2015)*, pages 40–51.

Kötzing, T. and Molter, H. (2012). ACO beats EA on a dynamic pseudo-boolean problem. In *Proceedings of the 12th International Conference on Parallel Problem Solving From Nature (PPSN 2012)*, pages 113–122.

Lehmann, E. L. (2006). *Nonparametrics. Statistical Methods Based on Ranks*. Springer.

Lengler, J. and Spooner, N. (2015). Fixed budget performance of the (1+1)-EA on linear functions. In *Foundations of Genetic Algorithms (FOGA 2015)*, pages 52–61.

Lissovoi, A. and Witt, C. (2013). Runtime analysis of ant colony optimization on dynamic shortest path problems. In *Genetic and Evolutionary Computation Conference (GECCO 2013)*, pages 1605–1612.

Lissovoi, A. and Witt, C. (2014). MMAS vs. population-based EA on a family of dynamic fitness functions. In *Genetic and Evolutionary Computation Conference (GECCO 2014)*, pages 1399–1406.

Morrison, R. W. (2004). *Designing Evolutionary Algorithms for Dynamic Environments*. Springer.

Nallaperuma, S., Neumann, F., and Sudholt, D. (2014). A fixed budget analysis of randomized search heuristics for the traveling salesperson problem. In *Genetic and Evolutionary Computation Conference (GECCO 2014)*, pages 807–814.

Neumann, F. and Witt, C. (2010). *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.

Nguyen, T. T., Yang, S., and Branke, J. (2012). Evolutionary dynamic optimization: a survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24.

Oliveto, P. S. and Zarges, C. (2013). Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. In *Genetic and Evolutionary Computation Conference (GECCO 2013)*, pages 837–844. ACM.

Oliveto, P. S. and Zarges, C. (2014). Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. *Theor. Comput. Sci.* To appear.

Rohlfshagen, P., Lehre, P. K., and Yao, X. (2009). Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1713–1720. ACM Press.

Stanhope, S. A. and Daida, J. M. (1999). (1+1) genetic algorithm fitness dynamics in a changing environments. In *CEC*, pages 1851–1858. IEEE.

Tifenbach, R. M. (2013). A combinatorial approach to nearly uncoupled Markov chains i: Reversible Markov chains. *Electronic Transactions on Numerical Analysis*, 40:120–147.

Tinos, R. and Yang, S. (2010). An analysis of the XOR dynamic problem generator based on the dynamical system. In *Parallel Problem Solving From Nature (PPSN)*, pages 274–283.

Tinos, R. and Yang, S. (2013). Analyzing evolutionary algorithms for dynamic optimization problems based on the dynamical systems approach. In Yang, S. and Yao, X., editors, *Evolutionary Computation for Dynamic Optimization Problems*, pages 241–267. Springer.

Tinos, R. and Yang, S. (2014). Analysis of fitness landscape modifications in evolutionary dynamic optimization. *Information Sciences*, 282:214–236.

Vose, M. D. (1998). *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press.

Weicker, K. (2003). *Evolutionary Algorithms and Dynamic Optimization Problems*. Der Andere Verlag.

Yang, S. and Yao, X. (2005). Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9:815–834.

Yang, S. and Yao, X., editors (2013). *Evolutionary Computation for Dynamic Optimization Problems*. Springer.