Theoretical Computer Science 561 (2015) 37–56

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change ☆

Pietro S. Oliveto [a,1], Christine Zarges [b,*]

[a] Department of Computer Science, The University of Sheffield, Regent Court, Portobello, Sheffield S1 4DP, United Kingdom
[b] School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom

ARTICLE INFO

*Article history:*
Received 15 November 2013
Received in revised form 1 July 2014
Accepted 15 October 2014
Available online 23 October 2014

*Keywords:*
Evolutionary dynamic optimisation
Diversity mechanisms
Runtime analysis
Populations

ABSTRACT

Evolutionary dynamic optimisation has become one of the most active research areas in evolutionary computation. We consider the Balance function for which the poor expected performance of the $(1+1)$ EA at low frequencies of change has been shown in the literature. We analyse the impact of populations and diversity mechanisms towards the robustness of evolutionary algorithms with respect to frequencies of change. We rigorously prove that there exists a sufficiently low frequency of change such that the $(\mu+1)$ EA without diversity requires exponential time with overwhelming probability for sublinear population sizes. The same result also holds if the algorithm is equipped with a genotype diversity mechanism. Furthermore we prove that a crowding mechanism makes the performance of the $(\mu+1)$ EA much worse (i.e., it is inefficient for any population size). On the positive side we prove that, independently of the frequency of change, a fitness-diversity mechanism turns the runtime from exponential to polynomial. Finally, we show how a careful use of fitness-sharing together with a crowding mechanism is effective already with a population of size 2. We shed light through experiments when our theoretical results do not cover the whole parameter range.



## 1. Introduction

Many real-world problems are subject to changing conditions over time. The field concerned with the application of Evolutionary Algorithms (EAs) to this class of problems is called *Evolutionary Dynamic Optimisation* (EDO). Especially in recent years EDO has attracted lots of research and has become one of the most active areas in evolutionary computation. Not surprisingly several monographs [2–5] and survey papers [6–8] on the topic have recently been published. Differently from static optimisation where the task is to find the global optimum in as few steps as possible, addressing *Dynamic Optimisation Problems* (DOPs) requires an optimisation algorithm not only to locate the optimum of a given problem, but also to *track* the optimal solution over time when the problem changes. While populations and related operators (i.e., crossover, stochastic selection, diversity mechanisms, etc.) are the main distinguishing features of bio-inspired search heuristics from other classes of heuristics for static optimisation, they are considered *essential* in the process of detecting changes and tracking the optimum after a change in the objective function has occurred. The most common features incorporated into EDO algorithms

☆ Parts of the results appeared in the Proceedings of 15th Annual Conference on Genetic and Evolutionary Computation (GECCO 2013) [1].

* Corresponding author.
*E-mail addresses:* P.Oliveto@sheffield.ac.uk (P.S. Oliveto), C.Zarges@cs.bham.ac.uk (C. Zarges).

[1] P.S. Oliveto was supported in part by EPSRC under grant N. EP/H028900/1.

http://dx.doi.org/10.1016/j.tcs.2014.10.028
0304-3975/

are diversity mechanisms (either triggered when a change is detected or maintained throughout the evolutionary process), memory-based and prediction-based approaches [7]. While the latter approaches are applied when it is known that the dynamics of the problem are periodical or recurrent, i.e., the optima may return to regions near the previous locations (memory approaches) or have some predictable patterns (prediction approaches), diversity mechanisms are meant to be more general and applied even when very limited knowledge about the problem is available. Commonly used mechanisms to enhance the population diversity include random immigrant introduction [9], fitness sharing [10], genotype diversity [11] and multi-populations (see [12,13] amongst many others).

In contrast to EA theory in the static domain which has rapidly grown in recent years [14–17], only very few theoretical results are available concerning EDO. Droste [18,19] analysed the $(1+1)$ EA on the dynamic version of the OneMax problem where the fitness function changes after each function evaluation according to some probability $p$. Jansen and Schellbach [20] analysed a $(1+\lambda)$ EA for a simple lattice problem. More recently Rohlfshagen et al. [21] analysed how the performance of the $(1+1)$ EA is affected by the *magnitude* and *frequency* of change in two counter-intuitive scenarios. They present an instance class called Magnitude where the algorithm is efficient if the magnitude of change is large while it requires exponential expected runtime to track the optimum if the magnitude of change is small. Concerning the frequency of change they present an instance class called Balance where the $(1+1)$ EA is efficient if the frequency of change is high while it requires exponential expected runtime if the frequency is low. Finally, very recently, Kötzing and Molter [22] constructed a pseudo-boolean instance class where a simple ant colony optimisation system can track the optimum while the $(1+1)$ EA gets lost, and Chen et al. [23] studied the impact of self-adaptive mutation rates for EDO. From these analyses great insight can be gained towards understanding how evolutionary processes react to changes in the objective function and how traditional analytical proof methods can be applied in the dynamic settings. However, by only considering algorithms using single individuals it is hard to relate the available results to the performance of the more sophisticated EDO algorithms used in experimental studies and practical applications. In fact researchers in the EDO community have emphasised the importance of achieving such results in the latest survey paper (see Section 5 in [7]).

In this paper we present a first step towards directing theoretical work to analyse EAs equipped with populations and the mechanisms that are considered essential to tackle dynamic problems in the EDO literature. In particular, we will consider the simplest population-based EA, the $(\mu+1)$ EA as well as a local search variant (Algorithm 5), and analyse its performance combined with different commonly used diversity mechanisms (in their simplest version) and verify how effective they are in overcoming the problems encountered by single individual EAs. Rather than considering new example functions especially constructed to serve our purposes, we analyse the $(\mu+1)$ EA on the Balance function, for which the performance of the $(1+1)$ EA is known [21]. This function class was introduced as a counter-intuitive example that is hard to optimise at low frequencies of change and easy at high frequencies. Our goal is to analyse whether more realistic EAs using a population and a diversity mechanism can efficiently optimise the Balance function independent of the frequency of change. Ideally the population should be able to efficiently optimise the function for any value of $\tau$, i.e., in the particular case of Balance, even at very low frequencies of change.

The rest of the paper is structured as follows. In Section 2 we introduce the Balance problem and the XoR benchmark framework used by Rohlfshagen et al. [21] to impose the dynamics on the function. In Section 3 we show that, if the population size $\mu$ is not too large, then there exists a sufficiently low frequency of change such that the $(\mu+1)$ EA requires exponential time with overwhelming probability to optimise Balance (i.e., the $(\mu+1)$ EA is not as robust towards $\tau$ as desired for sublinear population sizes). In Sections 4 and 5 it is proved that by adding respectively a genotype and a crowding diversity mechanism the $(\mu+1)$ EA still cannot optimise Balance with low frequencies of change efficiently. Then we turn to positive results. In Section 6 we rigorously prove how a fitness diversity mechanism allows the efficient optimisation of Balance with high probability independent of the frequency of change with population sizes $\mu$ that are at least sublinear. In Section 7 we show how a carefully used fitness sharing mechanism combined with a crowding mechanism can make the $(\mu+1)$ EA efficient for any frequency of change $\tau$ even by just using the most basic mutation operator and a population size as small as $\mu=2$. In Section 8 we present some experiments to fill in the gaps left by our theoretical results. We first look at the algorithms we have proved to be inefficient for not too large population sizes. In particular, we investigate how large the population sizes have to be to turn the algorithms into efficient optimisers for Balance at any frequency of change. Afterwards, we study to what extent crowding and fitness sharing need to be combined to make the $(\mu+1)$ EA effective for Balance. In the last section we discuss our conclusions and future work.

## 2. Definitions and framework

We use the XoR framework to impose dynamics to the stationary Balance function in exactly the same way as done in [21]. Although, the first theoretical paper to use the XoR framework explicitly was [21] the few previous works for DOPs essentially use an identical framework.

The framework, as defined in [24], can be used with any stationary pseudo-Boolean function by means of a bit-wise *exclusive-or* operation that is applied to each search point $x \in \{1,0\}^n$ prior to each function evaluation. The dynamic fitness function is simply $f(x(t) \oplus m(\pi))$ where $t$ is the number of generations, $\oplus$ the *xor* operator and $m(\pi) \in \{0,1\}^n$ is a binary mask which initially is equivalent to $0^n$ and is generated as $m(\pi) := m(\pi-1) \oplus p(\pi)$. Here $p(\pi) \in \{0,1\}^n$ is a randomly created template containing exactly $\lfloor \rho n \rfloor$ 1-bits, where $\rho \in (0,1]$ defines the $\rho n$ bits to be inverted. The period index