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

Oliveto, Pietro; Zarges, Christine

DOI:
10.1016/j.tcs.2014.10.028

License:
Other (please specify with Rights Statement)

Document Version
Peer reviewed version

Citation for published version (Harvard):
Oliveto, PS & Zarges, C 2015, 'Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change', Theoretical Computer Science, vol. 561, pp. 37-56. https://doi.org/10.1016/j.tcs.2014.10.028

Analysis of Diversity Mechanisms for Optimisation in Dynamic Environments with Low Frequencies of Change

Pietro S. Oliveto, Christine Zarges

PII: S0304-3975(14)00820-2
DOI: 10.1016/j.tcs.2014.10.028
Reference: TCS 9925

To appear in: *Theoretical Computer Science*

Received date: 15 November 2013
Revised date: 1 July 2014
Accepted date: 15 October 2014

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*

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, independent 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.

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

Email addresses: P.Oliveto@sheffield.ac.uk (Pietro S. Oliveto), C.Zarges@cs.bham.ac.uk (Christine Zarges)

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

Keywords: Evolutionary dynamic optimisation, diversity mechanisms, runtime analysis, populations

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, 3, 4, 5] and survey papers [6, 7, 8] on the topic have recently been published. Different to 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 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, 15, 16, 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]).

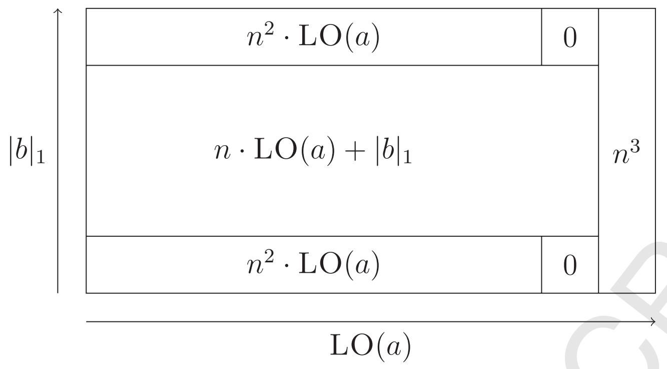
In this paper we present a first step towards directing theoretical work to analyse EAs equipped with populations and the mechanisms that are considered essential to tackle dynamic problems in the EDO literature. In particular, we will consider the simplest population-based EA, the $(\mu+1)$ EA as well 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 τ, 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 μ 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 τ 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 μ 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 τ 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 $\lfloor \rho n \rfloor$



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

Figure 1: Visualisation of BALANCE [21].

1-bits, where $\rho \in (0, 1]$ defines the ρn bits to be inverted. The period index $\pi = \lceil t/\tau \rceil$ is determined by the duration $\tau > 0$ between changes. Hence, the *magnitude of change* is unambiguously defined by the parameter ρ (i. e., referring to the number of bits a search point is rotated by).

The frequency of change (i. e., defined by $1/\tau$) determines how often the problem changes. Intuitively the higher the frequency of change, the harder it is to optimise the dynamic function since less time is available at each time period to find the new global optimum. Even if the optimum was to remain stationary, while the rest of the search space changes, it is assumed that high frequencies of change increase the problem difficulty due to the higher quantity of uncertainty introduced by the frequent changes. Rohlfshagen et al. [21] introduced the following BALANCE function to disprove this assumption (see also Figure 1).

Definition 1. *Let $a, b \in \{0,1\}^{n/2}$ and $x = ab \in \{0,1\}^n$. Then,*

$$\textsc{Balance}(x) = \begin{cases} n^3 & \text{if } \mathrm{LO}(a) = n/2, \text{ else} \\ |b|_1 + n \cdot \mathrm{LO}(a) & \text{if } n/16 < |b|_1 < 7n/16, \text{ else} \\ n^2 \cdot \mathrm{LO}(a) & \text{if } |a|_0 > \sqrt{n}, \text{ else} \\ 0 & \text{if otherwise} \end{cases}$$

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

Note, that $\mathrm{LO}(x)$ is often referred to as LEADINGONES while $|x|_1$ is also known as $\textsc{OneMax}(x) = \sum_{i=1}^{n} x_i$. Each search point for BALANCE consists of a prefix of length $n/2$ and a suffix of the same length. The fitness of a search point is determined by the number of leading 1-bits in the prefix and by the number of 1-bits in the suffix. A globally optimal search point has the maximal value $n/2$ of leading 1-bits in the prefix. However, before reaching the global optimum the algorithm may reach the two trap regions corresponding to search points with less than $n/16$ 0-bits or less than $n/16$ 1-bits in the suffix. The trap regions and the global optimum are separated by a region of 0-fitness of length $\sqrt{n}$ that makes it prohibitive for EAs to reach the optimum from the traps. Throughout the paper we will refer to the trap corresponding to search points with less than $n/16$ 0-bits in the suffix as the *upper trap* and to the trap corresponding to search points with less than $n/16$ 1-bits in the suffix as the *lower trap*.

We are mainly interested in the *optimisation time* T of the considered algorithm, which is defined as the number of generations needed until a global optimum is found for the first time. Since T is a random variable we investigate its mean $\mathrm{E}(T)$ as well as information on its distribution, i. e., $\mathrm{Prob}(T < t)$ or $\mathrm{Prob}(T > t)$ for relevant points of time t. We say that an event A occurs with *overwhelming probability* (w. o. p.) if $\mathrm{Prob}(A) = 1 - 2^{-\Omega(n)}$. Note, that we consider asymptotic optimisation times throughout this paper. This means that all our results hold for all values of n that are sufficiently large.

Following the XoR framework, cyclical dynamics were applied in [21] to the function using the following mask m as a function of the period index π:

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

Hence, only the suffix of the search points is affected and the magnitude of change is $n/2$. During odd periods π, the fitness increases with the increase of suffix 0-bits while for even π it increases with the increase of 1-bits. Rohlfshagen et al. [21] proved that the (1+1) EA would balance along the centre of the suffix (the centre of the y-axis in the figure) and efficiently optimise the leading 1-bits in the prefix for a high frequency of change (i. e., $\tau = 2$). This happens because the algorithm does not have enough time to optimise the OneMax suffix before the suffix-dependent part of the function changes into ZeroMax and vice versa. On the other hand, for sufficiently small frequency (i. e., $\tau > 40n$) with at least constant probability the (1+1) EA will be attracted into one of the trap regions before reaching the optimum, implying expected exponential optimisation time. In the rest of the paper we will examine whether populations equipped with diversity mechanisms can be robust enough to optimise Balance independent of the value of the frequency of change τ (i. e., even for very low frequencies).

3. No Diversity

Rohlfshagen et al. have shown that the expected runtime of the (1+1) EA on Balance is exponential if the frequency of change is low [21]. In this section we show that also a population-based EA such as the (μ+1) EA (see Algorithm 1) suffers from this problem if the population size is not too large. We will see that in this case the whole population will get stuck in one of the traps before any search point (also called individual) reaches the global optimum. In particular, we will prove that for sublinear population sizes $\mu \leq n^{1/2-\epsilon}$ there exists a sufficiently low frequency of change such that the optimisation time of the (μ+1) EA is exponential with overwhelming probability.

Note that we use set notation throughout this paper but allow for multiple copies of an individual in a population. We use $P \setminus \{z\}$ to indicate the removal of one specific instance z of an individual from the population P.

Algorithm 1 (μ+1) EA

```
1: Let $t := 0$ and choose $x_1, \ldots, x_\mu \in \{0,1\}^n$ independently uniformly at
   random (u. a. r.); $P_t := \{x_1, \ldots, x_\mu\}$.
2: repeat
3:    Choose $x \in P_t$ u. a. r. and set offspring $y := x$.
4:    Flip each bit in $y$ independently with probability $1/n$.
5:    Choose $z \in P_t$ with minimal fitness u. a. r.
6:    if $f(y) \geq f(z)$ then
7:       Let $P_{t+1} := P_t \setminus \{z\} \cup \{y\}$.
8:    else
9:       Let $P_{t+1} := P_t$.
10:   end if
11:   Let $t := t + 1$.
12: until some termination condition is met
```

The proof strategy is as follows. We first calculate the expected runtime for the whole population to reach the trap under the assumption that the global optimum is not found first (Lemma 2). Then, in Theorem 3 we will prove that with overwhelming probability the time for any individual to reach the optimum is much higher than the time required for the whole population to get trapped. Then the main result follows because the time required to escape from the trap is exponential. Along the way we will need the following helper lemma.

Lemma 1. *Let j be the number of leading 1-bits in the individual of the population with highest fitness. Assuming no other leading 1-bits are created first, for any time period τ the expected time for the (μ+1) EA to have all individuals with j leading 1-bits is at most $7\mu \log \mu$ iterations.*

Proof. Given that we have i individuals with j leading 1-bits, the probability we create another one is

$$P_{copy} \geq \frac{i}{\mu}\left(1 - \frac{1}{n}\right)^n \geq \frac{i}{4\mu}$$

since it is sufficient to select one of the i individuals and flip none of the bits. As long as there exists an individual with less leading 1-bits, the newly created individual with j leading 1-bits will be accepted. Hence, the expected time for the whole population to have j leading 1-bits is bounded from above by

$$\sum_{i=1}^{\mu} \frac{4\mu}{i} \leq 4\mu \cdot (\ln(\mu) + 1) \leq 7\mu \log \mu$$

□

Lemma 2. *Let $\mu \leq n^{1/2-\epsilon}$ and $\tau > 19\mu n$. Assuming the global optimum is not found first, in expected time less than $19\mu n$ iterations all the individuals of the (μ+1) EA on* Balance *have a suffix b with $|b|_1 < n/16$ or $|b|_1 > 7n/16$ (i. e., all individuals are in the trap).*

Proof. W. l. o. g., the goal is to prove that the whole population reaches the lower trap in time at most $cnn^{1/2-\epsilon}$. The lower trap is the region of the search space with less than $n/16$ 1-bits in the suffix (i. e., $|b|_1 < n/16$) and less than $n/2 - \sqrt{n}$ leading 1-bits in the prefix. We consider the individual x with least number of 0-bits in the suffix (i. e., independent from the number of leading 1-bits it has). All the other individuals have more 0-bits in the suffix. If x gets removed from the population, or "overtaken" by an individual with less 0-bits in the suffix, we will track the number of 0-bits in the suffix of the new

x (i. e., the individual with less 0-bits in the suffix in the new population). When x has reached the trap also the rest of the population will be in the trap.

We consider phases and count the expected number of 0-bits gained by x in a phase. The first phase starts after initialisation and lasts until a leading 1-bit is added to an individual. Each following phase will end when the next leading one is created in some individual. Each phase has an expected duration of n steps (i. e., the probability that a leading 1-bit is created is $1/n$ whatever the selected individual).

Since no leading 1-bits are created during a phase, the only way a negative drift may occur is when an individual with more leading 1-bits than x is selected and a new individual with the same number of leading 1-bits is obtained. In this case two different events may happen which influence our process:

1. x is removed from the population and the new individual has less 0-bits than x (i. e., hence it becomes the new x);
2. the new individual has less 0-bits than x (i. e., hence it becomes the new x).

We calculate the expected number of 0-bits of the new individual pessimistically assuming its parent has the same number of 0-bits as x. The negative drift (i. e., the expected decrease in number of 0-bits in one step) is:

$$E[\Delta^-] > -\left(\frac{n/2-i}{n} - \frac{i}{n}\right) > -\frac{3}{8}$$

where i is the number of 1-bits in the suffix of x.

Since by Lemma 1 in time $7\mu \log \mu$ all the individuals have the same number of leading 1-bits as the best individual, the negative drift in a whole phase can be at most $-(3/8) \cdot 7\mu \log \mu$.

For an increase of 0-bits to occur it is sufficient that an individual is selected, a 1-bit is flipped into a 0-bit and nothing else flips. Hence the expected number of 0-bits achieved (i. e., the positive drift) in one step is bounded by

$$E[\Delta^+] \geq \frac{n/16}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{48}.$$

Since the individuals are selected for reproduction uniformly, the expected increase in 0-bits per individual is $1/(48\mu)$, which is the positive drift (i. e., the expected increase of 0-bits in x).

Finally, when the leading 1-bit flips at the end of the phase the expected decrease in 0-bits is at most (and pessimistically assuming this individual replaces x as the one with minimal ZeroMax value in the suffix):

$$E[\Delta^-|LO] > -\left(\frac{n/2-i}{n} - \frac{i}{n}\right) > -\frac{3}{8}$$

Then the total expected increase of 0-bits in x in a whole phase of expected duration of n steps is

$$E[\Delta] > \frac{n}{48\mu} - \frac{3}{8}\cdot(7\mu\log\mu+1) \geq \frac{n^{1/2+\epsilon}}{48} - 3n^{1/2-\epsilon}\log n^{1/2-\epsilon} > \frac{n^{1/2+\epsilon}}{49}$$

because $\mu < n^{1/2-\epsilon}$.

Since the total number of 0-bits that need to be collected to reach the trap is at most $(3/8)n$, the expected number of phases is at most

$$\frac{(3/8)n}{n^{1/2+\epsilon}/49} = \frac{49\cdot 3}{8}\cdot n^{1/2-\epsilon} < 19n^{1/2-\epsilon}$$

Given that each phase lasts n steps in expectation the trap is reached in expected time $E[T_{Trap}] \leq 19n\cdot n^{1/2-\epsilon}$. □

Theorem 1. *Let $\tau > 20\mu n$ and $\mu \leq n^{1/2-\epsilon}$. Then expected time for the (μ+1) EA to optimise* Balance *is at least $n^{\Omega(\sqrt{n})}$. Let $\tau > 38\mu n^{3/2}$ and $\mu \leq n^{1/2-\epsilon}$. Then the ($\mu$+1) EA requires at least $n^{\Omega(\sqrt{n})}$ steps with overwhelming probability.*

Proof. By Lemma 2 the expected time for the whole population to reach the trap is at most $19n\mu \leq 19\cdot n\cdot n^{1/2-\epsilon}$. By applying Markov's inequality iteratively in $\sqrt{n}$ separate phases of length $2\cdot 19\cdot n\cdot n^{1/2-\epsilon}$ each, we get that the population has not converged to the trap in $2\cdot 19\cdot n^{2-\epsilon}$ steps with probability at most $2^{-\sqrt{n}}$.

Using similar reasoning to the proof of Witt [25] and that $\mu \leq n^{1/2-\epsilon}$, $m := n/2 - \sqrt{n}$ leading 1-bits in the prefix are optimised in less than m^2 iterations of the (μ+1) EA with probability less than $2^{-\Omega(m)} = 2^{-\Omega(n)}$. Hence with overwhelming probability $1 - 2^{-\Omega(n)}\cdot 2^{-\sqrt{n}}$ the population reaches the trap before it optimises the leading 1-bits in the prefix. Conditional to these events, the time to reach the optimum is at least $n^{\Omega(\sqrt{n})}$ since at least $\sqrt{n}$ bits need to be flipped to overcome the zero fitness region and reach the optimum.

To prove the first statement it is sufficient to apply Markov's inequality once to show that with probability at least 1/20 the population is trapped in $20\mu n$ steps. Then the expected runtime is bounded by

$$E(T) \geq (1/20)(1 - 2^{-\Omega(n)})n^{\Omega(\sqrt{n})}$$

□

4. Genotype Diversity

In the previous section we showed that there exists a sufficiently low frequency of change such that the (μ+1) EA with sublinear population sizes cannot optimise BALANCE in polynomial time with overwhelming probability. Now we analyse the effects of adding *genotype diversity* to the algorithm (see Algorithm 2). This mechanism simply does not allow multiple individuals of the population to have the same genotype and is probably the most simple mechanism proposed in the literature.

Algorithm 2 (μ+1) EA with genotype diversity

```
1: Let $t := 0$ and choose $x_1, \dots, x_\mu \in \{0,1\}^n$ independently uniformly at
   random (u. a. r.); $P_t := \{x_1, \dots, x_\mu\}$.
2: repeat
3:   Choose $x \in P_t$ u. a. r. and set offspring $y := x$.
4:   Flip each bit in $y$ independently with probability $1/n$.
5:   if $y \notin P_t$ then
6:     Choose $z \in P_t$ with minimal fitness u. a. r.
7:     if $f(y) \geq f(z)$ then
8:       Let $P_{t+1} := P_t \setminus \{z\} \cup \{y\}$.
9:     else
10:      Let $P_{t+1} := P_t$.
11:    end if
12:  else
13:    Let $P_{t+1} := P_t$.
14:  end if
15:  Let $t := t + 1$.
16: until some termination condition is met
```

The described algorithm has been previously analysed theoretically. Storch and Wegener analysed its behaviour on royal road functions [26]. Friedrich et al. [27] showed that the genotype diversity mechanism is not powerful enough to optimise the two branches of the simple bimodal function TwoMax.

In the following we show that the diversity mechanism is not effective for Balance either. The same proof strategy as in the previous section will lead to the main result and we see that not allowing copies in the population does not significantly change the behaviour of the (μ+1) EA. Two helper lemmas will be proved first. In the next lemma we achieve a bound on the positive drift towards the trap. Lemma 4, instead, is the analogue of Lemma 1 for the (μ+1) EA with genotype diversity.

Lemma 3. *Let $\mu \le n^{1/2-\epsilon}$ and consider the (μ+1) EA with genotype diversity for* Balance. *The expected gain in number of 0-bits in the suffix in one step of the individual x with least 0-bits in the suffix (i. e., the positive drift) is at least $1/(49\mu)$.*

Proof. For a positive drift to occur it is necessary that in the selected individual at least one 1-bit is flipped into a 0-bit. Given the genotype diversity mechanism, if an individual is created with the same genotype as another one already in the population, then the new individual will not be accepted. We pessimistically assume that all the individuals in the population are neighbours of Hamming distance 1 to each other. Then, given that these neighbours are μ, there are always at least $n/16 - \mu$ bits that, if flipped, lead to individuals with a genotype not present in the current population. Hence, the expected increase in one step of 0-bits in the suffix is bounded by

$$E[\Delta^+] \ge \frac{n/16-\mu}{n} \cdot \left(1-\frac{1}{n}\right)^{n-1} \ge \left(\frac{1}{16} - \frac{1}{n^{1/2+\epsilon}}\right)\frac{1}{e} \ge \frac{1}{49}.$$

Since the individuals are selected for reproduction uniformly, the expected increase in 0-bits per individual is $1/(49\mu)$, which is the positive drift (i. e., the expected increase of 0-bits in x). □

Lemma 4. *Let $\mu \le n^{1/2-\epsilon}$ and let j be the number of leading 1-bits in the individual of the population with highest fitness. Assuming no other leading 1-bits are created first, for any time period τ the expected time for the (μ+1) EA with genotype diversity to have all individuals with j leading 1-bits is at most $8e\mu \log \mu$ iterations.*

Proof. Each individual with j leading 1-bits has at least $n/2$ neighbours of Hamming distance 1 (i. e., the ones corresponding to the bits in the suffix) that have higher fitness compared to an individual in the population with fewer leading 1-bits. In the worst case $\mu - 1$ of these are already in the population, hence an individual with any of their genotypes will not be accepted due to the diversity mechanism. Then, given that an individual with j leading 1-bits has been selected for reproduction, the probability to create another individual with j leading 1-bits is at least

$$\frac{n/2-\mu}{n}\left(1-\frac{1}{n}\right)^{n-1} \geq \left(\frac{1}{2}-\frac{1}{n^{1/2+\epsilon}}\right)\frac{1}{e} \geq \frac{1}{3e}$$

which implies that given i individuals in the population with j leading 1-bits the probability to create another one is at least $i/(3e\mu)$. As long as there exists an individual with less leading 1-bits, the newly created individual with j leading 1-bits will be accepted. Hence, the expected time for the whole population to have j leading 1-bits is bounded from above by

$$\sum_{i=1}^{\mu}\frac{3e\mu}{i} \leq 3e\mu\cdot(\ln(\mu)+1) \leq 8e\mu\log\mu$$

□

Theorem 2. *Let $\tau > 20\mu n$ and $\mu \leq n^{1/2-\epsilon}$. Then expected time for the (μ+1) EA with genotype diversity to optimise* Balance *is at least $n^{\Omega(\sqrt{n})}$. Let $\tau > 38\mu n^{3/2}$ and $\mu \leq n^{1/2-\epsilon}$. Then the algorithm requires at least $n^{\Omega(\sqrt{n})}$ steps with overwhelming probability.*

Proof. We use the same proof idea of Lemma 2 for the simple (μ+1) EA without a diversity mechanism. That is we track the individual with least 0-bits in the suffix (i. e., x) and show by drift analysis that it reaches the trap in expected time $19\mu n$. We consider again separate phases of expected duration of n steps. Each phase ends when a new leading 1-bit is created in some individual. Just like in Lemma 2 we bound the negative drift (i. e., decreasing the number of 0-bits in the suffix) in one step by $-3/8$. Here we pessimistically assume that whenever a 0-bit from the suffix is removed this is accepted as a new genotype. By Lemma 4 in time $8e\mu\log\mu$ all the individuals have the same number of leading 1-bits and no negative drift can occur for the remainder of the phase. On the other hand by Lemma 3 the positive drift in a phase is bounded by $1/(49\mu)$. Together with a negative drift of $-3/8$ occurring when the leading 1-bit is created ending the phase, the total drift (i. e., the expected increase of 0-bits in the suffix in x) in a phase is

$$E[\Delta] > \frac{n}{49\mu} - \frac{3}{8} \cdot (8e\mu \log \mu + 1) \geq \frac{n^{1/2+\epsilon}}{49} - 3en^{1/2-\epsilon} \log n^{1/2-\epsilon} - \frac{3}{8} > \frac{n^{1/2+\epsilon}}{50}$$

because $\mu \leq n^{1/2-\epsilon}$.

Hence, the expected number of phases is at most

$$\frac{(3/8)n}{n^{1/2+\epsilon}/50} = \frac{50 \cdot 3}{8} \cdot n^{1/2-\epsilon} < 19n^{1/2-\epsilon}$$

and the expected time for the whole population to reach the trap is bounded by $19 \cdot n \cdot n^{1/2-\epsilon}$.

Finally, concerning the time to reach the optimum by optimising the leading 1-bits in the suffix, we follow again ideas from Witt's proof [25]. He shows that the (μ+1)-EA without diversity requires at least $c\mu n \log n + n^2$ steps to optimise LeadingOnes with probability at least $1 - 2^{-\Omega(n)}$. However, the algorithm without diversity creates copies of the current best individual to speed up the time to create each new leading 1-bit. The (μ+1) EA with genotype diversity cannot use this "speed up" since copies of the same individual are not accepted. Hence the algorithm considered here can only be slower than the simple (μ+1) EA without a diversity mechanism. As a result, with overwhelming probability the algorithm requires at least m^2 iterations to optimise $m := n/2 - \sqrt{n}$ leading 1-bits in the prefix, while by using Markov's inequality iteratively we get that the runtime to reach the trap is greater than $2 \cdot 19 \cdot n^{2-\epsilon}$ steps with probability at most $2^{-\sqrt{n}}$. Given that from the trap at least $\sqrt{n}$ bits have to be flipped for an improvement, the theorem is proved. The statement about the expected optimisation time for $\tau > 20\mu n$ follows exactly the same calculations as in the proof of Theorem 1. □

5. Deterministic Crowding

We consider the so-called *deterministic crowding* mechanism [28]. Here, the main idea is that offspring only compete with their parents. Thus, the resulting algorithm (see Algorithm 3) is very similar to a parallel (1+1) EA, i. e., the individuals of the population explore the fitness landscape independently. Previous work showed that this mechanism is very efficient on the simple bimodal TwoMax problem [27] and some instances of the vertex cover problem [29]. However, on Balance deterministic crowding does not help since parallelism does not prevent the population from getting stuck in one of the traps.

Algorithm 3 (μ+1) EA with deterministic crowding

```
1: Let $t := 0$ and choose $x_1, \ldots, x_\mu \in \{0,1\}^n$ independently uniformly at
   random (u. a. r.); $P_t := \{x_1, \ldots, x_\mu\}$.
2: repeat
3:    Choose $x \in P_t$ u. a. r. and set offspring $y := x$.
4:    Flip each bit in $y$ independently with probability $1/n$.
5:    if $f(y) \geq f(x)$ then
6:       Let $P_{t+1} := P_t \setminus \{x\} \cup \{y\}$.
7:    else
8:       Let $P_{t+1} := P_t$.
9:    end if
10:   Let $t := t + 1$.
11: until some termination condition is met
```

Theorem 3. *W. o. p. the (μ+1) EA using deterministic crowding and $\mu = n^{O(1)}$ requires exponential time to optimise* Balance *if $\tau > 8e\mu n$, where $e = \exp(1)$.*

Proof. Similarly to [30], we consider a game of balls and bins such that each bin represents an individual of the population and each ball is a 0-bit flipping into a 1-bit in the suffix. Clearly, if $n/2 - n/16$ bits have been flipped in the suffix of an individual, then it has reached the trap. Hence we want to obtain the expected time for all the μ independent bins to have at least $n/2 - n/16$ balls. At each step the probability that each bin is selected and receives a ball is $p_i \geq (n/16)/(e\mu n) = 1/(e16\mu)$ because there are always at least $n/16$ 0-bits to choose from. Hence the expected number of balls after $t = 8e\mu n$ steps in a given bin B_1 is $E(|B_1|) \geq 8e\mu n \cdot 1/(e16\mu) = n/2$. By Chernoff bounds the probability that $P(|B_1| \leq (1-(1/8))n/2)$ is at most $e^{-\Omega(n)}$. By the union bound with probability at most $\mu e^{-\Omega(n)}$ there is at least one bin without $n/2 - n/16$ balls after t steps.

Now we derive a bound on the probability that the algorithm optimises $m = n/3$ bits of the prefix in less than $O(\mu m^2)$ steps. We consider a phase of $c\mu m^2$, $c > 0$ constant, steps. Since each individual is selected uniformly at random with probability $1/\mu$, the expected number of times a given individual is selected is cm^2. By Chernoff bounds the probability that in $c\mu m^2$ generations an individual is selected more than $(1+\delta)cm^2 = c'm^2$ times is exponentially small in n and by a simple union bound no individual is selected that many times w. o. p. Since to optimise the first m prefix leading 1-bit bits the (1+1) EA requires $c''m^2$ with $c'' > c'$ steps with probability at least $1 - e^{-\Omega(m)} = 1 - e^{-\Omega(n)}$ we get a runtime of at least $c''\mu m^2 \gg t$ steps w. o. p. Multiplying the failure probability of the prefix time and that of the suffix time concludes the proof. □

6. Fitness Diversity

In this section we consider the *fitness diversity* mechanism which does not allow fitness duplicates, i. e., multiple individuals in the population with the same fitness (see Algorithm 4). It resembles the idea proposed by Hutter and Legg [31].

Algorithm 4 (μ+1) EA with fitness diversity

```
1:  Let $t := 0$ and choose $x_1, \dots, x_\mu \in \{0,1\}^n$ independently uniformly at
    random (u. a. r.); $P_t := \{x_1, \dots, x_\mu\}$.
2:  repeat
3:    Choose $x \in P_t$ u. a. r. and set offspring $y := x$.
4:    Flip each bit in $y$ independently with probability $1/n$.
5:    if there exists $z \in P_t$ with $f(y) = f(z)$ then
6:      $P_{t+1} := P_t \setminus \{z\} \cup \{y\}$.
7:    else
8:      Choose $z \in P_t$ with minimal fitness u. a. r.
9:      if $f(y) \geq f(z)$ then
10:       Let $P_{t+1} := P_t \setminus \{z\} \cup \{y\}$.
11:     else
12:       Let $P_{t+1} := P_t$.
13:     end if
14:   end if
15:   Let $t := t + 1$.
16: until some termination condition is met
```

The mechanism has been previously analysed theoretically with contrasting results. Friedrich et al. [32] show that the runtime of the (μ+1) EA with fitness diversity is exponential for a simple plateau function if μ is bounded above by a constant, while the algorithm is efficient if the population size μ is set very close to n. Friedrich et al. [27] present far less encouraging results in terms of the actual diversity that the mechanism can achieve. In particular, it is shown that even for a simple bimodal function such as TwoMax the expected time to find both optima is exponential for any population size $\mu = n^{O(1)}$.

In the following we show that the (μ+1) EA with fitness diversity is efficient w. o. p. on Balance independent of the frequency of change for any population size greater than $\mu > n - 2(\sqrt{n} - 1)$. This is due to the fact that each of the two traps contains search points with exactly $n/2 - \sqrt{n} - 1$ different function values. Thus, the (μ+1) EA with fitness diversity and sufficiently large population size is able to "fill up" both traps and optimise Balance with the remaining individuals afterwards.

Theorem 4. *Let $\mu > n-2(\sqrt{n}-1)$. Then w. o. p. the (μ+1) EA with fitness diversity optimises* Balance *in time $O(\mu n^3)$ for arbitrary $\tau \geq 0$.*

Proof. The proof idea is that inside each of the traps there are only $n/2 - \sqrt{n}-1$ different fitness levels (i. e., precisely one fitness level for all bit strings with exactly i leading 1-bits and $0 < i < n/2 - \sqrt{n}$).

By Chernoff bounds the probability that an individual is initialised in the trap is exponentially small. The probability that $n - 2(\sqrt{n} - 1)$ individuals are initialised in the trap is much smaller.

Let τ be initially odd (the proof for τ initially even is analogous). We pessimistically assume that $n/2 - \sqrt{n} - 1$ individuals end up in the upper trap before reaching the optimum. No more individuals will be allowed in the trap. We also pessimistically assume that the optimum is not found before the fitness function changes and that other $n/2 - \sqrt{n} - 1$ individuals "fill up" the fitness levels of the lower trap. Hence the remaining $\mu - (n - 2(\sqrt{n}-1)) \geq 1$ individual(s) will be "forced" to optimise the leading 1-bits in the prefix without accessing the trap because the fitness diversity mechanism does not accept further trap points. The expected time to reach the optimum is trivially bounded above by $e\mu n^2/2$ generations since the probability of increasing the current number of leading 1-bits by one is $1/(e\mu n)$ requiring $e\mu n$ expected generations and at most $n/2$ increases are necessary for the optimum to be reached.

Due to Markov's inequality the probability not to reach the optimum in $e\mu n^2$ generations is at most $1/2$. Moreover, after this number of generations we are not in a worse situation than before. Thus, considering $n/2$ phases of length $e\mu n^2$, the probability not to reach the optimum in all phases, i. e., in $e\mu n^3/2$ iterations, is bounded above by $2^{-\Omega(n)}$. Summing up the failure probabilities the theorem follows. □

7. Fitness Sharing

The so-called *fitness sharing* mechanism [28] attempts to achieve a diverse population by forcing similar individuals to "share" their fitness. The idea behind the mechanism is that in order to increase diversity, hence keep individuals far away from each other, similar individuals should be penalised by a decrease of their real fitness. More precisely fitness sharing removes an amount from the real fitness of each individual according to its similarity with the rest of the population. The similarity between two individuals x and y is measured by a *sharing function* $sh(x, y) \in [0, 1]$. Given some distance function d, the standard sharing function is defined as [28]

$$sh(x, y) = \max\left\{0, 1 - \left(\frac{d(x, y)}{\sigma}\right)^{\alpha}\right\},$$

where σ is called the *sharing distance*, such that only individuals of distance at most σ share their fitness, and α is a constant that regulates the shape of the sharing function. The standard setting, as suggested by Mahfoud [28], is $\alpha = 1$ while the parameter σ should be set according to the number of optima and their separation.

The *shared fitness* of an individual x with the rest of the population is defined by

$$f(x, P) = \frac{f(x)}{\sum_{y \in P} sh(x, y)}$$

and the fitness of the population is $f(P) = \sum_{x \in P} f(x, P)$.

The algorithm has been proved to be very effective for the TwoMax bimodal function by Friedrich et al. [27]. However, for the algorithm to work it was necessary to use the knowledge that TwoMax is a function of unitation and to use the number of 1-bits in individuals as distance function d.

We will show how fitness sharing can be very effective for Balance already with population size $\mu = 2$ and by using the natural Hamming distance as distance function, i. e., $d(x, y) = H(x, y)$. We use the standard setting $\alpha = 1$ and set the sharing distance to $\sigma = n$, implying that all individuals share their fitness (i. e., we assume no information about the peak distribution). In order to simplify the analysis we use 1-bit mutation instead of the standard bit mutation used in the previously analysed algorithms and add a crowding mechanism to simplify the selection step. Since 1-bit mutation flips exactly one bit (chosen uniformly at random) per mutation, the algorithm is a variant of random local search (RLS). The resulting algorithm is depicted in Algorithm 5.

Algorithm 5 (μ+1) RLS with fitness sharing and deterministic crowding

```
1: Let $t := 0$ and choose $x_1, \dots, x_\mu \in \{0,1\}^n$ independently uniformly at
   random (u. a. r.); $P_t := \{x_1, \dots, x_\mu\}$.
2: repeat
3:    Choose $x \in P_t$ u. a. r. and set offspring $y := x$.
4:    Choose $i \in \{1, \dots, n\}$ u. a. r. and flip bit $y[i]$.
5:    Let $P'_t = P_t \setminus \{x\} \cup \{y\}$.
6:    if $f(P'_t) \geq f(P_t)$ then
7:       Let $P_{t+1} := P'_t$.
8:    else
9:       Let $P_{t+1} := P_t$.
10:   end if
11:   Let $t := t + 1$.
12: until some termination condition is met
```

In the following, we consider the case $\mu = 2$, i. e., a (2+1) RLS. We will show that fitness sharing can prevent the population from entering the traps by guiding the population to individuals with complementary suffixes satisfying $n/16 < |b|_1 < 7n/16$. Once achieved, this property is never lost and thus, the algorithm is able to optimise the LeadingOnes part in the prefix.

In the analysis we first examine the individuals after initialisation and show that these points have w. o. p. a linear number of *non-overlapping* 1- and 0-bits in the suffix, i. e., positions where one individual has a 0-bit while the other one has a 1-bit. Afterwards, we investigate which kind of mutation steps are accepted by the algorithm. We consider different cases depending on the increase/decrease of fitness as well as Hamming distance (the sharing distance). We do this separately for mutations affecting the prefix and the suffix (recall, that Algorithm 5 only uses 1-bit mutations). In particular, we prove under which conditions a decrease in fitness is accepted due to the fitness sharing mechanism. Plugging these results together yields our main theorem.

Lemma 5. *Let $\mu = 2$, $P_0 = \{x, y\}$ the initial population and $0 < \epsilon < 1/8$ some constant. W. o. p. x and y have at least $n/8 - \epsilon n$ non-overlapping 1-bits (i. e., 1-bits in x at positions where there is a 0-bit in y) and at least $n/8 - \epsilon n$ non-overlapping 0-bits in the suffix.*

Proof. In expectation an individual x is initialised with $n/4$ 1-bits and $n/4$ 0-bits in the suffix. The same holds for individual y. Half of the 1-bits of x in the suffix (i. e., $n/8$) are expected to overlap (i. e., are in the same position) with the 1-bits in y while the other $n/8$ 1-bits *do not* overlap with 1-bits in y. The same amount of overlapping and *non-overlapping* 0-bits are expected in the suffix. By Chernoff bounds w. o. p. x and y have at least $n/8 - \epsilon n$ non-overlapping 1-bits and at least $n/8 - \epsilon n$ non-overlapping 0-bits in the suffix. □

For the sake of readability, we omit the index t in the following two lemmas. Moreover, we define $f(x, y) := f(x) + f(y)$. Recall that P' is the population including the offspring while P denotes the population including the parent.

Lemma 6. *Let $\mu = 2$, $P = \{x, y\}$ and $d = H(x, y)$.*

For $f(x, y) + d > 2n$, bit-flips in the suffix are accepted if and only if they increase the Hamming distance.

For $f(x, y) + d < 2n$, bit-flips in the suffix are accepted if and only if they increase the fitness.

Proof. We first observe that the cases where both the fitness and the Hamming distance increase or decrease are trivial: we accept the offspring in case both are increased and keep the parent otherwise. In the case where the fitness increases by 1 while the Hamming distance decreases by 1 we have

$$f(P) = \frac{f(x, y)}{2 - \frac{d}{n}}, \quad f(P') = \frac{f(x, y) + 1}{2 - \frac{d}{n} + 1/n}$$

A straightforward calculation yields that $f(P') < f(P)$ if and only if $f(x, y) > 2n - d$. In the very same way, we can consider the case that fitness decreases while Hamming distance increases and get that

$$f(P') = \frac{f(x, y) - 1}{2 - \frac{d}{n} - \frac{1}{n}} > \frac{f(x, y)}{2 - \frac{d}{n}} = f(P)$$

holds for $f(x,y) > 2n - d$.

For the second claim, i. e., $f(x,y) < 2n - d$, we can repeat the very same calculations with inverted inequality sign. □

With very similar calculations the following lemma about mutations in the prefix can be shown.

Lemma 7. *Let $\mu = 2$, $P = \{x,y\}$, $d = H(x,y)$ and $c \in \mathbb{N}$ some constant that denotes the increase in leading 1-bits after mutation.*

For $f(x,y) + dcn > 2cn^2$, bit-flips in the prefix increasing/decreasing the fitness by at least n are accepted if and only if they increase the Hamming distance.

For $f(x,y) + dcn < 2cn^2$, bit-flips in the prefix increasing/decreasing the fitness by at least n are accepted if and only if they increase the fitness independent of Hamming distance.

Proof. As in the previous lemma we see that the cases where both the fitness and the Hamming distance increase or decrease are trivial: we accept the offspring in case both are increased and keep the parent otherwise. We consider the remaining cases.

In the case where the fitness increases by cn, i. e., we add c leading 1-bits, while the Hamming distance decreases by 1 we have

$$f(P) = \frac{f(x,y)}{2 - \frac{d}{n}}, \quad f(P') = \frac{f(x,y) + cn}{2 - \frac{d}{n} + 1/n}$$

Again a straightforward calculation shows that $f(P') < f(P)$ if and only if $f(x,y) > 2cn^2 - dcn$.

In the very same way, we can show that

$$f(P') = \frac{f(x,y) - cn}{2 - \frac{d}{n} - \frac{1}{n}} > \frac{f(x,y)}{2 - \frac{d}{n}} = f(P)$$

holds for $f(x,y) > 2cn^2 - dcn$.

For the second claim, i. e., $f(x,y) < 2cn^2 - dcn$, we can repeat the very same calculations with inverted inequality sign. □

We are now ready to show that the considered (2+1) RLS optimises Balance in a polynomial number of steps with at least constant probability.

Theorem 5. *With probability at least $p = 1/2 - e^{-\Omega(n)}$ the (2+1) RLS with fitness sharing and crowding finds the optimum of* Balance *in $O(n^2)$ steps for arbitrary $\tau \geq 0$.*

Proof. First we will show that with probability bounded below by a constant (i. e., exponentially close to 1/2) the two individuals will never reach the trap (Part 1). Afterwards we prove that, if the trap is not reached, the optimum will be found in time $O(n^2)$ by optimising the LeadingOnes part in the prefix (Part 2).

With probability 1/2, we have $\text{LO}(x)+\text{LO}(y) \geq 2$ at initialisation (Event E_1) since

$$\begin{aligned} &\text{Prob}\left(\text{LO}(x) = 2 \vee \text{LO}(y) = 2\right) \\ &= \text{Prob}\left(\text{LO}(x) = 2\right) + \text{Prob}\left(\text{LO}(y) = 2\right) - \text{Prob}\left(\text{LO}(x) = 2 \wedge \text{LO}(y) = 2\right) \\ &= \frac{1}{4} + \frac{1}{4} - \frac{1}{16} = \frac{7}{16} \end{aligned}$$

and $\text{Prob}\left(\text{LO}(x) = 1 \wedge \text{LO}(y) = 1\right) = \frac{1}{16}$.

If the sum of leading 1-bits in x and y is greater or equal than 2, then by Lemma 6 only bit-flips in the suffix increasing the Hamming distance are accepted since it is trivial to see that $f(x, y) + d > 2n$.

By Lemma 5 w. o. p. there are at least $n/8 - \epsilon n$ *non-overlapping* 0-bits in the suffix of x and at least $n/8 - \epsilon n$ *non-overlapping* 0-bits in the suffix of y (Event E_2). These bits, if flipped, will lead to individuals with lower Hamming distance, thus by Lemma 6 will not be accepted. As a result, conditional to Events E_1 and E_2, both x and y have $n/8 - \epsilon n$ 0-bits that will never be removed with probability $p = 1$. This implies that they can achieve at most $n/2 - n/8 + \epsilon n = (3/8 + \epsilon)n = (6/16 + \epsilon)n < (7n)/16$ 1-bits, thus never reach the upper trap.

With the same reasoning we can show that the lower trap will never be reached using from Lemma 5 that w. o. p. there are at least $n/8 - \epsilon n$ *non-overlapping* 1-bits in the suffices of x and y that will lead to non-accepted individuals if flipped to 0-bits. Furthermore, since only the suffix of the genotype is affected by the dynamics of the fitness function, the two leading 1-bits will never be removed and $f(x, y) > 2n$ will hold independent of the frequency of change τ. Hence Part 1 is proved.

It remains to be shown that the optimum will be found by optimising the LeadingOnes part in the prefix (Part 2). By Lemma 7 increases in leading 1-bits are always accepted as long as $f(x, y) + dcn < 2cn^2$. Let $f(x, y)_a$ and

$H(x,y)_a$ be respectively the contributions to fitness and to distance due to the prefix and $f(x,y)_b$, $H(x,y)_b$ the contributions due to the suffix. We aim to show that

$$f(x,y)+dcn < 2cn^2 \iff f(x,y)_a+f(x,y)_b+[H(x,y)_a+H(x,y)_b]\cdot cn < 2cn^2.$$

If the above inequality holds, then the LeadingOnes part will be optimised, hence the optimum will be found.

We start with the suffix. Recall, that only moves increasing the Hamming distance are accepted here. Hence, the Hamming distance in the suffix will be maximised until a point where $H(x,y)_b = f(x,y)_b = n/2$ is reached. We observe that the underlying process corresponds to the well-known coupon collector problem with n different coupons (i. e., non-overlapping 0- or 1-bits) where each coupon is obtained with probability $1/n$. Note, that in fact, we only need to collect $n/2$ specific coupons and that initially, we already have at least $n/8-\epsilon n$ coupons w. o. p. The expected number of steps until $H(x,y)_b = f(x,y)_b = n/2$ is at most $n\log n + O(n)$ [14] since the considered process is easier than the original coupon collector. Moreover, the probability that more than $\beta n \ln n$, $\beta > 1$, steps are needed is bounded above by $n^{-(\beta-1)}$ [14].

Concerning the prefix, its fitness contribution is at most

$$f(x,y)_a = n\cdot(LO(x)+LO(y)) \le n\cdot(n/2+n/2) \le n^2$$

because the prefix has length $n/2$. Bit-flips in the prefix after the leftmost 0-bit are always accepted if and only if they increase the Hamming distance. This trivially follows because these bits do not contribute to fitness. Hence, again by using results for the coupon collector problem with probability $1-n^{-(\beta-1)}$ after $\beta n\ln n$, $\beta > 1$, steps the Hamming distance of the bits after the leftmost 0-bit is maximised. This implies $H(x,y)_a = n/2 - \min(\mathrm{LO}(x),\mathrm{LO}(y))$. Putting everything together we get after $O(\beta n \ln n)$ steps,

$$\begin{aligned} & f(x,y)+dcn \\ =\ & f(x,y)_a+f(x,y)_b+[H(x,y)_a+H(x,y)_b]\cdot cn \\ \le\ & n^2+n/2+[n/2+n/2-\min(\mathrm{LO}(x),\mathrm{LO}(y))]\cdot cn \\ =\ & (c+1)n^2+n/2-\min(\mathrm{LO}(x),\mathrm{LO}(y))\cdot cn < 2cn^2 \end{aligned}$$

where the last inequality holds as long as $\min(\mathrm{LO}(x),\mathrm{LO}(y)) > 1/2$.

Conditional to Event E_1 either x or y have at least one leading 1-bit. If one individual has a leading 0-bit it suffices to flip it for the inequality to hold. This requires $2n$ expected steps. The probability not to flip this bit in $2n^2$ steps is at most $(1-1/n)^{2n^2} \leq e^{-2n}$.

Finally, with similar arguments to Theorem 17 in [33] it follows that the $n/2$ leading 1-bits will be created in cn^2 ($c > 0$ constant) steps with probability at least $1 - e^{-\Omega(n)}$. Summing up the runtimes, multiplying the failure probabilities and setting $\beta = \Theta(n/\log n)$ concludes the proof since the dynamics of the fitness function do not affect the prefix (i. e., the proof holds independent of τ). □

We have shown that the (2+1) RLS is able to optimise Balance in polynomial time with at least some probability converging to 1/2 exponentially fast. However, if the frequency of change is small we can also show that there is a small probability to reach the traps with both individuals.

Theorem 6. *Let $\tau > 12n + 1$. With probability bounded below by a constant the (2+1) RLS with fitness sharing and crowding requires infinite time to optimise* Balance.

Proof. W. l. o. g., we assume that the time step τ is such that fitness in the suffix increases with 1-bits. The proof strategy considers four consecutive phases. By the end of the last phase both individuals will be in opposite traps. The statement of the theorem will follow by multiplying the success probabilities of each phase. The first phase starts at initialisation and ends when individual x reaches a point with $n/16+1$ 0-bits in the suffix conditional to x and y starting without any leading 1-bits and never creating any (i. e., $LO(x) + LO(y) = 0$) (Event E_1). Since $LO(x) + LO(y) = 0$ during the whole phase, by Lemma 6 bit-flips in the suffix are accepted if and only if they increase fitness (i. e., $f(x, y) + d < 2n$ holds trivially since $f(x, y) < n$ until a leading 1-bit is created and $d \leq n$ always holds).

The second phase ends when x enters the upper trap. Since without leading 1-bits the trap points have zero fitness, for x to enter the trap, firstly, a leading 1-bit has to be created and then the trap will be entered in the following step by flipping a 0-bit into a 1-bit. Such a leading 1-bit will be accepted because it increases both fitness and Hamming distance. At the end of the phase $LO(x) = 1$ and $LO(y) = 0$.

The third phase ends when individual y reaches a point with $n/16 + 1$ 1-bits conditional to no other leading 1-bits being created in the mean time

(i. e., $LO(x) = 1$ and $LO(y) = 0$ throughout the phase). Since $LO(x) = 1$, by Lemma 6 bit-flips in the suffix are accepted if and only if they increase the Hamming distance (i. e., $f(x, y) + d > 2n$ which follows trivially since $f(x) = n^2$ due to the first leading 1-bit). Hence, in this phase we calculate the time for y to reach a "limit-point" before entering the lower trap. A crucial observation for this phase is that the current trap point x will never have more than $n/16$ 0-bits in the suffix, because that would mean abandoning the trap. This would imply a decrease in fitness of more than n, an event that by Lemma 7 will not be accepted as long as $f(x, y) + dn < 2n^2$ (here c is conveniently set to 1). Given that throughout the phase $d \leq n-1$, $f(x) = n^2$ and $f(y) < n/2$ we get

$$f(x, y) + dn \leq n^2 + n/2 + (n - 1)n = 2n^2 - n/2 < 2n^2 \tag{1}$$

implying that as long as the second leading 1-bit is not created the x individual cannot leave the trap (i. e., by increasing the Hamming distance y is directed towards the opposite trap).

The fourth phase ends when y enters the lower trap. Just like for x, it is necessary for y to first create a leading 1-bit otherwise the trap points would have zero fitness. Since x and y still have in total only one leading 1-bit $f(x, y) + dcn < 2n^2$ (still by Equation (1)) and a leading 1-bit increasing fitness by n will be accepted independent of Hamming distance (i. e., Lemma 7).

Once both individuals are in each of the traps and have one leading 1-bit each, by both Lemmas 6 and 7 only bit-flips increasing Hamming distance are accepted (i. e., $f(x, y) + d > 2n$ and $f(x, y) + dcn > 2cn^2$). As a consequence, the $(7/16)n$ non-overlapping bits in each individual determining the trap points will not be accepted if flipped, i. e., the individuals will never leave their respective traps unless they reach the optimum, which is impossible with 1-bit mutation. Hence the proof is concluded by calculating the probabilities of each phase happening. These will be calculated in the remainder of the proof.

For the first phase we will apply the additive drift theorem by He and Yao [34]. Since $f(x, y) < n$ by Lemma 6 bit-flips in the suffix are accepted if and only if they increase fitness. The probability that the current best individual is selected is $1/2$. The probability that a 0-bit is flipped into a 1-bit is always at least $(n/16)/n = 1/16$ because there are always at least $n/16$ 0-bits to choose from in a mutation step (unless the trap is reached).

Overall, the drift is $E(\Delta) > (1/2) \cdot (1/16) = 1/32$. At initialisation each individual has with overwhelming probability at most $n/4 + \epsilon n$ 0-bits in the suffix. Hence the drift theorem yields

$$E(T_{trap}) \leq \frac{d^0}{E(\Delta)} = \frac{n/4 + \epsilon n - n/16}{1/32} = 32 \cdot (3/16 + \epsilon)n = (6 + \epsilon')n < 7n$$

steps to reach the point with $n/16 + 1$ 0-bits. By Markov's inequality the probability that T_{trap} is greater than $8n$ is less than $7/8$.

Now we calculate the probability of Event E_1 that no leading 1-bit is created before the current trap-limit point is reached. The probability that neither x or y are initialised with a leading 1-bit is $1/4$. At each step with probability $1 - 1/n$ the leading 1-bit is not created. Hence with probability

$$\left(1 - \frac{1}{n}\right)^{8n} \geq \left(\frac{1}{2e}\right)^{8}$$

it is not created for $8n$ steps. Altogether Phase 1 is concluded in $8n$ steps with probability at least $(1/8) \cdot (1/4) \cdot (1/2e)^8$.

For Phase 2 trap points will not be accepted until the first leading 1-bit is created. The expected time for the first leading 1-bit in x is $2n$ (i. e., with probability $1/2$ x is selected and with probability $1/n$ the first bit is flipped). By a simple application of Markov's inequality, it will not require more than $4n$ steps with probability at least $1/2$. Furthermore, with probability $(1/2) \cdot (n/16 + 1)/n > 1/32$ in the step straight after the leading 1-bit is created, x reaches the trap by flipping the last necessary suffix 0-bit into a 1-bit. Since we also require that $LO(x) = 1$ at the end of the phase, we need to also multiply by the probability p_f that the the second bit is not a free rider (i. e., $p_f = 1/2$ since the bit is not subject to fitness). Hence, Phase 2 is concluded after $4n + 1$ steps with probability at least $1/128$. Phase 1 and 2 imply that as long as $\tau > 12n + 1$ the first individual will end up in the trap with probability bounded below by a constant.

Now, for Phase 3, we calculate the time for the other individual y to reach the other trap. Recall that, due to the first leading 1-bit created in the previous phase, by Lemma 6 only bit-flips in the suffix that increase the Hamming distance are accepted. Hence the rest of the proof holds for any τ. Also recall that by Lemma 7 x cannot leave the upper trap as long as the second leading 1-bit is not created, hence none of its at least $(7/16)n$ 1-bits in the suffix may be flipped into 0-bits. This implies that there are at least

$(7/16)n$ positions in the suffix of y that either already are 0-bits or will be accepted if flipped into 0-bits. Hence, we apply a similar proof strategy to that used for the first individual. By Applying drift analysis again we get

$$E(T_{trap}) \leq \frac{d^0}{E(\Delta)} \leq \frac{n/2 - n/16}{1/32} = 32 \cdot (7/16)n = 14n$$

as bound on the expected time to reach the lower limit-trap point with $n/16+1$ 1-bits. By Markov's inequality the probability that T_{trap} is greater than $15n$ is at most $14/15$.

With probability

$$\left(1 - \frac{1}{n}\right)^{15n} \geq \left(\frac{1}{2e}\right)^{15}$$

the second leading 1-bit is not created for $15n$ steps. Hence Phase 3 is concluded in $14n$ steps with probability at least $1/15 \cdot (1/2e)^{15}$.

For Phase 4 we calculate the probability of the two consecutive steps, one creating the first leading 1-bit in y and the other adding the final suffix 0-bit to enter the trap. Recall that by Lemma 7 a leading 1-bit increasing fitness by n will be accepted independent of Hamming distance. Following the same calculations as for Phase 2 we get a probability of $1/2$ that y creates its first leading 1-bit in $4n$ steps and a probability of at least $1/32$ that the trap is entered in the next step. Hence Phase 4 is concluded in $4n+1$ steps with probability at least $1/64$.

Since both individuals have at least one leading 1-bit, by both Lemmas 6 and 7 only bit-flips increasing the Hamming distance are accepted. Furthermore the Hamming distance in the suffix is maximised and hence, the two individuals will never leave their respective traps. Finally, given that the RLS algorithm is not able to pass the region of zero fitness in one step, multiplying the probabilities of each phase concludes the proof. □

Through the previous analysis it is clear that fitness sharing is considerably sensitive to the difference between fitness values of the individuals. For this reason we can only prove a success probability close to $1/2$ for the (2+1) RLS algorithm to efficiently optimise Balance. One way to increase the success probability would be to modify the shape of the sharing function by tuning parameter α. However, lots of problem knowledge would have to be included to tune α correctly. Another way around the problem is to consider mechanisms available in the literature to deal with algorithm sensitivity towards difference in fitness values. A classical example is the fitness scaling used to improve the performance of fitness-proportional selection EAs when the difference in fitness values between close individuals is not detected by selection leading to low selection pressure [35]. Neumann et al. [36] have rigorously proved how scaling mechanisms can turn the runtime of fitness proportional selection EAs from exponential to polynomial even for simple functions such as OneMax. In the following we show how scaling the fitness function will make the (2+1) RLS with fitness sharing and crowding very effective for Balance.

Theorem 7. *With probability $1 - e^{-\Omega(n)}$ the (2+1) RLS with fitness sharing and crowding finds the optimum of $f(x) = \text{Balance}\ (x) + n$ in $O(n^2)$ steps for arbitrary $\tau \geq 0$.*

Proof. The proof follows the same line of thought of Theorem 5. By Lemma 5 with overwhelming probability there are $n/8 - \epsilon n$ non-overlapping 0-bits in x and the same number in y at initialisation. Also, since through the fitness scaling both x and y have a fitness value of at least n, it follows that $f(x, y) > 2n$. Hence, by Lemma 6 only bit-flips with increasing Hamming distance are accepted and the non-overlapping bits will never be removed. As a result the trap will never be reached and after $\beta n \log n$ steps and $\beta > 1$ the Hamming distance between x and y is maximised with probability $1-n^{-(\beta-1)}$. Hence we get, $f(x, y)_b = n/2 + n = (3/2)n, h(x, y)_b = n/2, f(x, y)_a = n \cdot (LO(x) + LO(y)) + n$ and $h(x, y)_a = n/2 - \min(\text{LO}(x), \text{LO}(y))$. This implies that:

$$\begin{aligned}
& f(x, y) + dcn \\
&= f(x, y)_a + f(x, y)_b + [h(x, y)_a + h(x, y)_b] \cdot cn \\
&\leq n^2 + n + (3/2)n + [n/2 + n/2 - \min(\text{LO}(x), \text{LO}(y))] \cdot cn \\
&= (c + 1)n^2 + (5/2)n - \min(\text{LO}(x), \text{LO}(y)) \cdot cn < 2cn^2
\end{aligned}$$

which always holds for $c > 1$ and large enough n and holds for $c = 1$ as long as $\min(\text{LO}(x), \text{LO}(y)) > 5/2$. This implies that both individuals need three leading 1-bits for the inequality to hold when $c = 1$.

Let one or both individual(s) not have the 3 leading 1-bits at this stage. Then $f(x, y)_a \leq n^2/2 + 3n$. Plugging this into the above calculations implies that $f(x, y) + dcn < 2cn^2$, so the first three leading 1-bits will also be accepted.

Finally, just like in Theorem 5, we show that the $n/2$ leading 1-bits will be created in cn^2 steps with probability at least $1-e^{-\Omega(n)}$ since the dynamics of the function do not affect the prefix. By multiplying the failure probabilities the theorem statement follows. □

8. Experimental Supplements

We perform experiments to shed light on some additional aspects of the considered algorithms. These include, in particular, situations where our theoretical results do not cover the whole possible parameter range. We give extra insights into the working principles of the different diversity mechanisms considered and point out, based on these findings, interesting directions for future research.

Since our main interest is in the ability of diversity mechanisms with respect to small frequencies of change, we perform all experiments on the static version of Balance, i. e., we set $\tau = \infty$. We have implemented all algorithms analysed in the previous sections ((μ+1) EA without diversity, with genotype diversity, with fitness diversity, with deterministic crowding, and with fitness sharing and crowding) as well as a (μ+1) EA with pure fitness sharing. In all the experiments we stop the algorithms if one individual of the population has reached a global optimum or if the whole population has reached either of the traps.

We start our experimental analysis considering the success rates of the different algorithms, i. e., the number of times we reach a global optimum over a certain number of independent runs. In a second step, we concentrate on fitness sharing with and without crowding and investigate different additional properties such as the Hamming distance of individuals over a run, the influence of larger population sizes as well as the number of function evaluations needed to reach an optimal solution. We present our findings with respect to the different algorithms and parameterisations in the following.

8.1. No Diversity

Considering the (μ+1) EA without diversity mechanisms (Algorithm 1), we have proven that the algorithm with population of size $\mu \leq n^{1/2-\epsilon}$ requires exponential runtime with overwhelming probability (Theorem 1). However, it is an open question how large the population sizes have to be for the algorithm to be efficient. We conjecture that large μ values increase the drift away from the traps and thus help the algorithm not to get trapped.

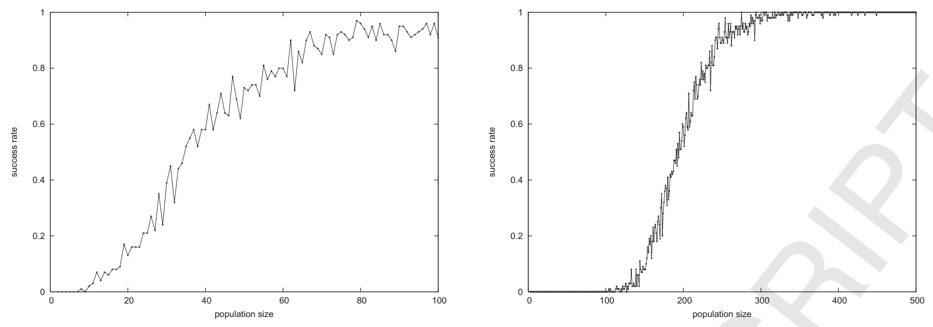


Figure 2: Success rate over 100 runs for no diversity mechanism, $n = 100$ and $n = 500$.

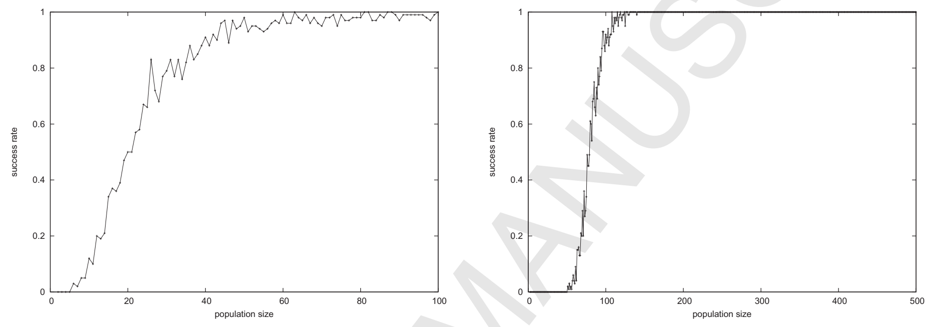


Figure 3: Success rate over 100 runs for genotype diversity, $n = 100$ and $n = 500$.

We perform experiments for $n = 100$ and $n = 500$ and population sizes of $1 \leq \mu \leq n$ and visualise the success rates in 100 independent runs in Figure 2. We observe that for linear μ the success rate approaches 1, supporting our conjecture. In fact there appears to be a phase transition for populations of sizes $\mu = cn$ for some value of c.

8.2. Genotype Diversity

For the (μ+1) EA with genotype diversity (Algorithm 2) we also have negative results that hold with overwhelming probability for $\mu \leq n^{1/2-\epsilon}$ (Theorem 2) and again conjecture that larger population sizes do help. We perform the same experiments as for the (μ+1) EA without diversity. We depict the results in Figure 3 and see that the observed phase transition is even more apparent and appears to happen for lower population sizes compared to the algorithm without diversity.

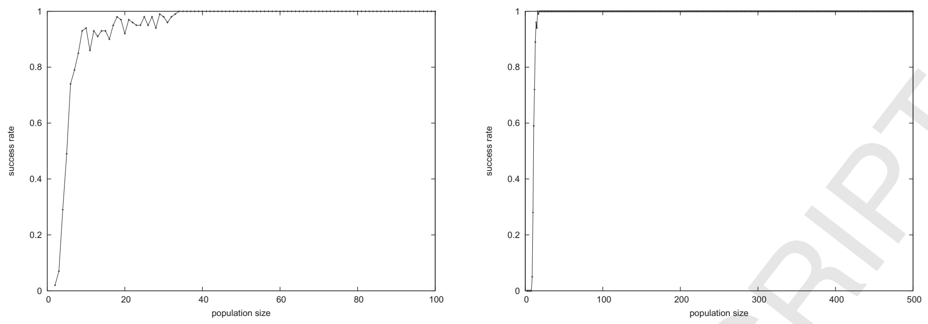


Figure 4: Success rate over 100 runs for fitness diversity, $n = 100$ and $n = 500$.

8.3. Deterministic Crowding

We have proved that the (μ+1) EA with deterministic crowding (Algorithm 3) is not able to optimise BALANCE efficiently with overwhelming probability for $\mu = n^{O(1)}$ (Theorem 3). We have performed experiments for $n = 100$, $n = 1000$ and $n = 10000$ with $\mu = 2$, $\mu = \lfloor\sqrt{n}\rfloor$ and $\mu = n$ (100 independent runs each) and have observed only a single successful run for $n = 100$ and $\mu = n$. So even for small input sizes and reasonable population sizes deterministic crowding does not help.

8.4. Fitness Diversity

For the (μ+1) EA with fitness diversity (Algorithm 4) we have proved that using a population size $\mu > n - 2(\sqrt{n} - 1)$ enables the algorithm to optimise BALANCE in polynomial time (Theorem 4). Recall that this result is based on the fact that there is only a limited number of different fitness values within the traps and that the algorithm cannot get trapped once all these places are filled up. However, we conjecture that the trap does not need to be filled up entirely for the algorithm to optimise BALANCE efficiently. We therefore perform the same experiments as we did for the (μ+1) EA without diversity and with genotype diversity. We visualise the results in Figure 4 and see that much smaller population sizes are sufficient. It is an interesting question to further investigate this observed sharp phase transition.

8.5. Fitness Sharing

We finally consider the fitness sharing mechanism. Recall that, in order to simplify the proofs, we have modified the considered algorithm by only using 1-bit mutation and by adding a crowding mechanism (Algorithm 5).

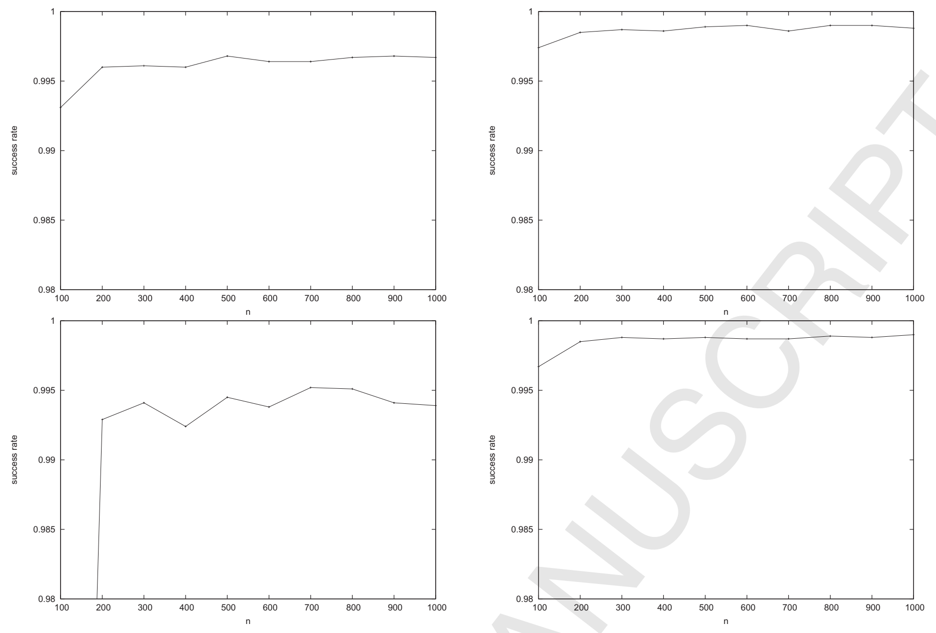


Figure 5: Success rate over 10000 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $\mu = 2$.

Moreover, we have only proved results for $\mu = 2$. We have shown that with probability at least $p = 1/2 - e^{-\Omega(n)}$ this algorithm is able to optimise Balance in polynomial time (Theorem 5). However, we have also proved that the algorithm fails with a probability bounded by a very small constant (Theorem 6).

We perform experiments for four different variants of the algorithm, with and without the crowding mechanism as well as with 1-bit mutation and standard bit mutation. We start with similar experiments as before regarding the success rate, however, due to the rather small failure probability we perform 10.000 runs. We first consider $\mu = 2$ according to our theorems and investigate the success probabilities for $n \in \{100, 200, \ldots, 1000\}$. The results are depicted in Figure 5. We see that the success probability is close to 1 for all four algorithms. While the crowding mechanism does not have a huge influence, standard bit mutations increase the success probability even further.

Since we use Hamming distance in our sharing mechanism, it is interesting to see how the Hamming distance of the two individuals of the population

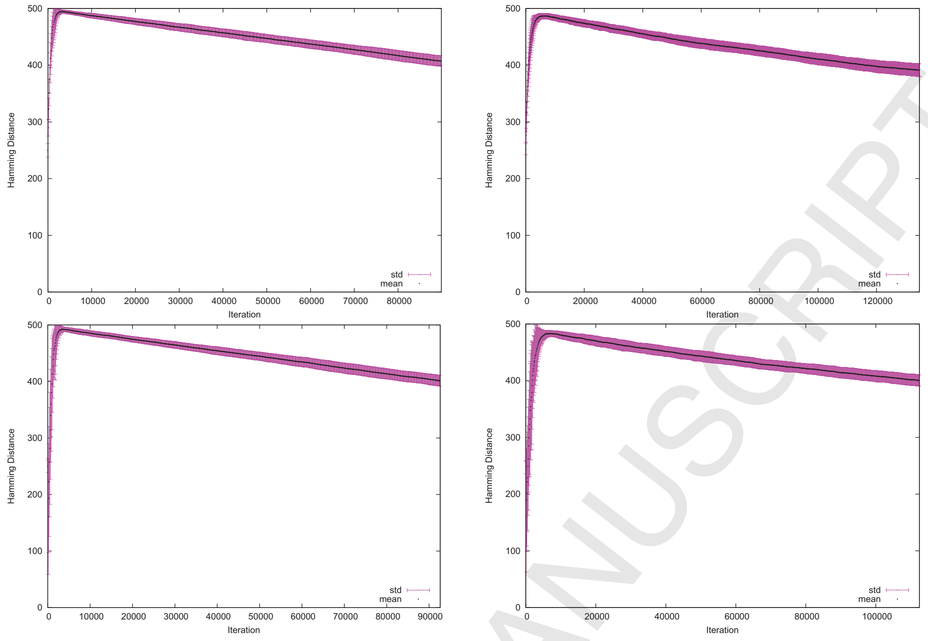


Figure 6: Hamming distance over 100 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $\mu = 2$, $n = 500$.

evolve over the run of the algorithm. We do this for $n = 500$ and plot both, the Hamming distance of the entire individual (Figure 6) and of its suffix (Figure 7). In the latter figure, we see that the Hamming distance in the suffix quickly converges to its maximum, i. e., $n/2$, and is not changed afterwards when using 1-bit mutation. For standard bit mutation it converges to $n/2$, too, and then fluctuates close to $n/2$. For the entire individual we see an increase in Hamming distance in the beginning while it decreases towards the end of the algorithm. Since the Hamming distance in the suffix is unchanged (or close to unchanged) towards the end of the process, this can only be explained by the events in the prefix, i. e., while optimising the LeadingOnes part in the prefix of both individuals their prefixes become more similar, thus decreasing the overall Hamming distance. These figures resemble exactly the proof ideas used in our proofs (see in particular Theorem 5).

Given the observation that the Hamming distance in the suffix is maximal for most of the optimisation process, we are now interested in the concrete number of 1-bits in the suffix since this determines the position of the indi-

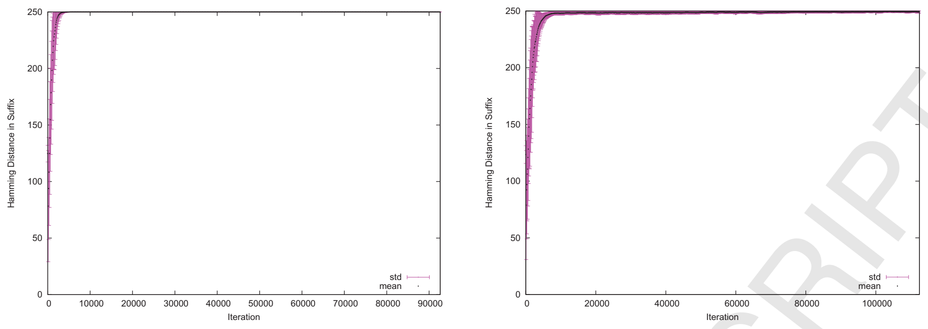


Figure 7: Hamming distance within the suffix over 100 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $\mu = 2$, $n = 500$.

viduals with respect to the traps. We depict this in Figure 8 for $n = 500$ and see that both individuals have roughly $n/4$ 1-bits in the suffix over most of the run (i. e., the individuals proceed along the centre of the corridor towards the optimum).

For $\mu = 2$, we are also interested in the question if adding the crowding mechanism or not as well as using 1-bit mutation or standard bit mutation have a significant influence on the runtime in case of a successful run. We therefore investigate the number of function evaluations given that we reach a global optimum and depict the result as box-and-whisker plots (showing the minimum, maximum, median and lower and upper quartile over 100 runs) in Figure 9. We see that the runtime behaviour with respect to adding crowding or not is very similar while standard bit mutations slow down the optimisation process.

Finally, we want to investigate the influence of large population sizes. We perform experiments for $n = 100$ and $\mu \in \{2, 3, \ldots, 100\}$ and depict the results in Figure 10. We observe a clear difference between the algorithm with crowding and the one using pure fitness sharing as the population size increases. While for the latter algorithm the success rate is still very close to 1 even for large population sizes, the success rate for fitness sharing and crowding decreases drastically with increasing population size. We conclude that only comparing the offspring to the parent is detrimental when using fitness sharing with larger populations while the size of the population has no significant influence on the success rate when using pure fitness sharing. We remark that with some small probability the (μ+1) EA using fitness sharing

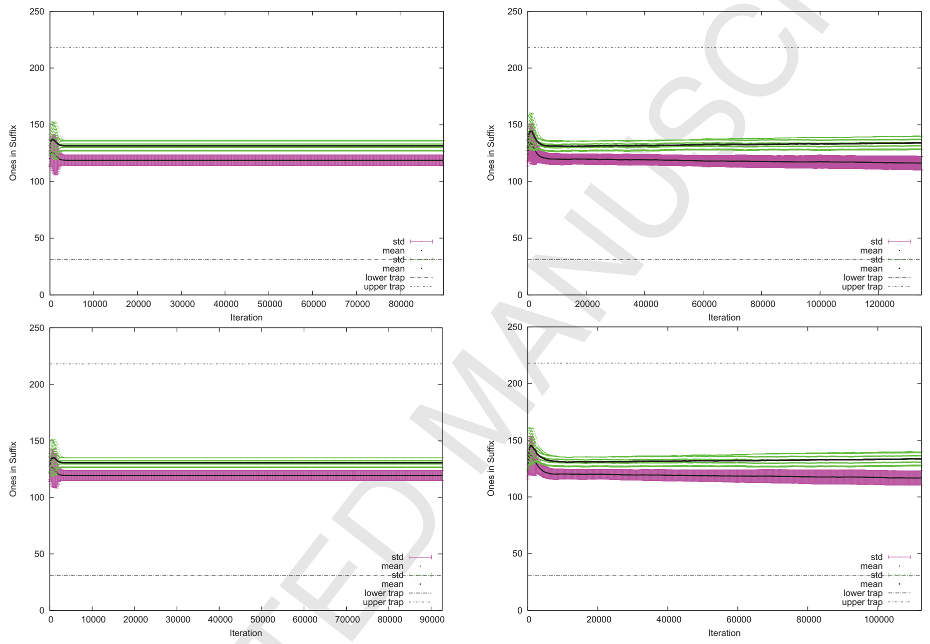


Figure 8: Number of 1-bits within the suffix over 100 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $\mu = 2$, $n = 500$.

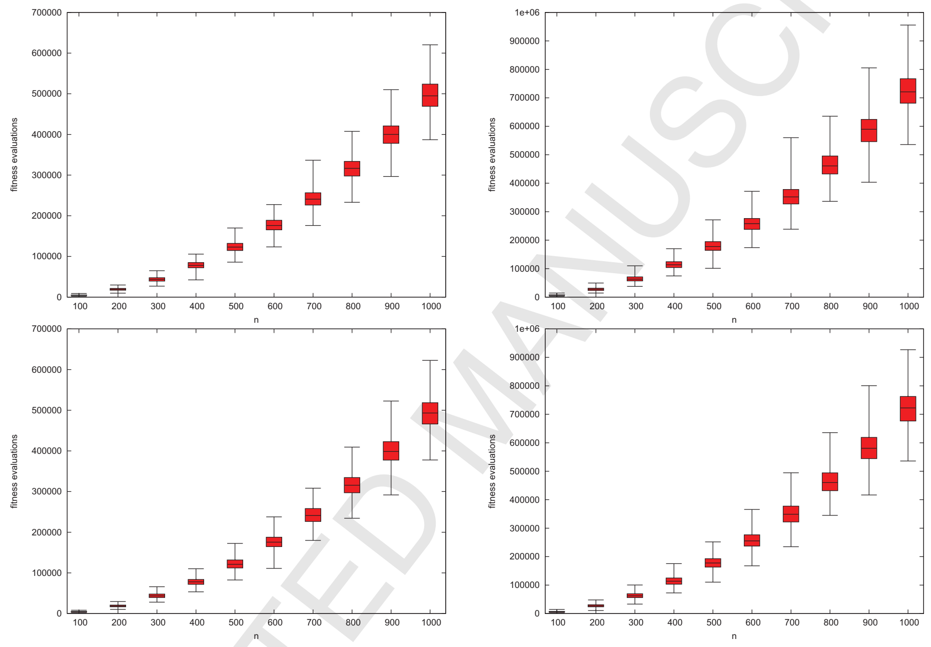
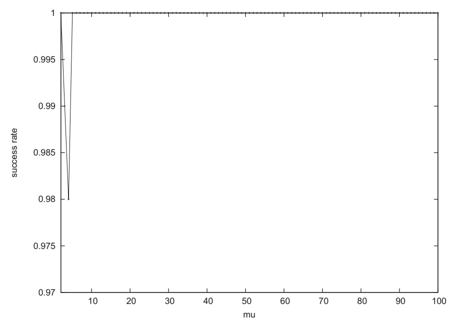


Figure 9: Runtime over 100 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $\mu = 2$ and different values of n.



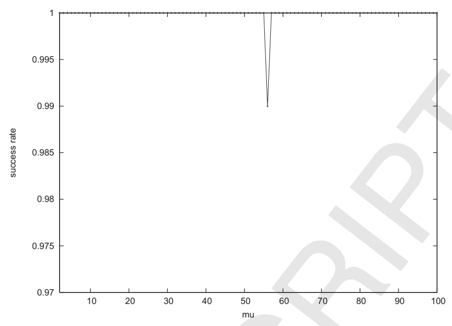


Figure 10: Success rate over 100 runs for fitness sharing with (top) and without (bottom) crowding using local search (left) and standard bit mutations (right), $n = 100$ and different population sizes.

and crowding in combination with 1-bit mutation can already get stuck along the 'path' when using large population sizes.

9. Conclusion

We have considered the Balance function previously used in the literature to show exponential expected runtime of the (1+1) EA at low frequencies of change. We have analysed a (μ+1) EA in its basic version and equipped with several diversity mechanisms to shed light on whether populations and diversity can be sufficiently robust to avoid getting trapped in local optima independent of the frequency of change (i. e., for any value of τ, even when the frequency of change is very low).

Our results show that the basic (μ+1) EA without diversity is inefficient with overwhelming probability for sublinear population sizes (i. e., $\mu \leq n^{1/2-\epsilon}$). The same holds if a genotype diversity mechanism is added to the algorithm. Furthermore we rigorously prove that adding a deterministic crowding mechanism (previously shown elsewhere to be very effective) makes the algorithm extremely inefficient for Balance. On the other hand, independent of the frequency τ, we show that a simple fitness diversity mechanism easily turns the (μ+1) EA into an efficient algorithm for Balance. Finally, fitness sharing can be very effective even for population sizes as small as $\mu = 2$ if used carefully.

We extended our understanding of the (μ+1) EA for Balance with and without the various diversity mechanisms through empirical work. From our experiments it appears that linear population sizes are sufficient to make both the (μ+1) EA without diversity and equipped with genotype diversity efficient for BALANCE. Also there appears to be a sharp threshold at population size cn for some $c > 0$ at which the expected performance of these two algorithms turns from exponential to polynomial. On the other hand, the (μ+1) EA with fitness diversity appears to be effective for population sizes considerably smaller than the sublinear ones required for our proof to work. Concerning fitness sharing, the experiments confirm that using a population size of $\mu = 2$, 1-bit mutation and a crowding mechanism, indeed simplifies the proof. However, for larger population sizes the crowding mechanism not only is not necessary but it also becomes detrimental.

[1] P. S. Oliveto, C. Zarges, Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change, in: Proc. of GECCO, ACM, 2013, pp. 837–844.

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

[3] C.-K. Goh, K. C. Tan, Evolutionary Multi-objective Optimization in Uncertain Environments, Springer, 2009.

[4] R. W. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, Springer, 2004.

[5] K. Weicker, Evolutionary Algorithms and Dynamic Optimization Problems, Der Andere Verlag, 2003.

[6] C. Cruz, J. Gonzales, D. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Computing 15 (7) (2011) 1427–1448.

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

[8] Y. Jin, J. Branke, Evolutionary optimization in dynamic environments – a survey, IEEE Transactions on Evolutionary Computation 9 (3) (2005) 1427–1448.

[9] J. Grefenstette, Genetic algorithms for changing environments, in: Parallel Problem Soving from Nature (PPSN), Elsevier, 1992, pp. 137–144.

[10] H. Andersen, An investigation into genetic algorithms and relationship between speciation and the tracking of optima in dynamic functions, Ph.D. thesis, Queensland University of Technology, Brisbane, Australia, honour Thesis (1991).

[11] Y. Wang, M. WineBerg, Estimation of evolvability genetic algortihm and dynamic environments, Genetic Programming and Evolvable Machines 7 (4) (2006) 355–382.

[12] H. Cheng, S. Yang, Multi-population genetic algorithms with immigrants scheme for dynamic shortest path routing problems in mobile ad hoc networks, in: Applications of Evolutionary Computation (EvoApplications), LNCS 6024, Springer, 2010, pp. 562–571.

[13] F. Oppacher, M. Wineberg, The shifting balance genetic algorithm: Improving the ga in a dynamic environment, in: Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann, 1999, pp. 504–510.

[14] A. Auger, B. Doerr (Eds.), Theory of Randomized Search Heuristics, World Scientific Review, 2011.

[15] T. Jansen, Analyzing Evolutionary Algorithms. The Computer Science Perspective, Springer, 2013.

[16] F. Neumann, C. Witt, Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity, Springer, 2010.

[17] P. Oliveto, J. He, X.Yao, Time complexity of evolutionary algorithms: A decade of results, International Journal of Automation and Computing 4 (3) (2007) 281–293.

[18] S. Droste, Analysis of the (1+1) EA for a dynamically changing onemax-variant, in: Proc. of CEC'02, IEEE Press, 2002, pp. 55–60.

[19] S. Droste, Analysis of the (1+1) EA for a dynamically bitwise changing onemax, in: Proc. of GECCO '03, LNCS 2723, Springer, 2003, pp. 909–921.

[20] T. Jansen, U. Schellbach, Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in the lattice, in: Proc. of GECCO '05, ACM Press, 2005, pp. 841–848.

[21] P. Rohlfshagen, P. K. Lehre, X. Yao, Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change, in: Proc. of Gecco '09, ACM Press, 2009, pp. 1713–1720.

[22] T. Kötzing, H. Molter, Aco beats EA on a dynamic pseudo-boolean function, in: Proc. of PPSN'12, LNCS 7491, Springer, 2012, pp. 113–122.

[23] T. Chen, Y. Chen, K. Tang, G. Chen, X. Yao, The impact of mutation rate on the computation time of evolutionary dynamic optimization, Tech. Rep. arXiv:1106.0566 (2011).

[24] S. Yang, Non-stationary problem optimization using the primal-dual genetic algorithm, in: Proc. of CEC '03, IEEE Press, 2003, pp. 2246–2253.

[25] C. Witt, Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions, Evolutionary Computation 14 (1) (2006) 65–86.

[26] T. Storch, I. Wegener, Real royal road functions for constant population size, Theoretical Computer Science 320 (1) (2004) 123–134.

[27] T. Friedrich, P. S. Oliveto, D. Sudholt, C. Witt, Analysis of diversity-preserving mechanisms for global exploration, Evolutionary Computation 17 (4) (2009) 455–476.

[28] S. W. Mahfoud, Niching methods, in: T. Bäck, D. B. Fogel, Z. Michalewicz (Eds.), Handbook of evolutionary computation, IOP Publishing and Oxford University Press, 1997, pp. C6.1:1–4.

[29] P. S. Oliveto, J. He, X. Yao, Analysis of population-based evolutionary algorithms for the vertex cover problem, in: Proc. of CEC '08, IEEE Press, 2008, pp. 1563–1570.

[30] T. Jansen, R. P. Wiegand, The cooperative coevolutionary (1+1) EA, Evolutionary Computation 12 (4) (2004) 405–434.

[31] M. Hutter, S. Legg, Fitness uniform optimization, IEEE Transactions on Evolutionary Computation 10 (5) (2006) 568–589.

[32] T. Friedrich, N. Hebbinghaus, F. Neumann, Comparison of simple diversity mechanisms on plateau functions, Theoretical Computer Science 410 (26) (2009) 2455–2462.

[33] S. Droste, T. Jansen, I. Wegener, On the analysis of the (1+1) evolutionary algorithm, Theoretical Computer Science 276 (2002) 51–81.

[34] J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms, Artificial Intelligence 127 (1) (2001) 57–85.

[35] D. E. Goldberg, Genetic Algorithms for Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[36] F. Neumann, P. S. Oliveto, C. Witt, Theoretical analysis of fitness-proportional selection: landscapes and efficiency, in: Proc. of GECCO '09, ACM Press, 2009, pp. 835–842.