On the Runtime Analysis of the Clearing Diversity-Preserving Mechanism

Edgar Covantes Osuna and Dirk Sudholt
Department of Computer Science
University of Sheffield, United Kingdom

May 10, 2018

Abstract

Clearing is a niching method inspired by the principle of assigning the available resources among a niche to a single individual. The clearing procedure supplies these resources only to the best individual of each niche: the winner. So far, its analysis has been focused on experimental approaches that have shown that clearing is a powerful diversity-preserving mechanism. Using rigorous runtime analysis to explain how and why it is a powerful method, we prove that a mutation-based evolutionary algorithm with a large enough population size, and a phenotypic distance function always succeeds in optimising all functions of unitation for small niches in polynomial time, while a genotypic distance function requires exponential time. Finally, we prove that with phenotypic and genotypic distances clearing is able to find both optima for TwoMax and several general classes of bimodal functions in polynomial expected time. We use empirical analysis to highlight some of the characteristics that makes it a useful mechanism and to support the theoretical results.

1 Introduction

Evolutionary Algorithms (EAs) with elitist selection are suitable to locate the optimum of unimodal functions as they converge to a single solution of the search space. This behaviour is also one of the major difficulties in a population-based EA, the premature convergence toward a suboptimal individual before the fitness landscape is explored properly. Real optimisation problems, however, often lead to multimodal domains and so require the identification of multiple optima, either local or global (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006).

In multimodal optimisation problems, there exist many attractors for which finding a global optimum can become a challenge to any optimisation algorithm. A diverse population can deal with multimodal functions and can explore several hills in the fitness landscape simultaneously, so they can therefore support global exploration and help to locate several local and global optima. The algorithm can offer several good solutions to the user, a feature desirable in multiobjective optimisation. Also, it provides higher chances to find dissimilar individuals and to create good offspring with the possibility of enhancing the performance of other procedures such as crossover (Friedrich et al., 2009).

Diversity-preserving mechanisms provide the ability to visit many and/or different unexplored regions of the search space and generate solutions that differ in various significant ways from those seen before (Gendreau and Potvin, 2010; Lozano and García-Martínez, 2010). Most analyses and comparisons made between diversity-preserving mechanisms are assessed by means of empirical investigations (Chaiyaratana et al., 2007; Ursem, 2002) or theoretical runtime analyses (Jansen and Wegener, 2005; Friedrich et al., 2007; Oliveto and Sudholt, 2014; Oliveto et al., 2014; Gao and Neumann, 2014; Doerr et al., 2016). Both approaches are important to understand how these mechanisms impact the EA runtime and if they enhance the search for obtaining good individuals. These different results imply where/which diversity-preserving mechanism should be used and, perhaps even more importantly, where they should not be used.

One particular way for diversity maintenance are the niching methods, such methods are based on the mechanics of natural ecosystems. A niche can be viewed as a subspace in the environment that can support different types of life. A specie is defined as a group of individuals with similar features capable of interbreeding among themselves but that are unable to breed with individuals outside their group. Species can be defined as similar individuals of a specific niche in terms of similarity metrics. In EAs the term niche is used for the search space domain, and species for the set of individuals with similar characteristics. By analogy, niching methods tend to achieve a natural emergence of niches and species in the search space (Sareni and Krahenbuhl, 1998).

A niching method must be able to form and maintain multiple, diverse, final solutions for an exponential to infinite time period with respect to population size, whether these solutions are of identical fitness or of varying fitness. Such requirement is due to the necessity to distinguish cases where the solutions found represent a new niche or a niche localised earlier (Mahfoud, 1995).

Niching methods have been developed to reduce the effect of genetic drift resulting from the selection operator in standard EAs. They maintain population diversity and permit the EA to investigate many peaks in parallel. On the other hand, they prevent the EA from being trapped in local optima of the search space (Sareni and Krahenbuhl, 1998). In the majority of algorithms, this effect is attained due to the modification of the process of selection of individuals, which takes into account not only the value of the fitness function but also the distribution of individuals in the space of genotypes or phenotypes (Glibovets and Gulayeva, 2013).

Many researchers have suggested methodologies for introducing niche-preserving techniques so that, for each optimum solution, a niche gets formed in the population of an EA. Most of the analyses and comparisons made between niching mechanisms are assessed by means of empirical investigations using benchmark functions (Sareni and Krahenbuhl, 1998; Singh and Deb, 2006). There are examples where empirical investigations are used to support theoretical runtime analyses and close the gap between theory and practice (Friedrich et al., 2009; Oliveto et al., 2014; Oliveto and Zarges, 2015; Covantes Osuna and Sudholt, 2017; Covantes Osuna et al., 2017).

Both fields use artificially designed functions to highlight characteristics of the studied EAs when tackling optimisation problems. They exhibit such properties in a very precise, distinct, and paradigmatic way. Moreover, they can help to develop new ideas for the design of new variants of EAs and other search heuristics. This leads to valuable insights about new algorithms on a solid basis (Jansen, 2013).

Most of the theoretical analyses are made on example functions with a clear and concrete structure so that they are easy to understand. They are defined in a formal way and allow for the derivation of theorems and proofs allowing to develop knowledge about EAs in a sound scientific way. In the case of empirical analyses, most of the results are based on the analysis of more complex example functions and algorithmic frameworks for a specific set of experiments. This approach allows to explore the general characteristics more easily than in the theoretical field.

Our contribution is to provide a rigorous theoretical runtime analysis for the *clearing* diversity mechanism, to find out whether the mechanism is able to provide good solutions by means of experiments, and we include theoretical runtime analysis to prove how and why an EA is able to obtain good solutions depending on how the population size, *clearing radius*, *niche capacity*, and the dissimilarity measure are chosen.

This manuscript extends a preliminary conference paper (Covantes Osuna and Sudholt, 2017) in the following ways. We extend the theory for large niches by looking into the choice of the population size. While it was known that a population size of $\mu \geq \kappa n^2/4$ is sufficient (Covantes Osuna and Sudholt, 2017), here we show that population sizes of at least $\Omega(n/\text{polylog}(n))$ are necessary to escape from local optima. The reason is that for smaller population sizes, winners in local optima spawn offspring that repeatedly take over the whole population, and this happens before individuals can escape from the optima's basin of attraction. We further extend our analysis to more general classes of examples landscapes defined by Jansen and Zarges (2016), showing that *clearing* is effective across a range of functions with different slopes and optima having different basins of attraction. Finally, we extend the experimental analysis for smaller population sizes of μ with small $n = 30$ and large $n = 100$.

In the remainder of this paper, we first present the algorithmic framework in Section 2. The definition of *clearing*, algorithmic approach, and dissimilarity measures are defined in Section 3. The theoretical analysis is divided in Sections 4 and 5 for small and large niches, respectively. In Section 4 we show how *clearing* is able to solve, for small niches and the right distance function, all functions of unitation, and in Section 5 we show how the population dynamics with large population size in *clearing* solves TWOMAX with the most natural distance function: Hamming distance while it fails with a small population size. In Section 6 we show that the analysis made in Section 5 is general enough to be applied into more general function classes. Section 7 contains the experimental results showing how well our theoretical results matches with empirical results for the general behaviour of the algorithm and providing a closer look into the impact of the population size on performance. We present our conclusions in Section 8, giving additional insight into the dynamic behaviour of the algorithm.

2 Preliminaries

We focus our analysis on the simplest EA with a finite population called (μ+1) EA (hereinafter, μ denotes the size of the current population, see Algorithm 1). Our aim is to develop rigorous runtime bounds of the (μ+1) EA with the *clearing* diversity mechanism. We want to study how diversity helps to escape from some optima. The (μ+1) EA uses random parent selection and elitist selection for survival and has already been investigated by Witt (2006).

Algorithm 1 (μ+1) EA

```
1: Let $t := 0$ and initialise $P_0$ with $\mu$ individuals chosen uniformly at random.
2: while optimum not found do
3:     Choose $x \in P_t$ uniformly at random.
4:     Create $y$ by flipping each bit in $x$ independently with probability $1/n$.
5:     Choose $z \in P_t$ with worst fitness uniformly at random.
6:     if $f(y) \geq f(z)$ then $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ else $P_{t+1} = P_t$ end if
7:     Let $t := t + 1$.
8: end while
```

We consider functions of unitation $f : \{0,1\}^n \to \mathbb{R}$, where $f(x)$ depends only on the number of 1-bits contained in a string x and is always non-negative, i. e., f is entirely defined by a function $u : \{0,\dots,n\} \to \mathbb{R}^+$, $f(x) = u(|x|_1)$, where $|x|_1$ denotes the number of 1-bits in individual x. In particular, we consider the bimodal function of unitation called TwoMax (see Definition 2.1) for the analysis of large niches. TwoMax can be seen as a bimodal equivalent of OneMax. The fitness landscape consists of two hills with symmetric slopes. In contrast to Friedrich et al. (2009) where an additional fitness value for 1^n was added to distinguish between a local optimum 0^n and a unique global optimum, we have opted to use the same approach as Oliveto et al. (2014), and leave unchanged TwoMax since we aim at analysing the global exploration capabilities of a population-based EA.

Definition 2.1 (TwoMax)**.** *A bimodal function of unitation which consists of two different symmetric slopes* ZeroMax *and* OneMax *with 0^n and 1^n as global optima, respectively.*

$$\text{TwoMax}(x) := \max\left\{\sum_{i=1}^{n} x_i, n - \sum_{i=1}^{n} x_i\right\}.$$

The fitness of ones increases with more than $n/2$ 1-bits, and the fitness of zeroes increases with less than $n/2$ 1-bits. These sets are refereed as branches. The aim is to find a population containing both optima (see Figure 1).

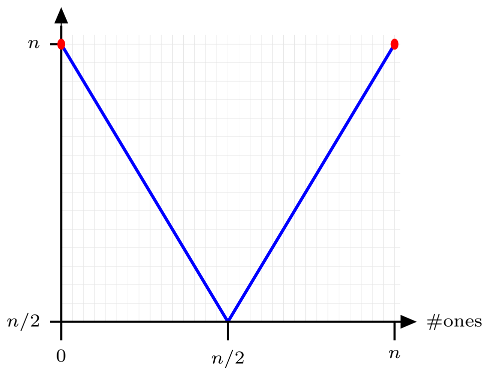


Figure 1: Sketch of the function TwoMax with $n = 30$.

We analyse the expected time until both optima have been reached. Twomax is an ideal benchmark function for *clearing* as it is simply structured, hence facilitating a theoretical analysis, and it is hard for EAs to find both optima as they have the maximum possible Hamming distance. Its choice further allows comparisons with previous approaches like in Friedrich et al. (2009) and Oliveto et al. (2014) in the context of diversity-preserving mechanisms.

The (μ+1) EA with no diversity-preserving mechanism (Algorithm 1) has already been analysed for the Twomax function. The selection pressure is quite low, nevertheless, the (μ+1) EA is not able to maintain individuals on both branches for a long time. Without any diversification, the whole population of the (μ+1) EA collapses into the 0^n branch with probability at least $1/2 - o(1)$ (see Motwani and Raghavan, 1995 for the asymptotic notation) in time n^{n-1}, once the population contains copies of optimum individuals in one of the two branches, it will be necessary to flip all the bits at the same time in order to allow that individual to survive and find both optima on Twomax, so the expected optimisation time is $\Omega(n^n)$ (Friedrich et al., 2009, Theorem 1).

Adding other diversity-preserving mechanisms into the (μ+1) EA like *avoiding genotype or phenotype duplicates* does not work, the algorithm cannot maintain individuals in both branches, so the population collapse into the 0^n branch with probability at least $1/2 - o(1)$ with an expected optimisation time of $\Omega(n^{n-1})$ and $2^{\Omega(n)}$ (Friedrich et al., 2009, Theorem 2 and 3), respectively. *Deterministic crowding* with sufficiently large population is able to reach both optima with high probability in expected time $O(\mu n \log n)$ (Friedrich et al., 2009, Theorem 4).

A *modified version of fitness sharing* is analysed in Friedrich et al. (2009): rather than selecting individuals based on their shared fitness, selection was done on a level of populations. The goal was to select the new population out of the union of all parents and all offspring such that it maximises the overall shared fitness of the population. The drawback of this approach is that all possible size-μ subsets of this union of size $\mu + \lambda$, where λ is the number of offspring, need to be examined. For large μ and λ, this is prohibitive. It is proved that a population-based shared fitness approach with $\mu \geq 2$ reaches both optima of Twomax in expected time $O(\mu n \log n)$ (Friedrich et al., 2009, Theorem 5).

In Oliveto et al. (2014), the performance of the original *fitness sharing* approach is analysed. The analysis showed that using the conventional (phenotypic) sharing approach leads to considerably different behaviours. A population size $\mu = 2$ is not sufficient to find both optima on Twomax in polynomial time: with probability $1/2 + \Omega(1)$ the population will reach the same optimum, and from there the expected time to find both optima is $\Omega(n^{n/2})$ (Oliveto et al., 2014, Theorem 1). However, there is still a constant probability $\Omega(1)$ to find both optima in polynomial expected time $O(n \log n)$, if the two search points are initialised on different branches, and if these two search points maintain similar fitness values throughout the run (Oliveto et al., 2014, Theorem 2).

With $\mu \geq 3$, once the population is close enough to one optimum, individuals descending the branch heading towards the other optimum are accepted. This threshold, that allows successful runs with probability 1, lies further away from the local optimum as the population size increases finding both optima in expected time $O(\mu n \log n)$ (Oliveto et al., 2014, Theorem 3). Concerning the effects of the offspring population, increasing the offspring population of a (μ+λ) EA, with $\mu = 2$ and $\lambda \geq \mu$ cannot guarantee convergence to populations with both optima, i. e., depending on λ one or both optima can get lost, the expected time for finding both optima is $\Omega(n^{n/2})$ (Oliveto et al., 2014, Theorem 4).

3 Clearing

Clearing is a niching method inspired by the principle of sharing limited resources within a niche (or subpopulation) of individuals characterised by some similarities. Instead of evenly sharing the available resources among the individuals of a niche, the *clearing* procedure supplies these resources only to the best individual of each niche: the winner. The winner takes all rather than sharing resources with the other individuals of the same niche as it is done with fitness sharing (Pétrowski, 1996).

Like in fitness sharing, the *clearing* algorithm uses a dissimilarity measure given by a threshold called *clearing radius* σ between individuals to determine if they belong to the same niche or not. The basic idea is to preserve the fitness of the individual that has the best fitness (also called dominant individual), while it resets the fitness of all the other individuals of the same niche to zero[1]. With such a mechanism, two approaches can be considered. For a given population, the set of winners is unique. The winner and all the individuals that it dominates are then fictitiously removed from the population. Then the algorithm proceeds in the same way with the new population which is then obtained. Thus, the list of all the winners is produced after a certain number of steps.

On the other hand, the population can be dominated by several winners. It is also possible to generalise the *clearing* algorithm by accepting several winners chosen among the *niche capacity* κ (best individuals of each niche defined as the maximum number of winners that a niche can accept). Thus, choosing niching capacities between one and the population size offers intermediate situations between the maximum *clearing* ($\kappa = 1$) and a standard EA ($\kappa \geq \mu$).

Empirical investigations made in Pétrowski (1996, 1997a,b); Sareni and Krahenbuhl (1998); Singh and Deb (2006) mentioned that *clearing* surpasses all other niching methods because of its ability to produce a great quantity of new individuals by randomly recombining elements of different niches, controlling this production by resetting the fitness of the poor individuals in each different niche. Furthermore, an elitist strategy prevents the rejection of the best individuals.

We incorporate the *clearing* method into Algorithm 1, resulting in Algorithm 2. The idea behind Algorithm 2 is: once a population with μ individuals is generated, an individual x is selected and changed according to mutation. A temporary population P_t^* is created from population P_t and the offspring y, then the fitness of each individual in P_t^* is updated according to the *clearing* procedure shown in Algorithm 3.

Algorithm 2 (μ+1) EA with clearing

```
1: Let $t := 0$ and initialise $P_0$ with $\mu$ individuals chosen uniformly at random.
2: while optimum not found do
3:    Choose $x \in P_t$ uniformly at random.
4:    Create $y$ by flipping each bit in $x$ independently with probability $1/n$.
5:    Let $P_t^* = P_t \cup \{y\}$.
6:    Update $f(P_t^*)$ with the clearing procedure (Algorithm 3).
7:    Choose $z \in P_t$ with worst fitness uniformly at random.
8:    if $f(y) \geq f(z)$ then $P_{t+1} = P_t^* \setminus \{z\}$ else $P_{t+1} = P_t^* \setminus \{y\}$ end if
9:    Let $t := t + 1$.
10: end while
```

Each individual is compared with the winner(s) of each niche in order to check if it belongs to a certain niche or not, and to check if its a winner or if it is cleared. Here $\mathrm{d}(P[i], P[j])$ is any dissimilarity measure (distance function) between two individuals $P[i]$ and $P[j]$ of population P. Finally, we keep control of the *niche capacity* defined by κ. For the sake of clarity, the replacement policy will be the one defined in Witt (2006): the individuals with best fitness are selected (set of winners) and individuals coming from the new generation are preferred if their fitness values are at least as good as the current ones (novelty is rewarded).

Algorithm 3 Clearing

```
1: Sort $P$ according to fitness of individuals by decreasing values.
2: for $i := 1$ to $|P|$ do
3:    if $f(P[i]) > 0$ then
4:       $winners := 1$
5:       for $j := i + 1$ to $|P|$ do
6:          if $f(P[j]) > 0$ and $\mathrm{d}(P[i], P[j]) < \sigma$ then
7:             if $winners < \kappa$ then $winners := winners + 1$ else $f(P[j]) := 0$ end if
8:          end if
9:       end for
10:   end if
11: end for
```

Finally, as dissimilarity measures, we have considered genotypic or Hamming distance, defined

[1]We tacitly assume that all fitness values are larger than 0 for simplicity. In case of a fitness function f with negative fitness values we can change clearing to reset fitness to $f_{\min} - 1$, where $f_{\min}$ is the minimum fitness value of f, such that all reset individuals are worse than any other individuals.

as the number of bits that have different values in x and y: $\mathrm{d}(x,y) := \mathrm{H}(x,y) := \sum_{i=0}^{n-1} |x_i - y_i|$, and phenotypic (usually defined as Euclidean distance between two phenotypes). As TwoMax is a function of unitation, we have adopted the same approach as in previous work (Friedrich et al., 2009; Oliveto et al., 2014) for the phenotypic distance function, allowing the distance function d to depend on the number of ones: $\mathrm{d}(x,y) := \left| |x|_1 - |y|_1 \right|$ where $|x|_1$ and $|y|_1$ denote the number of 1-bits in individual x and y, respectively.

4 Small Niches

In this section we prove that the (μ+1) EA with phenotypic clearing and a small niche capacity is not only able to achieve both optima of TwoMax but is also able to optimise all functions of unitation with a large enough population, while genotypic clearing fails in achieving such a task (hereinafter, we will refer as phenotypic or genotypic clearing to Algorithm 3 with phenotypic or genotypic distance function, respectively).

4.1 Phenotypic Clearing

First it is necessary to define a very important property of *clearing*, which is its capacity of preventing the rejection of the best individuals in the (μ+1) EA, and once μ is defined large enough, *clearing* and the population size pressure will always optimise any function of unitation.

Note that on functions of unitation all search points with the same number of ones have the same fitness, and for phenotypic clearing with *clearing radius* $\sigma = 1$ all search points with the same number of ones form a niche. We refer to the set of all search points with i ones as niche i. In order to find an optimum for any function of unitation, it is sufficient to have all niches i, for $0 \leq i \leq n$, being present in the population.

In the (μ+1) EA with phenotypic clearing with $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1)\cdot\kappa$, a niche i can only contain κ winners with i ones. The condition on μ ensures that the population is large enough to store individuals from all possible niches.

Lemma 4.1. *Consider the* (μ+1) EA *with phenotypic clearing with* $\sigma = 1$, $\kappa \in \mathbb{N}$ *and* $\mu \geq (n+1)\cdot\kappa$ *on any function of unitation. Then winners are never removed from the population, i. e., if* $x \in P_t$ *is a winner then* $x \in P_{t+1}$*.*

Proof. After the first evaluation with *clearing*, individuals dominated by other individuals are cleared and the dominant individuals are declared as winners. Cleared individuals are removed from the population when new winners are created and occupy new niches. Once an individual becomes a winner, it can only be removed if the size of the population is not large enough to maintain it, as the worst winner is removed if a new winner reaches a new better niche. Since there are at most $n+1$ niches, each having at most κ winners, if $\mu \geq (n+1)\cdot\kappa$ then there must be a cleared individual amongst the $\mu+1$ parents and offspring considered for deletion at the end of the generation. Thus, a cleared individual will be deleted, so winners cannot be removed from the population. □

The behaviour described above means, that with the defined parameters and sufficiently large μ to occupy all the niches, we have enough conditions for the furthest individuals (individuals with the minimum and maximum number of ones in the population) to reach the opposite edges. Now that we know that a winner cannot be removed from the population by Lemma 4.1, it is just a matter of finding the expected time until 0^n and 1^n are found.

Because of the elitist approach of the (μ+1) EA, winners will never be replaced if we assume a large enough population size. In particular, the minimum (maximum) number of ones of any search point in the population will never increase (decrease). We first estimate the expected time until the two most extreme search points 0^n and 1^n are being found, using arguments similar to the well-known fitness-level method (Wegener, 2002).

Lemma 4.2. *Let* f *be a function of unitation and* $\sigma = 1$, $\kappa \in \mathbb{N}$ *and* $\mu \geq (n+1)\cdot\kappa$*. Then, the expected time for finding the search points* 0^n *and* 1^n *with the* (μ+1) EA *with phenotypic clearing on* f *is* $O(\mu n \log n)$*.*

Proof. First, we will focus on estimating the time until the 1^n individual is found (by symmetry, the same analysis applies for the 0^n individual). If the current maximum number of ones in any search point is i, it has a probability of being selected at least of $1/\mu$. In order to create a niche with $j > i$ ones, it is just necessary that one of the $n-i$ zeroes is flipped into 1-bit and the other bits remains unchanged. Each bit flip has a probability of being changed (mutated) of $1/n$ and the probability of the other bits remaining unchanged is $(1-1/n)^{n-1}$. Hence, the probability of creating some niche with $j > i$ ones is at least

$$\frac{1}{\mu} \cdot \frac{n-i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{n-i}{\mu e n}.$$

The expected time for increasing the maximum number of ones, i, is hence at most $(\mu e n)/(n-i)$ and the expected time for finding 1^n is at most

$$\sum_{i=0}^{n-1} \frac{\mu e n}{n-i} = \mu e n \sum_{i=1}^{n} \frac{1}{i} \leq \mu e n \ln n = O(\mu n \log n).$$

Where the summation $\mathrm{H_n} = \sum_{i=1}^{n} 1/i$ is known as the *harmonic number* and satisfies $\mathrm{H_n} = \ln n + \Theta(1)$. Adding the same time for finding 0^n proves the claim. □

Once the search points 0^n and 1^n have been found, we can focus on the time required for the algorithm until all intermediate niches are discovered.

Lemma 4.3. *Let f be any function of unitation, $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1) \cdot \kappa$, and assume that the search points 0^n and 1^n are contained in the population. Then, the expected time until all niches are found with the* $(\mu$+1) EA *with phenotypic clearing on f is $O(\mu n)$.*

Proof. According to Lemma 4.1 and the elitist approach of $(\mu$+1) EA, winners will never be replaced if we assume a large enough population size and by assumption we already have found both search points 0^n and 1^n.

As long as the algorithm has not yet found all niches with at least $n/2$ ones, then there must be an index $i \geq n/2$ such that the population does not cover the niche with i ones, but it does cover the niche with $i+1$ ones. We can mutate an individual from niche $i+1$ to populate niche i. The probability of selecting an individual from niche $i+1$ is at least $1/\mu$, and since it is just necessary to flip one of at least $n/2$ 0-bits with probability $1/n$, we have a probability of at least $1/2$ to do so, and a probability of leaving the remaining bits untouched of $(1-1/n)^{n-1} \geq 1/e$. Together, the probability is bounded from below by $1/(2\mu e)$. Using the level-based argument used before for the case of the niches, the expected time to occupy all niches with at least $n/2$ ones is bounded by

$$\sum_{i=n/2}^{n-1} \frac{2\mu e}{1} \leq 2\mu e n = O(\mu n).$$

A symmetric argument applies for the niches with fewer than $n/2$ ones, leading to an additional time of $O(\mu n)$. □

Theorem 4.4. *Let f be a function of unitation and $\sigma = 1$, $\kappa \in \mathbb{N}$ and $\mu \geq (n+1) \cdot \kappa$. Then, the expected optimisation time of the* $(\mu$+1) EA *with phenotype clearing on f is $O(\mu n \log n)$.*

Proof. Now that we have defined and proved all conditions where the algorithm is able to maintain every winner in the population (Lemma 4.1), to find the extreme search points (Lemma 4.2) and intermediate niches (Lemma 4.3) of the function f, we can conclude that the total time required to optimise the function of unitation f is $O(\mu n \log n)$. □

4.2 Genotypic Clearing

In the case of genotypic clearing with $\sigma = 1$, the $(\mu$+1) EA behaves like the diversity-preserving mechanism called *no genotype duplicates*. The $(\mu$+1) EA with no genotype duplicates rejects the new offspring if the genotype is already contained in the population. The same happens for the $(\mu$+1) EA with genotypic clearing and $\sigma = 1$ if the population is initialised with μ mutually different genotypes (which happens with probability at least $1 - \binom{\mu}{2}2^{-n}$). In other words, conditional on the population being initialised with mutually different search points, both algorithms are identical. In Friedrich et al. (2009, Theorem 2), it was proved that the $(\mu$+1) EA with no genotype duplicates and $\mu = o(n^{1/2})$ is not powerful enough to explore the landscape and can be easily trapped in one optimum of TwoMax. Adapting Friedrich et al. (2009, Theorem 2) to the goal of finding both optima and noting that $\binom{\mu}{2}2^{-n} = o(1)$ for the considered μ yields the following.

Corollary 4.5. *The probability that the* (μ+1) EA *with genotypic clearing,* $\sigma = 1$ *and* $\mu = o(n^{1/2})$ *finds both optima on* TwoMax *in time* n^{n-2} *is at most* $o(1)$*. The expected time for finding both optima is* $\Omega(n^{n-1})$.

As mentioned before, the use of a proper distance is really important in the context of *clearing*. In our case, we use phenotypic distance for functions of unitation, which has been proved to provide more significant information at the time it is required to define small differences (in our case small niches) among individuals in a population, so the use of that knowledge can be taken into consideration at the time the algorithm is set up. Otherwise, if there is no more knowledge related to the specifics of the problem, genotypic clearing can be used but with larger niches as shown in the following section.

5 Large Niches

While small niches work with phenotypic clearing, Corollary 4.5 showed that with genotypic clearing small niches are ineffective. This makes sense as for phenotypic clearing with $\sigma = 1$ a niche with i ones covers $\binom{n}{i}$ search points, whereas a niche in genotypic clearing with $\sigma = 1$ only covers one search point. In this section we turn our attention to larger niches, where we will prove that cleared search points are likely to spread, move, and climb down a branch.

We first present general insights into these population dynamics with *clearing*. These results capture the behaviour of the population in the presence of only one winning genotype x^* (of which there may be κ copies). We estimate the time until in this situation the population evolves a search point of Hamming distance d from said winner, for any $d \leq \sigma$, or for another winner to emerge (for example, in case an individual of better fitness than x^* is found).

These time bounds are very general as they are independent of the fitness function. This is possible since, assuming the winners are fixed at x^*, all other search points within the *clearing radius* receive a fitness of 0 and hence are subject to a random walk. We demonstrate the usefulness of our general method by an application to TwoMax with a *clearing radius* of $\sigma = n/2$, where all winners are copies of either 0^n or 1^n. The results hold both for genotypic clearing and phenotypic clearing as the phenotypic distance of any point x to 0^n (1^n, resp.) equals the Hamming distance of x to 0^n (1^n, resp.).

5.1 Large Population Dynamics with Clearing

We assume that the population contains only one winner genotype x^*, of which there are κ copies. For any given integer $0 \leq d \leq \sigma$, we analyse the time for the population to reach a search point of Hamming distance at least d from x^*, or for a winner different from x^* to emerge. To this end, we will study a potential function φ that measures the dynamics of the population. Let

$$\varphi(P_t) = \sum_{x \in P_t} \mathrm{H}(x, x^*)$$

be the sum of all Hamming distances of individuals in the population to the winner x^*. The following lemma shows how the potential develops in expectation.

Lemma 5.1. *Let* P_t *be the current population of the* (μ+1) EA *with genotypic clearing on any fitness function such that the only winners are* κ *copies of* x^* *and* $\mathrm{H}(x, x^*) < \sigma$ *for all* $x \in P_t$*. Then if no winner different from* x^* *is created, the expected change of the potential is*

$$\mathrm{E}(\varphi(P_{t+1}) - \varphi(P_t) \mid P_t) = 1 - \frac{\varphi(P_t)}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu - \kappa}\right).$$

Before proving the lemma, let us make sense of this formula. Ignore the term $\frac{\kappa}{\mu-\kappa}$ for the moment and consider the formula $1 - \frac{\varphi(P_t)}{\mu} \cdot \frac{2}{n}$. Note that $\varphi(P_t)/\mu$ is the average distance to the winner in P_t. If the population has spread such that it has reached an average distance of $n/2$ then the expected change would be $1 - \frac{\varphi(P_t)}{\mu} \cdot \frac{2}{n} = 1 - \frac{n}{2} \cdot \frac{2}{n} = 0$. Moreover, a smaller average distance will give a positive drift (expected value in the decrease of the distance after a single function evaluation) and an average distance larger than $n/2$ will give a negative drift. This makes sense as a search point performing an independent random walk will attain an equilibrium state around Hamming distance $n/2$ from x^*.

The term $\frac{\kappa}{\mu-\kappa}$ reflects the fact that losers in the population do not evolve in complete isolation. The population always contains κ copies of x^* that may create offspring and may prevent the population from venturing far away from x^*. In other words, there is a constant influx of search points descending from winners x^*. As the term $\frac{\kappa}{\mu-\kappa}$ indicates, this effect grows with κ, but (as we will see later) it can be mitigated by setting the population size μ sufficiently large.

Proof of Lemma 5.1. We use the notation from Algorithm 2, where x is the parent, y is its offspring, and z is the individual considered for removal from the population. If an individual $x \in P_t$ is selected as parent, the expected distance of its mutant to x^* is

$$\mathrm{E}(\mathrm{H}(y, x^*) \mid x) = \mathrm{H}(x, x^*) + \frac{n - \mathrm{H}(x, x^*)}{n} - \frac{\mathrm{H}(x, x^*)}{n} = 1 + \mathrm{H}(x, x^*)\left(1 - \frac{2}{n}\right).$$

Hence after a uniform parent selection and mutation, the expected distance in the offspring is

$$\mathrm{E}(\mathrm{H}(y, x^*) \mid P_t) = \sum_{x \in P_t} \frac{1}{\mu} \cdot \left(1 + \mathrm{H}(x, x^*)\left(1 - \frac{2}{n}\right)\right) = 1 + \frac{\varphi(P_t)}{\mu}\left(1 - \frac{2}{n}\right). \tag{1}$$

After mutation and *clearing* procedure, there are μ individuals in P_t, including κ winners, which are copies of x^*. Let C denote the multiset of these κ winners. As all $\mu - \kappa$ non-winner individuals in P_t have fitness 0, one of these will be selected uniformly at random for deletion. The expected distance to x^* in the deleted individual is

$$\mathrm{E}(\mathrm{H}(z, x^*) \mid P_t) = \sum_{x \in P_t \setminus C} \frac{1}{\mu - \kappa} \cdot \mathrm{H}(x, x^*) = \sum_{x \in P_t} \frac{1}{\mu - \kappa} \cdot \mathrm{H}(x, x^*) = \frac{\varphi(P_t)}{\mu - \kappa}. \tag{2}$$

Recall that the potential is the sum of Hamming distances to x^*, hence adding y and removing z yields $\varphi(P_{t+1}) = \varphi(P_t) + \mathrm{H}(y, x^*) - \mathrm{H}(z, x^*)$. Along with (1) and (2), the expected change of the potential is

$$\begin{aligned}\mathrm{E}(\varphi(P_{t+1}) - \varphi(P_t) \mid P_t) &= \mathrm{E}(\mathrm{H}(y, x^*) \mid P_t) - \mathrm{E}(\mathrm{H}(z, x^*) \mid P_t)\\ &= 1 + \frac{\varphi(P_t)}{\mu}\left(1 - \frac{2}{n}\right) - \frac{\varphi(P_t)}{\mu - \kappa}.\end{aligned}$$

Using that

$$\frac{\varphi(P_t)}{\mu} - \frac{\varphi(P_t)}{\mu - \kappa} = \frac{(\mu - \kappa)\varphi(P_t)}{\mu(\mu - \kappa)} - \frac{\mu\varphi(P_t)}{\mu(\mu - \kappa)} = -\frac{\kappa\varphi(P_t)}{\mu(\mu - \kappa)},$$

the above simplifies to

$$\begin{aligned}\mathrm{E}(\varphi(P_{t+1}) - \varphi(P_t) \mid P_t) &= 1 - \frac{2\varphi(P_t)}{\mu n} - \frac{\kappa\varphi(P_t)}{\mu(\mu - \kappa)}\\ &= 1 - \frac{\varphi(P_t)}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu - \kappa}\right).\end{aligned}$$ □

The potential allows us to conclude when the population has reached a search point of distance at least d from x^*. The following lemma gives a sufficient condition.

Lemma 5.2. *If P_t contains κ copies of x^* and $\varphi(P_t) > (\mu - \kappa)(d - 1)$ then P_t must contain at least one individual x with $\mathrm{H}(x, x^*) \geq d$.*

Proof. There are at most $\mu - \kappa$ individuals different from x^*. By the pigeon-hole principle, at least one of them must have at least distance d from x^*. □

In order to bound the time for reaching a high potential given in Lemma 5.2, we will use the following drift theorem, a straightforward extension of the variable drift theorem (Johannsen, 2010) towards reaching any state smaller than some threshold a. It can be derived with simple adaptations to the proof in Rowe and Sudholt (2014).

Theorem 5.3 (Generalised variable drift theorem)**.** *Consider a stochastic process $X_0, X_1, \ldots$ on $\{0, 1, \ldots, m\}$. Suppose there is a monotonic increasing function $h : \mathbb{R}^+ \to \mathbb{R}^+$ such that*

$$\mathrm{E}(X_t - X_{t+1} \mid X_t = k) \geq h(k)$$

for all $k \in \{a, \ldots, m\}$. Then the expected first hitting time of the set $\{0, \ldots, a-1\}$ for $a \in \mathbb{N}$ is at most

$$\frac{a}{h(a)} + \int_a^m \frac{1}{h(x)}\,\mathrm{d}x.$$

The following lemma now gives an upper bound on the first hitting time (the random variable that denotes the first point in time to reach a certain point) of a search point with distance at least d to the winner x^*.

Lemma 5.4. *Let P_t be the current population of the* (μ+1) EA *with genotypic clearing and $\sigma \leq n/2$ on any fitness function such that P_t contains κ copies of a unique winner x^* and $\mathrm{H}(x, x^*) < d$ for all $x \in P_t$. For any $0 \leq d \leq \sigma$, if $\mu \geq \kappa \cdot \frac{dn-2d+2}{n-2d+2}$ then the expected time until a search point x with $\mathrm{H}(x, x^*) \geq d$ is found, or a winner different from x^* is created, is $O(\mu n \log \mu)$.*

Proof. We pessimistically assume that no other winner is created and estimate the first hitting time of a search point with distance at least d. As φ can only increase by at most n in one step, $h_{\max} := (\mu - \kappa)(d-1) + n$ is an upper bound on the maximum potential that can be achieved in the generation where a distance of d is reached or exceeded for the first time.

In order to apply drift analysis, we define a distance function that describes how close the algorithm is to reaching a population where a distance d was reached. We consider the random walk induced by $X_t := h_{\max} - \varphi(P_t)$, stopped as soon as a Hamming distance of at least d from x^* is reached. Due to our definition of $h_{\max}$, the random walk only attains values in $\{0, \ldots, h_{\max}\}$ as required by the variable drift theorem.

By Lemma 5.1, abbreviating $\alpha := \frac{1}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu-\kappa}\right)$, X_t decreases in expectation by at least $h(P_t) := 1 - \alpha\varphi(P_t) = 1 - \alpha h_{\max} + \alpha h(P_t)$, provided $h(P_t) > 0$. By definition of h and Lemma 5.2, the population reaches a distance of at least d once the distance $h_{\max} - \varphi(P_t)$ has dropped below n. Using the generalised variable drift theorem, the expected time till this happens is at most

$$\frac{n}{1 - \alpha h_{\max} + \alpha n} + \int_n^{h_{\max}} \frac{1}{1 - \alpha h_{\max} + \alpha x}\mathrm{d}x.$$

Using $\int \frac{1}{ax+b}\,\mathrm{d}x = \frac{1}{a}\ln|ax+b|$ (Abramowitz, 1974, Equation 3.3.15), we get

$$\begin{aligned}
&\frac{n}{1 - \alpha h_{\max} + \alpha n} + \left[\frac{1}{\alpha}\ln(1 - \alpha h_{\max} + \alpha x)\right]_n^{h_{\max}} \\
&= \frac{n}{1 - \alpha h_{\max} + \alpha n} + \frac{1}{\alpha} \cdot (\ln(1) - \ln(1 - \alpha h_{\max} + \alpha n)) \\
&= \frac{n}{1 - \alpha h_{\max} + \alpha n} + \frac{1}{\alpha}\ln((1 - \alpha h_{\max} + \alpha n)^{-1}).
\end{aligned}$$

We now bound the term $1 - \alpha h_{\max} + \alpha n$ from below as follows.

$$\begin{aligned}
1 - \alpha h_{\max} + \alpha n &= 1 - (\mu - \kappa)(d-1) \cdot \frac{1}{\mu}\left(\frac{2}{n} + \frac{\kappa}{\mu - \kappa}\right) \\
&= 1 - \frac{2(\mu-\kappa)(d-1) + \kappa(d-1)n}{\mu n} \\
&= \frac{\mu n - 2\mu d + 2\kappa d - \kappa d n + 2\mu - 2\kappa + \kappa n}{\mu n} \\
&= \frac{\kappa}{\mu} + \frac{n - 2d + 2}{n} - \frac{\kappa d n - 2\kappa d + 2\kappa}{\mu n} \\
&\geq \frac{\kappa}{\mu} + \frac{n - 2d + 2}{n} - \frac{n - 2d + 2}{n} \\
&= \frac{\kappa}{\mu}
\end{aligned}$$

where in the penultimate step we used the assumption $\mu \geq \kappa \cdot \frac{dn-2d+2}{n-2d+2}$. Along with $\alpha \geq 2/(\mu n)$, the expected time bound simplifies to

$$\frac{n}{1-\alpha h_{\max}+\alpha n} + \frac{1}{\alpha}\ln((1-\alpha h_{\max}+\alpha n)^{-1}) \leq \frac{n}{\kappa/\mu} + \frac{\mu n}{2}\ln(\mu/\kappa) = O(\mu n \log \mu). \qquad \square$$

The minimum threshold $\kappa \cdot \frac{dn-2d+2}{n-2d+2}$ for μ contains a factor of κ. The reason is that the fraction of winners in the population needs to be small enough to allow the population to escape from the vicinity of x^*. The population size hence needs to grow proportionally to the number of winners κ the population is allowed to store.

Note that the restriction $d \leq \sigma \leq n/2$ is necessary in Lemma 5.4. Individuals evolving within the *clearing radius*, but at a distance larger than $n/2$ to x^* will be driven back towards x^*. If d is significantly larger than $n/2$, we conjecture that the expected time for reaching a distance of at least d from x^* becomes exponential in n.

5.2 Upper Bound for Twomax

It is now easy to apply Lemma 5.4 in order to achieve a running time bound on TWOMAX. Putting $d = \sigma = n/2$, the condition on μ simplifies to

$$\mu \geq \kappa \cdot \frac{dn-2d+2}{n-2d+2} = \kappa \cdot \frac{n^2/2-n+2}{2},$$

which is implied by $\mu \geq \kappa n^2/4$. Lemma 5.4 then implies the following. Recall that for $x^* \in \{0^n, 1^n\}$, genotypic distances $\mathrm{H}(x, x^*)$ equal phenotypic distances, hence the result applies to both genotypic and phenotypic clearing.

Corollary 5.5. *Consider the* (μ+1) EA *with genotypic or phenotypic clearing,* $\kappa \in \mathbb{N}, \mu \geq \kappa n^2/4$ *and* $\sigma = n/2$ *on* TWOMAX *with a population containing* κ *copies of* 0^n (1^n)*. Then the expected time until a search point with at least (at most)* $n/2$ *ones is found is* $O(\mu n \log \mu)$*.*

Theorem 5.6. *The expected time for the* (μ+1) EA *with genotypic or phenotypic clearing,* $\mu \geq \kappa n^2/4$, $\mu \leq \mathrm{poly}(n)$ *and* $\sigma = n/2$ *finding both optima on* TWOMAX *is* $O(\mu n \log n)$*.*

Proof. We first estimate the time to reach one optimum, 0^n or 1^n. The population is elitist as it always contains a winner with the best-so-far fitness. Hence we can apply the level-based argument as follows. If the current best fitness is i, it can be increased by selecting an individual with fitness i (probability at least $1/\mu$) and flipping only one of $n-i$ bits with the minority value (probability at least $(n-i)/(en)$). The expected time for increasing the best fitness i is hence at most $\mu \cdot en/(n-i)$ and the expected time for finding some optimum $x^* \in \{0^n, 1^n\}$ is at most

$$\sum_{i=n/2}^{n-1} \mu \cdot \frac{en}{n-i} = e\mu n \sum_{i=1}^{n/2} \frac{1}{i} \leq e\mu n \ln n.$$

In order to apply Corollary 5.5, we need to have κ copies of x^* in the population. While this isn't the case, a generation picking x^* as parent and not flipping any bits creates another winner x^* that will remain in the population. If there are j copies of x^*, the probability to create another winner is at least $j/\mu \cdot (1-1/n)^n \geq j/(4\mu)$ (using $n \geq 2$). Hence the time until the population contains κ copies of x^* is at most

$$\sum_{j=1}^{\kappa} \frac{4\mu}{j} = O(\mu \log \kappa) = O(\mu \log n)$$

as $\kappa \leq \mu \leq \mathrm{poly}(n)$.

By Corollary 5.5, the expected time till a search point on the opposite branch is created is $O(\mu n \log \mu) = O(\mu n \log n)$. Since the best individual on the opposite branch is a winner in its own niche, it will never be removed. This allows the population to climb this branch as well. Repeating the arguments from the first paragraph of this proof, the expected time till the second optimum is found is at most $e\mu n \ln n$. Adding up all expected times proves the claim. $\square$

One limitation of Theorem 5.6 is the steep requirement on the population size: $\mu \geq \kappa n^2/4$. The condition on μ was chosen to ensure a positive drift of the potential for all populations that haven't reached distance d yet, including the most pessimistic scenario of all losers having distance $d-1$ to x^*. Such a scenario is unlikely as we will see in Sections 7.1 and 7.2 where experiments suggest that the population tends to spread out, covering a broad range of distances. With such spread, a distance of d can be reached with a much smaller potential than that indicated by Lemma 5.2. We conjecture that the (μ+1) EA with clearing is still efficient on TWOMAX if $\mu = O(n)$. However, proving this theoretically may require new arguments on the distribution of the κ winners and losers inside the population.

5.3 On the Choice of the Population Size for Twomax

To get further insights into what population sizes μ are necessary, we show in the following that the (μ+1) EA with *clearing* becomes inefficient on TWOMAX if μ is too small, that is, smaller than $n/\text{polylog}(n)$. The reason is as follows: assume that the population only contains a single optimum x^*, and further individuals that are well within a niche of size $\sigma = n/2$ surrounding x^*. Due to *clearing*, the population will always contain a copy of x^*. Hence there is a constant influx of individuals that are offspring, or, more generally, recent descendants of x^*. We refer to these individuals informally as *young*; a rigorous notation will be provided in the proof of Theorem 5.7. Intuitively, young individuals are similar to x^*, and thus are likely to produce further offspring that are also *young*, i. e., similar to x^* when chosen as parents.

We will show in the following that if the population size μ is small, young individuals will frequently take over the whole population, creating a population where all individuals are similar to x^*. This takeover happens much faster than the time the algorithm needs to evolve a lineage that can reach a Hamming distance $n/2$ to the optimum.

The following theorem shows that if the population size is too small, the (μ+1) EA is unable to escape from one local optimum, assuming that it starts with a population of search points that have recently evolved from said optimum.

Theorem 5.7. *Consider the* (μ+1) EA *with genotypic or phenotypic clearing on* TWOMAX *with* $\mu \leq n/(4\log^3 n)$, $\kappa = 1$ *and* $\sigma = n/2$, *starting with a population containing only search points that have evolved from one optimum* x^* *within the last* $\mu n/32$ *generations. Then the probability that both optima are found within time* $n^{(\log n)/2}$ *is* $n^{-\Omega(\log n)}$.

The following lemma describes a stochastic process that we will use in the proof of Theorem 5.7 to model the number of "young" individuals over time. We are interested in the first hitting time of state μ as this is the first point in time where young individuals have taken over the whole population of size μ. The transition probabilities for states $1 < X_t < \mu$ reflect the evolution of a fixed-size population containing two species (young and old in our case): in each step one individual is selected for reproduction, and another individual is selected for replacement. If they stem from the same species, the size of both species remains the same. But if they stem from different species, the size of the first species can increase or decrease by 1, with equal probability.

This is similar to the *Moran process* in population genetics (Ewens, 2004, Section 3.4) which ends when one species has evolved to fixation (i. e. has taken over the whole population) or extinction. Our process differs as state 1 is reflecting, hence extinction of young individuals is impossible. Notably, we will show that, compared to the original Moran process, the expected time for the process to end is larger by a factor of order $\log \mu$. Other variants of the Moran process have also appeared in different related contexts such as the analysis of Genetic Algorithms (Lemma 6 in Dang et al., 2016) and the analysis of the compact Genetic Algorithm (Lemma 7 in Sudholt and Witt, 2016). The following lemma gives asymptotically tight bounds on the time young individuals need to evolve to fixation.

Lemma 5.8. *Consider a Markov chain* $\{X\}_{t\geq 0}$ *on* $\{1, 2, \ldots, \mu\}$ *with transition probabilities* $\text{Prob}(X_{t+1} = X_t + 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$ *for* $1 \leq X_t < \mu$, $\text{Prob}(X_{t+1} = X_t - 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$ *for* $1 < X_t < \mu$ *and* $X_{t+1} = X_t$ *with the remaining probability. Let* T *be the first hitting time of state* μ, *then for all starting states* X_0,

$$\frac{1}{2} \cdot (\mu - X_0)\mu \ln(\mu - 1) \leq \text{E}(T \mid X_0) \leq 4(\mu - X_0)\mu \text{H}_\text{n}(\mu/2) \leq 4\mu^2 \ln \mu.$$

In addition, if $\mu \leq n$ *then* $\text{Prob}\big(T \geq 8\mu^2 \log^3 n\big) \leq n^{-\log n}$.

Proof. Let us abbreviate $\mathrm{E}_i = \mathrm{E}(T \mid X_t = i)$, then $\mathrm{E}_\mu = 0$, $\mathrm{E}_1 = \frac{\mu^2}{\mu-1} + \mathrm{E}_2$, and for all $1 < i < \mu$ we have

$$\begin{aligned}
\mathrm{E}_i &= 1 + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i+1} + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i-1} + \left(1 - \frac{2i(\mu-i)}{\mu^2}\right) \cdot \mathrm{E}_i \\
\Leftrightarrow \quad \frac{2i(\mu-i)}{\mu^2} \cdot \mathrm{E}_i &= 1 + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i+1} + \frac{i(\mu-i)}{\mu^2} \cdot \mathrm{E}_{i-1} \\
\Leftrightarrow \quad 2\mathrm{E}_i &= \frac{\mu^2}{i(\mu-i)} + \mathrm{E}_{i+1} + \mathrm{E}_{i-1} \\
\Leftrightarrow \quad \mathrm{E}_i - \mathrm{E}_{i+1} &= \frac{\mu^2}{i(\mu-i)} + \mathrm{E}_{i-1} - \mathrm{E}_i.
\end{aligned}$$

Introducing $D_i := \mathrm{E}_i - \mathrm{E}_{i+1}$, this is

$$D_i = \frac{\mu^2}{i(\mu-i)} + D_{i-1}.$$

For D_1 we get

$$D_1 = \mathrm{E}_1 - \mathrm{E}_2 = \left(\frac{\mu^2}{\mu-1} + \mathrm{E}_2\right) - \mathrm{E}_2 = \frac{\mu^2}{\mu-1}.$$

More generally, we expand D_i to get

$$D_i = \sum_{j=1}^{i} \frac{\mu^2}{j(\mu-j)} = \mu^2 \sum_{j=1}^{i} \frac{1}{j(\mu-j)}$$

Now we can express E_i in terms of D_i variables as follows. For all $1 \le i < \mu$,

$$\mathrm{E}_i = \underbrace{(\mathrm{E}_i - \mathrm{E}_{i+1})}_{D_i} + \underbrace{(\mathrm{E}_{i+1} - \mathrm{E}_{i+2})}_{D_{i+1}} + \cdots + \underbrace{(\mathrm{E}_{\mu-1} - \mathrm{E}_\mu)}_{D_{\mu-1}} + \underbrace{\mathrm{E}_\mu}_{0}$$

hence

$$\mathrm{E}_i = D_i + \ldots + D_{\mu-1} = \mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)}$$

We now bound this double-sum from above and below.

$$\begin{aligned}
\mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)} &\le \mu^2 \sum_{k=i}^{\mu-1} \left(\sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} + \sum_{j=\lfloor \mu/2 \rfloor+1}^{k} \frac{1}{j(\mu-j)} \right) \\
&\le \mu^2 \sum_{k=i}^{\mu-1} \left(\sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} + \sum_{j=1}^{\lfloor \mu/2 \rfloor} \frac{1}{j(\mu-j)} \right) \\
&\le \mu^2 \sum_{k=i}^{\mu-1} \frac{4}{\mu} \cdot \mathrm{H_n}(\lfloor \mu/2 \rfloor) = 4(\mu-i)\mu \mathrm{H_n}(\lfloor \mu/2 \rfloor).
\end{aligned}$$

The final inequality follows from $(\mu-i) \cdot \mathrm{H_n}(\lfloor \mu/2 \rfloor) \le \mu \ln \mu$.

The lower bound follows from

$$\begin{aligned}
\mu^2 \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j(\mu-j)} \ge \mu \sum_{k=i}^{\mu-1} \sum_{j=1}^{k} \frac{1}{j} \ge \mu \sum_{k=i}^{\mu-1} \ln(k) &= \mu \ln \left(\prod_{k=i}^{\mu-1} k \right) \\
&\ge \mu \ln \left((\mu-1)^{(\mu-i)/2} \right) \\
&= \frac{1}{2} \cdot (\mu-i)\mu \ln(\mu-1).
\end{aligned}$$

where in the last inequality we used that $(\mu-1-j)(i+j) \ge \mu-1$ for all $0 \le j \le \mu-1$, allowing us to group factors in $\lfloor (\mu-i)/2 \rfloor$ pairs whose product is at least $\mu-1$, leaving a remaining factor of at least $\mu/2 \ge (\mu-1)^{1/2}$ if $(\mu-i)$ is odd.

For the second statement, we use standard arguments on independent phases. By Markov's inequality, the probability that takeover takes longer than $2 \cdot (4\mu^2 \ln \mu)$ is at most $1/2$. Since the upper bound holds for any X_0, we can iterate this argument $\log^2 n$ times. Then the probability that we do not have a takeover in $2 \cdot (4\mu^2 \ln \mu \cdot \log^2 n) \leq 8\mu^2 \log^3 n$ steps (using $\mu \leq n$) is $2^{-\log^2 n} = n^{-\log n}$. □

Now we prove that the time required to reach a new niche with $\sigma = n/2$ is larger than the time required for "young" individuals to take over the population. In other words, once a winner x^* is found and assigned to an optimum, with a small μ, the time for a takeover is shorter than the required time to find a new niche. This will imply that the algorithm needs superpolynomial time to escape from the influence of the winner x^* and in consequence it needs superpolynomial time to find the opposite optimum.

We analyse the dynamics within the population by means of so-called *family trees*. The analysis of EAs with family trees has been introduced by Witt (2006) for the analysis of the (μ+1) EA. According to Witt, a family tree is a directed acyclic graph whose nodes represent individuals and edges represent direct parent-child relations created by a mutation-based EAs. After initialisation, for every initial individual r^* there is a family tree containing only r^*. We say that r^* is the root of the family tree $T(r^*)$. Afterwards, whenever the algorithm chooses an individual $x \in T(r^*)$ as parent and creates an offspring y out of x, a new node representing y is added to $T(r^*)$ along with an edge from x to y. That way, $T(r^*)$ contains all descendants from r^* obtained by direct and indirect mutations.

There may be family trees containing only individuals that have been deleted from the current population. As μ individuals survive in every selection, at least one tree is guaranteed to grow. A subtree of a family tree is, again, a family tree. A (directed) path within a family tree from x to y represents a sequence of mutations creating y out of x. The number of edges on a longest path from the root x^* to a leaf determines the depth of $T(r^*)$.

Witt (2006) showed how to use family trees to derive lower bounds on the optimisation time of mutation-based EAs. Suppose that after some time t the depth of a family tree $T(r^*)$ is still small. Then typically the leaves are still quite similar to the root. Here we make use of Lemma 1 in Sudholt (2009) (which is an adaptation from Lemma 2 and proof of Theorem 4 in Witt, 2006) to show that the individuals in $T(r^*)$ are still concentrated around r^*. If the distance from r^* to all optima is not too small, then it is unlikely that an optimum has been found after t steps.

Lemma 5.9 (Adapted from Lemma 1 in Sudholt, 2009)**.** *For the* (μ+1) EA *with or without clearing, let r^* be an individual entering the population in some generation t^*. The probability that within the following t generations some $y^* \in T(r^*)$ emerges with* $\mathrm{H}(r^*, y^*) \geq 8t/\mu$ *is* $2^{-\Omega(t/\mu)}$.

Lemma 1 in Sudholt (2009) applies to (μ+λ) EAs without *clearing*. We recap Witt's basic proof idea to make the paper self-contained and also to convince the reader why the result also applies to the (μ+1) EA with *clearing*.

The analysis is divided in two parts. In the first part it is shown that family trees are unlikely to be very deep. Since every individual is chosen as parent with probability $1/\mu$, the expected length of a path in the family tree after t generations is bounded by t/μ. Large deviations from this expectation are unlikely. Lemma 2 in Witt (2006) shows that the probability that a family tree has depth at least $3t/\mu$ is $2^{-\Omega(t/\mu)}$. This argument only relies on the fact that parents are chosen uniformly at random, which also holds for the (μ+1) EA with *clearing*.

For family trees whose depth is bounded by $3t/\mu$, all path lengths are bounded by $3t/\mu$. Each path corresponds to a sequence of standard bit mutations, and the Hamming distance between any two search points on the same path can be bounded by the number of bits flipped in all mutations that lead from one search point to the other. By applying Chernoff bounds (see Motwani and Raghavan, 1995) with respect to the upper bound $4t/\mu$ on the expectation instead of the expectation itself (cf. Witt, 2006, page 75), we obtain that the probability of an individual of Hamming distance at least $8t/\mu$ to r^* emerging on a particular path is at most $e^{-4t/(3\mu)}$. Taking the union bound over all possible paths in the family tree still gives a failure probability of $2^{-\Omega(t/\mu)}$. Adding the failure probabilities from both parts proves the claim.

Now, Lemma 5.9 implies the following corollary.

Corollary 5.10. *The probability that, starting from a search point x^*, within $\mu n/16$ generations the* (μ+1) EA *with clearing evolves a lineage that reaches Hamming distance at least $n/2$ to its founder x^* is* $2^{-\Omega(n)}$.

Now we put Lemma 5.8 and Corollary 5.10 together to prove Theorem 5.7.

Proof of Theorem 5.7. By assumption, all individuals in the population are descendants of individuals with genotype x^*, and this property will be maintained over time. This means that every individual x in the population P_t at time t will have an ancestor that has genotype x^* (our notion of *ancestor* and *descendant* includes the individual itself). Tracing back x's ancestry, let $t^* \le t$ be the most recent generation where an ancestor of x has genotype x^*. Then we define the *age* of x as $t - t^*$. Informally, the age describes how much time a search point has had to evolve differences from the genotype x^*. Note that the age of x^* itself is always 0 and as the population always contains a winner x^*, it always contains at least one individual of age 0.

Now assume that a new search point x is created with $\mathrm{H}(x, x^*) \ge n/2$. If x has age at most $\mu n/16$ then there exists a lineage from a copy of x^* to x that has emerged in at most $\mu n/16$ generations. This corresponds to the event described in Corollary 5.10, and by said corollary the probability of this event happening is at most $2^{-\Omega(n)}$. Taking the union bound over all family trees (of which there are at most μ in every generation) and the first $n^{\log n}$ generations, the probability that such a lineage does emerge in any family tree and at any point in time within the considered time span is still bounded by $\mu n^{\log n} \cdot 2^{-\Omega(n)} = 2^{-\Omega(n)}$.

We now show using Lemma 5.8 that it is very unlikely that individuals with age larger than $\mu n/16$ emerge. We say that a search point x is *T-young* if it has genotype x^* or if its most recent ancestor with genotype x^* was born during or after generation T. Otherwise, x is called *T-old*. We omit the parameter "T" whenever the context is obvious. A key observation is that youth is inheritable: if a young search point is chosen as parent, then the offspring is young as well. If an old search point is chosen as parent, then the offspring is old as well, unless mutation turns the offspring into a copy of x^*.

Let X_t be the number of young individuals in the population at time t, and pessimistically ignore the fact that old individuals may create young individuals through lucky mutations. Then in order to increase the number of young individuals, it is necessary and sufficient to choose a young individual as parent (probability X_t/μ) and to select an old parent for replacement. The probability of the latter is $(\mu - X_t)/\mu$ as there are $\mu - X_t$ old parents and the individual to be removed is chosen uniformly at random among μ individuals whose fitness is cleared. Hence, for $1 \le X_t < \mu$, $\mathrm{Prob}(X_{t+1} = X_t + 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$. Similarly, the number of old individuals increases if and only if an old individual is chosen as parent (probability $(\mu - X_t)/\mu$) and a young individual is chosen for replacement (probability X_t/μ), hence for $1 < X_t < \mu$ we have $\mathrm{Prob}(X_{t+1} = X_t - 1 \mid X_t) = X_t(\mu - X_t)/\mu^2$. Otherwise, $X_{t+1} = X_t$. Note that $X_t \ge 1$ since the winner x^* is young and will never be removed. This matches the Markov chain analysed in Lemma 5.8.

Now consider a generation T where all individuals in the population have ages at most $\mu n/32$. By assumption, this property is true for the initial population. At time T, the population contains at least one T-young individual: the winner x^*. By Lemma 5.8, with probability at least $1 - n^{-\log n}$, within the next $8\mu^2 \log^3 n \le \mu n/32$ generations, using the condition $\mu \le n/(4 \log^3 n)$, the population will reach a state where $X_t = \mu$, that is, all individuals are T-young. Assuming this does happen, let $T' \le T + \mu n/32$ denote the first point in time where this happens. Then at time T' all individuals have ages at most $\mu n/32$, and we can iterate the above arguments with T' instead of T.

Each such iteration carries a failure probability of at most $n^{-\log n}$. Taking the union bound over failure probabilities $n^{-\log n}$ over the first $n^{(\log n)/2}$ generations yields that the probability of an individual of age larger than $\mu n/16$ emerging is only $n^{(\log n)/2} \cdot n^{-\log n} = n^{-(\log n)/2}$.

Adding failure probabilities $2^{-\Omega(n)}$ and $n^{-(\log n)/2}$ completes the proof. □

We conjecture that a population size of $\mu = O(n)$ is sufficient to optimise TwoMax in expected time $O(\mu n \log n)$, that is, that the conditions in Theorem 5.6 can be improved.

6 Generalisation to Other Example Landscapes

Note that, in contrast to previous analyses of *fitness sharing* (Friedrich et al., 2009; Oliveto et al., 2014), our analysis of the *clearing* mechanism does not make use of the specific fitness values of TwoMax. The main argument of how to escape from one local optimum only depends on the size of its basin of attraction. Our results therefore easily extend to more general function classes that can be optimised by leaving a basin of attraction of width at most $n/2$.

We consider more general classes of examples landscapes introduced by Jansen and Zarges (2016) addressing the need for suitable benchmark functions for the theoretical analysis of evolutionary algorithms on multimodal functions. Such benchmark functions allows the control of different features like the number of peaks (defined by their position), their slope and their height (provided in an indirect way), while still enabling a theoretical analysis. Since this benchmark setting is defined in the search space $\{0,1\}^n$ and it uses the Hamming distance between two bit strings, it matches perfectly with the current investigation.

Jansen and Zarges (2016) define their notion of a landscape as the number of peaks $k \in \mathbb{N}$ and the definition of the k peaks (numbered $1, 2, \ldots, k$) where the i-th peak is defined by its position $p_i \in \{0,1\}^n$, its slope $a_i \in \mathbb{R}^+$, and its offset $b_i \in \mathbb{R}_0^+$. The general idea is that the fitness value of a search point depends on peaks in its vicinity. The main objective for any optimisation algorithm operating in this landscape is to identify those peaks: a highest peak in exact optimisation or a collection of peaks in multimodal optimisation. A peak has been identified or reached if the Hamming distance of a search point x and a peak p_i is $\mathrm{H}(x, p_i) = 0$. Since we are considering maximisation, it is more convenient to consider $\mathrm{G}(x, p_i) := n - \mathrm{H}(x, p_i)$ instead.

There are three different fitness functions used to deal with multiple peaks in Jansen and Zarges (2016); we consider the two most interesting function classes f_1 and f_2 defined in the following. We only consider genotypic clearing in the following as phenotypic clearing only makes sense for functions of unitation.

Definition 6.1 (Definition 3 in Jansen and Zarges, 2016)**.** *Let $k \in \mathbb{N}$ and k peaks $(p_1, a_1, b_1), (p_2, a_2, b_2), \ldots, (p_k, a_k, b_k)$ be given, then*

- $f_1(x) := a_{\mathrm{cp}(x)} \cdot \mathrm{G}\big(x, p_{\mathrm{cp}(x)}\big) + b_{\mathrm{cp}(x)}$, *called the nearest peak function.*
- $f_2(x) := \max\limits_{i \in \{1,2,\ldots,k\}} a_i \cdot \mathrm{G}(x, p_i) + b_i$, *called the weighted nearest peak function.*

where $\mathrm{cp}(x)$ *is defined by the closest peak (given by its index i) to a search point*

$$\mathrm{cp}(x) := \operatorname*{arg\,min}_{i \in \{1,2,\ldots,k\}} \mathrm{H}(x, p_i).$$

The nearest peak function, f_1, has the fitness of a search point x determined by the closest peak $i = \mathrm{cp}(x)$ that determines the slope a_i and the offset b_i. In cases where there are multiple i that minimise $\mathrm{H}(x, p_i)$, i should additionally maximise $a_i \cdot \mathrm{G}(x, p_i) + b_i$. If there is still not a unique individual, a peak i is selected uniformly at random from those that minimises $\mathrm{H}(x, p_i)$ and those that maximises $a_i \cdot \mathrm{G}(x, p_i) + b_i$.

The weighted nearest peak function, f_2, takes the height of peaks into account. It uses the peak i that yields the largest value to determine the function value. The bigger the height of the peak, the bigger its influence on the search space in comparison to smaller peaks.

6.1 Nearest Peak Functions

We first argue that our results easily generalise to nearest peak functions with two complementary peaks $p_2 = \overline{p_1}$, arbitrary slopes $a_1, a_2 \in \mathbb{R}^+$, and arbitrary offsets $b_i \in \mathbb{R}_0^+$. The generalisation from peaks $0^n, 1^n$ as for TwoMax to peaks $p_2 = \overline{p_1}$ is straightforward: we can swap the meaning of zeros and ones for any selection of bits without changing the behaviour of the algorithm, hence the (μ+1) EA with *clearing* will show the same stochastic behaviour on peaks $0^n, 1^n$ as well as on arbitrary peaks $p_2 = \overline{p_1}$. As for TwoMax, if only one peak x^* has been found, the basin of attraction of the other peak is found once a search point with Hamming distance at least $n/2$ to x^* is generated. If the *clearing radius* is set to $\sigma = n/2$, the (μ+1) EA with *clearing* will create a new niche, and from there it is easy to reach the complementary optimum $\overline{x^*}$. In fact, our analyses from Section 5 never exploited the exact fitness values of TwoMax; we only used information about basins of attraction, and that it is easy to locate peaks via hill climbing. We conclude our findings in the following corollary.

Corollary 6.2. *The expected time for the* (μ+1) EA *with genotypic clearing, $\kappa \in \mathbb{N}$, $\mu \geq \kappa n^2/4$, $\mu \leq \mathrm{poly}(n)$ and $\sigma = n/2$ finding both peaks on any nearest peak function f_1 with two complementary peaks $p_2 = \overline{p_1}$ is $O(\mu n \log n)$.*

If $\mu \le n/(4\log^3 n)$, $\kappa = 1$ and $\sigma = n/2$, and the (μ+1) EA *starts with a population containing only search points that have evolved from one optimum x^* within the last $\mu n/32$ generations, the probability that both optima are found within time $n^{(\log n)/2}$ is $n^{-\Omega(\log n)}$.*

6.2 Weighted Nearest Peak Functions

For f_2 things are different: the larger the peak, the larger is its influence area of the search space in comparison to smaller peaks and thus will have a larger basin of attraction. These asymmetric variants with suboptimal peaks with smaller basin of attraction and peaks with larger basin of attraction are similar to the analysis made in Section 6.1, as long as the parameter σ is set as the maximum distance between the peaks necessary to form as many niches as there are peaks in the solution, and the restriction $0 \le d \le n/2$ of Lemma 5.4 is met, the same analysis can be applied for this instance of the family of landscapes benchmark.

According to f_2 in Definition 6.1, the bigger the height of the peak, the bigger its influence on the search space in comparison to the smaller peaks. Let $\mathcal{B}_i$ denote the basin of attraction of the highest peak p_i, as long as $0 \le \mathcal{B}_i \le n/2$ from Lemma 5.4 it will be possible to escape from the influence of p_i and create a new winner from a new niche with distance $\mathrm{H}(x, p_i) \ge \mathcal{B}_i$. Jansen and Zarges show (Jansen and Zarges, 2016, Theorem 2) that for two complementary peaks $p_2 = \overline{p_1}$ the basin of attraction of p_1 contains all search points x with

$$n - \mathrm{H}(x, p_1) < \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2}.$$

Using that the peaks are complementary, a symmetric statement holds for $\mathcal{B}_2$. Note that in the special case of $a_1 = a_2$ and $b_1 = b_2$ the right-hand side simplifies to $n/2$.

Along with our previous upper bound on TwoMax from Theorem 5.6 it is easy to show the following result for a large class of weighted nearest peak functions f_2.

Theorem 6.3. *For all weighted nearest peak functions f_2 with two complementary peaks $p_2 = \overline{p_1}$ meeting the following conditions on $a_1, a_2 \in \mathbb{R}^+$ and $b_1, b_2 \in \mathbb{R}_0^+$ and the clearing radius σ*

$$f_2(p_1) \le f_2(p_2) \;\Rightarrow\; \frac{a_1}{a_1 + a_2} \cdot n + \frac{b_1 - b_2}{a_1 + a_2} \le \sigma \le \frac{n}{2}$$
$$f_2(p_2) \le f_2(p_1) \;\Rightarrow\; \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2} \le \sigma \le \frac{n}{2}$$

the expected time for the (μ+1) EA *with genotypic clearing, $\kappa \in \mathbb{N}$, $\mu \ge \kappa \cdot \frac{\sigma n - 2\sigma + 2}{n - 2\sigma + 2}$, $\mu \le \mathrm{poly}(n)$ and clearing radius σ finding all global optima of f_2 is $O(\mu n \log n)$.*

Note that in case $f_2(p_1) \ne f_2(p_2)$ there is only one global optimum: the fitter of the two peaks. Then the respective condition (where the left-hand side inequality is true) implies that the basin of attraction of the less fit peak must be bounded by $n/2$. If this condition is not satisfied, the function is deceptive as the majority of the search space leads towards a non-optimal local optimum.

In case $f_2(p_1) = f_2(p_2)$ both peaks are global optima and the conditions require that both basins of attraction have size $n/2$:

$$\sigma = \frac{a_1}{a_1 + a_2} \cdot n + \frac{b_1 - b_2}{a_1 + a_2} = \frac{a_2}{a_1 + a_2} \cdot n + \frac{b_2 - b_1}{a_1 + a_2} = \frac{n}{2}.$$

Proof of Theorem 6.3. The proof is similar to the proof of Theorem 5.6. Assume without loss of generality that $f_2(p_1) \le f_2(p_2)$. Using the same arguments as in said proof (with straightforward changes to the fitness-level calculations), the (μ+1) EA finds one peak in expected time $O(\mu n \log n)$. If this is p_1, the (μ+1) EA still needs to find p_2. By the same arguments as in the proof of Theorem 5.6, the (μ+1) EA's population will contain κ copies of p_1 in expected time $O(\mu \log n)$. Applying Lemma 5.4 with $d = \sigma$ yields that the expected time to find a search point x with Hamming distance at least σ to p_1 is $O(\mu n \log n)$. Since $\frac{a_1}{a_1+a_2} \cdot n + \frac{b_1 - b_2}{a_1+a_2} \le \sigma$, by Theorem 2 in Jansen and Zarges (2016), x is outside the basin of attraction of p_1. As it is also a winner in a new niche, this new niche will never be removed, and p_2 can be reached by hill climbing on a OneMax-like slope from x. By previous arguments, p_2 will then be found in expected time $O(\mu n \log n)$. □

As a final remark, the analysis has shown that it is possible to escape of the basin of attraction of the higher peak with $\mathcal{B} \leq n/2$, this does not mean that the analysis cannot be applied to $\mathcal{B} \geq n/2$. We need to remember that the current investigation considers a distance $d \leq n/2$ because any distance larger than $n/2$ may lead to a exponential expected time in n for reaching a distance of at least d from x^*. One way to avoid this limitation is by dividing the distance d into several niches by setting the parameter $\sigma \leq n/2$ properly. In this analysis we just considered the population dynamics and its ability of escaping a basin of attraction of at most $n/2$ or escaping from a niche with radius at most $n/2$ but it may be possible to generalise the population dynamics for more than 2 niches with sizes $\leq n/2$ by changing/adapting our definition of the potential function. For the time being we rely on experiments in Section 7.3 to show that the population can jump from niches with $\sigma \leq n/2$ allowing to find both optimum in different variants of TWOMAX from the classes of example functions and leave the generalisation of the population dynamics for future theoretical work.

7 Experiments

The experimental approach is focused on the analysis of the $(\mu{+}1)$ EA and is divided in 3 experimental frameworks. Section 7.1 is focused on an empirical analysis for the general behaviour of the algorithm, the relationship between the parameters σ, κ, and μ, and how these parameters can be set. The main objective is to compare our asymptotic theoretical results with empirical data for concrete parameter values.

For the second empirical analysis (Section 7.2), we focus our attention on the population size for small ($n = 30$) and large ($n = 100$) problem sizes. The objective is to observe whether smaller population sizes than $\mu = \kappa n^2/4$ are capable of optimising TWOMAX and compare if the quadratic dependence on n is an artefact of our approach. Also we compare two different forms of initialising the population: the standard uniform random initialisation against a biased initialisation where the whole population is initialised with copies of one peak (0^n for TWOMAX). Biased initialisation is used in order to observe how *clearing* is able to escape from a local optimum and how fast it is compared to a random initialisation.

Finally, for the third analysis (Section 7.3), we show that it is possible to escape from different basin of attractions for weighted peak functions with two peaks in cases where the two peaks are not complementary, but have different Hamming distances.

7.1 General Behaviour

We are interested in observing if the $(\mu{+}1)$ EA with *clearing* is able to find both optima on TWOMAX, so we consider exponentially increasing population sizes $\mu = \{2, 4, 8, \dots, 1024\}$ for just one size of $n = 30$ and perform 100 runs with different settings of parameters σ and κ, so for this experimental framework, we have defined $\sigma = \{1, 2, \sqrt{n}, n/2\}$, $\kappa = \{1, \sqrt{\mu}, \mu/2, \mu\}$ with phenotypic distance since it has been proven that this distance metric works for both cases, small and large niches (when the genotypic distance is used it will be explicitly mentioned).

Since we are interested in proving how good/bad *clearing* is, we define the following outcomes and stopping criteria for each run. *Success*, the run is stopped if the population contains both 0^n and 1^n in the population. *Failure*, once the run has reached 1 million of generations and one of the two (or both) optima are not contained in the population. All the results are shown in Table 1.

For small values of $\sigma = \{1, 2\}$ and $\kappa = 1$, with sufficiently many individuals $\mu = (n/2 + 1) \cdot \kappa$, every individual can create its own niche, and since only one individual is allowed to be the winner, the individuals are spread in the search space reaching both optima with 1.0 success rate. In this scenario, since we are allowing sufficiently many individuals in the population, individuals can be initialised in any of both branches, climbing down the branch reaching the opposite branch and reaching the other extreme optima as shown in Figure 2 (in this case we only show the behaviour of the population with $\mu = \{8, 16, 32\}$, with $\mu \geq 8$ have the same behaviour).

The previous experiment set-up confirms what is mentioned in Section 4.1: with a small *clearing radius*, *niche capacity* and large enough *population size*, the algorithm is able to exhaustively explore the search space without losing the progress reached so far. The population size provides enough pressure to optimise TWOMAX. In this scenario the small differences between individuals allow the algorithm to discriminate between the two branches or optima, this forces to have individuals on both branches or occupy all niches supporting the statement of Theorem 4.4. Individuals with the

Table 1: Success rate measured among 100 runs for the (μ+1) EA with phenotypic clearing on TwoMax for $n = 30$ for the different parameters *clearing radius* σ, *niche capacity* κ and *population size* μ.

$\boldsymbol{\sigma = 1}$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.0	0.05	0.96	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.0	0.0	0.0	0.38	0.89	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.0	0.0	0.0	0.0	0.04	0.20	0.42	0.77	0.97	0.98
μ	0.0	0.0	0.0	0.0	0.01	0.13	0.24	0.55	0.75	0.94

$\boldsymbol{\sigma = 2}$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.02	0.88	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.01	0.03	0.55	0.99	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.0	0.0	0.0	0.0	0.07	0.18	0.48	0.67	0.93	0.99
μ	0.0	0.0	0.0	0.0	0.00	0.04	0.25	0.60	0.80	0.97

$\boldsymbol{\sigma = \sqrt{n}}$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	0.33	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	0.35	0.67	0.97	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	0.40	0.78	0.95	1.0	1.0	1.0	1.0	1.0	1.0	1.0
μ	0.0	0.0	0.0	0.0	0.0	0.07	0.28	0.50	0.80	0.93

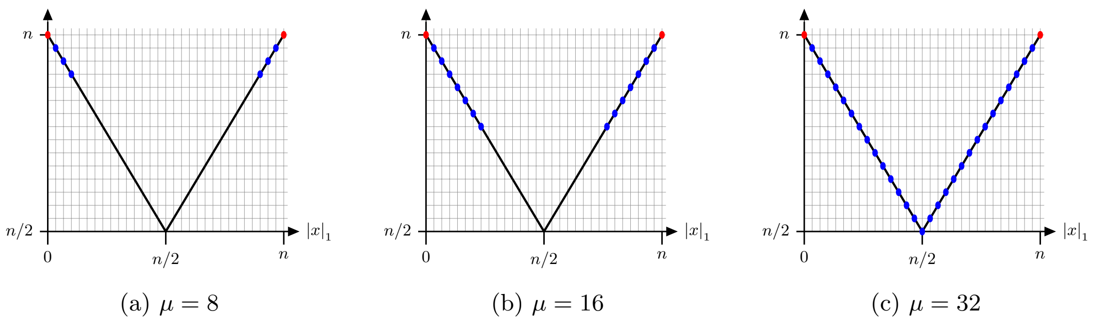
$\boldsymbol{\sigma = n/2}$										
κ	μ									
	2	4	8	16	32	64	128	256	512	1024
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\sqrt{\mu}$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$\mu/2$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
μ	0.0	0.0	0.0	0.0	0.02	0.12	0.29	0.60	0.81	0.98

same phenotype may have a large Hamming distance, creating winners with the same fitness (as proved by Corollary 4.5 in Section 4.2).

It is mentioned in Pétrowski (1996) that while the value of the *niche capacity* $\kappa > \mu/2$ approaches the size of the population, the clearing effect vanishes and the search becomes a standard EA. This effect is verified in the present experimental approach. For a large κ and $\sigma = \{1, 2\}$, one branch takes over, removing the individuals on the other branch reducing the performance of the algorithm. In order to avoid this, it is necessary to define $\mu \geq (n + 1) \cdot \kappa$ to occupy all the winners slots and create new winners in other niches (Theorem 4.4) or increase the *clearing radius* to $\sqrt{n} \leq \sigma \leq n/2$ in order to let more individuals participate in the niche. A reduced *niche capacity* $1 \leq \kappa \leq \sqrt{\mu}$ seems to have a better effect exploring both branches.

Now that theory and practice have shown that a small *clearing radius*, *niche capacity* and large enough population size μ are able to optimise TwoMax, and in order to avoid the take over of a certain branch due to a large *niche capacity* it is necessary to either have: (1) a large enough population, or (2) to increase the *clearing radius*. For (1) we already have defined and proved that one way to overcome this scenario is to define μ according to Theorem 4.4.

In the case of large niches (2), with $\sigma = \{\sqrt{n}, n/2\}$ it is possible to divide the search space in fewer niches. Here the individuals have the opportunity to move, change inside the niche, reach other niches allowing the movement between branches, reaching the opposite optimum. Since it is

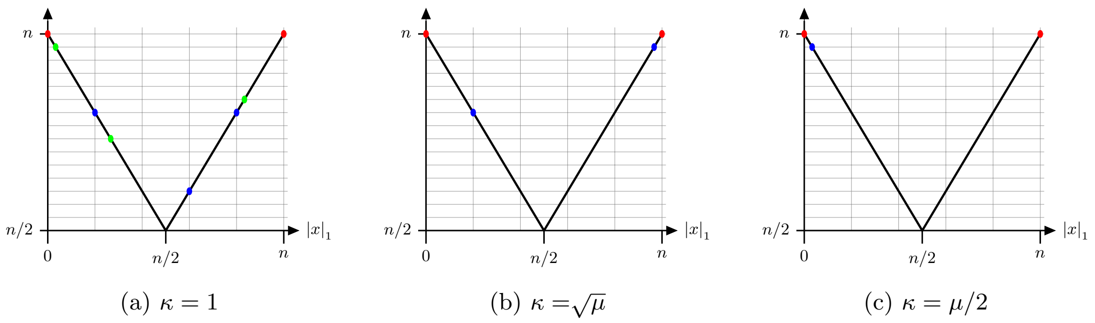


(a) $\mu = 8$ (b) $\mu = 16$ (c) $\mu = 32$

Figure 2: Snapshot of a typical population at the time both optima were reached, showing the spread of individuals in branches of TWOMAX for $n = 30$, $\sigma = 1$ and $\kappa = 1$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners. The rows on the grid represents the fitness value of an individual and its position on TWOMAX and the vertical lines represent the partitioned search space (niches) created by the parameter σ.

possible to reach other niches, defining the *niche capacity* $\kappa = \sqrt{\mu}$ will allow to have more winners in each niche but will still allow to move inside the niche.

For example, with $\sigma = \sqrt{n}$, $\kappa = \sqrt{\mu}$ and $\mu \geq 8$, the algorithm is able to reach both optima with at least 0.97 success rate. In Figure 3 the effect of κ can be seen with sufficiently many individuals. With restrictive *niche capacity* (Figure 3a), the population is scattered in the search space but when the *niche capacity* is increased, the spread is reduced as we allow more individuals to be part of each niche (Figure 3b and 3c). This behaviour can be generalised and is more evident for larger values of μ.

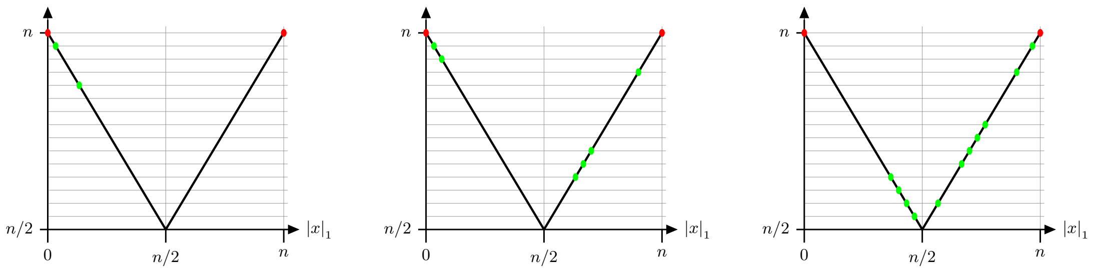


(a) $\kappa = 1$ (b) $\kappa = \sqrt{\mu}$ (c) $\kappa = \mu/2$

Figure 3: Snapshot of a typical population at the time both optima were reached, showing the spread of individuals in branches of TWOMAX for $n = 30$, $\sigma = \sqrt{n}$ and $\mu = 8$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners, and the green points represent cleared individuals. The rows on the grid represents the fitness value of an individual and its position on TWOMAX and the columns represent the partitioned search space (niches) created by the parameter σ.

The theoretical results described in Section 5 are confirmed by the previous experimental results, a large enough population is necessary in order to fill the positions of the winner x^* with κ winners, then force those κ winners with the rest of the population to be subject to a random walk, where it is just necessary for at least one individual to reach the next niche as mentioned in Section 5.1. In the case of TWOMAX, it is after the repetition of moving, climbing down through different niches for a certain period of time when some individual is able to reach both optima as mentioned in Theorem 5.6 and confirmed by the experiments.

Now we have defined the conditions where the algorithm is able to optimise TWOMAX, we can set-up the parameters in a more informed/smart way. With $\mu \geq 2$ it is possible to optimise TWOMAX if σ and κ are chosen appropriately. For example, with $\sigma = n/2$ (as the minimum distance required to distinguish between one branch and the other), $\kappa = \{1, \sqrt{\mu}, \mu/2\}$ and $\mu \geq 2$, the algorithm is able to optimise TWOMAX because there is always an individual moving around that is able to reach a new niche (Figure 4), and finally achieve 1.0 success rate.



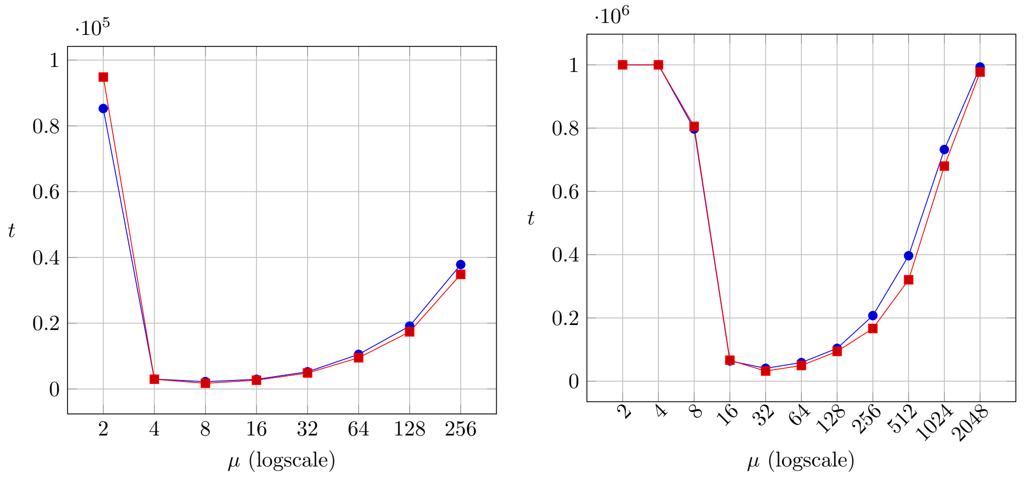
(a) $\mu = 8$ (b) $\mu = 16$ (c) $\mu = 32$

Figure 4: Snapshot of a typical population at the time both optima were reached, showing the spread of individuals in branches of TWOMAX for $n = 30$, $\sigma = n/2$ and $\kappa = 1$. Where the red (extreme) points represent optimal individuals, blue points represent niche winners, and the green points represent cleared individuals. The rows on the grid represents the fitness value of an individual and its position on TWOMAX and the columns represent the partitioned search space (niches) created by the parameter σ.

7.2 Population Size

In this section we address the limitation of Theorem 5.6 related to the steep requirement of the population size: $\mu \geq \kappa n^2/4$. As observed from Section 7.1, experiments suggest that a smaller population size is able to optimise TWOMAX. So for the analysis of the population size we have considered the population size $2 \leq \mu \leq \kappa n^2/4$ in order to observe what is the minimum population size below the threshold $\kappa n^2/4$ able to optimise TWOMAX. With $\sigma = n/2$, and $\kappa = 1$ for $n = \{30, 100\}$ with phenotypic clearing, we report the average of generations of 100 runs, the run is stopped if both optima have been found or the algorithm has reached a maximum of 1 million generations, this is enough time for the algorithms to converge on one or both optima.

Figure 5a shows the average number of generations among 100 runs with $n = 30$. Even for $\mu = 2$ the average runtime is below the 1 million threshold, hence some of the runs were able to find both optima on TWOMAX in fewer than 1 million generations. The reason for this high average runtime is because once both individuals have reached one optimum, it will be one winner, and one loser subjected to a random walk until it gets replaced by an offspring of the winner. This process will continue until the loser reaches a Hamming distance of $n/2$ from the optimum to escape of the basin of attraction. Once this is achieved, it is just necessary for the individual to climb the other branch.



(a) $n = 30$ (b) $n = 100$

Figure 5: The average number of generations measured among 100 runs at the time both optima were found on TWOMAX or $t = 1$ million generations have been reached for $n = 30$ and $n = 100$, $\sigma = n/2$, $\kappa = 1$ and $2 \leq \mu \leq \kappa n^2/4$ for the populations with randomised and biased initialisation (blue and red line, respectively).

Most importantly, all population sizes in the interval $2 \leq \mu \leq \kappa n^2/4$ are able to optimise TwoMax; this experimental setting shows that with a smaller population size for a relatively small $n = 30$ the algorithm is able to optimise TwoMax. Another interesting characteristic of the algorithm is its capacity for escaping from a local optimum.

For $n = 100$, in Figure 5b it is more evident that with a small population size, it is not possible to escape from the basin of attraction of a peak, and the takeover happens before the population has the chance to evolve a distance of at least $n/2$ confirming the theoretical arguments described in Section 5.3. Once the population size is increased, the population is able to escape of the basin of attraction. The most interesting result shown in Figure 5 is that even with a population size $\mu \leq \kappa n^2/4$ the algorithm is able to find both optima on TwoMax (even if runs require more than 1 million generations), indicating that the quadratic dependence on n in $\kappa n^2/4$, is an artefact of our approach.

Finally, for larger μ sizes and $n = 100$ it can be seen that biased initialisation is noticeably faster than random initialisation, and as the population grows the difference between the means grows. One reason could be simply because one peak has already been found, and the algorithm only needs to find the remaining peak.

7.3 Escaping from Different Basins of Attraction

Finally, in this section we show that the runtime analysis used in Section 5.1, and used to prove the theoretical analysis on the general classes of example landscapes functions in Section 6 can be used for weighted peak functions with two peaks of different Hamming distances. For simplicity we restrict our attention to equal slopes and heights: $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$.

We can simplify this class of f_2 functions by using that the (μ+1) EA is *unbiased* as defined by Lehre and Witt (2012): simply speaking, the algorithm treats all bit values and all bit positions in the same way. Hence we can assume without loss of generality that $p_1 = 0^n$. We can further imagine shuffling all bits such that $p_2 = 0^{n-\mathrm{H}(p_1,p_2)}1^{\mathrm{H}(p_1,p_2)}$, which again does not change the stochastic behaviour of the (μ+1) EA. Then all f_2 functions with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ are covered by choosing $p_1 = 0^n$ and p_2 from the set $\{0^n, 0^{n-1}1, 0^{n-2}1^2, \dots, 1^n\}$. As can be seen the peaks p_1 and p_2 can be as close as 0^n and $0^{n-1}1$, or as far as 0^n and 1^n.

It contrast to the simple setting of TwoMax where $\sigma = n/2$ makes most sense, in this more general setting it is necessary to define the *clearing radius* σ according to the Hamming distance between peaks. In particular, the following conditions should be satisfied.

1. $\sigma \leq \mathrm{H}(p_1, p_2)$ as otherwise one peak is contained in the *clearing radius* around the other peak,
2. $\sigma \leq n/2$ as otherwise a niche can contain the majority of search points in the search space, leading to potentially exponential times to escape from the basin of attraction of a local optimum if $\sigma \geq (1 + \Omega(1)) \cdot n/2$, and
3. $\sigma \geq \mathrm{H}(p_1, p_2)/2$ as this is the minimum distance that distinguishes the two peaks.

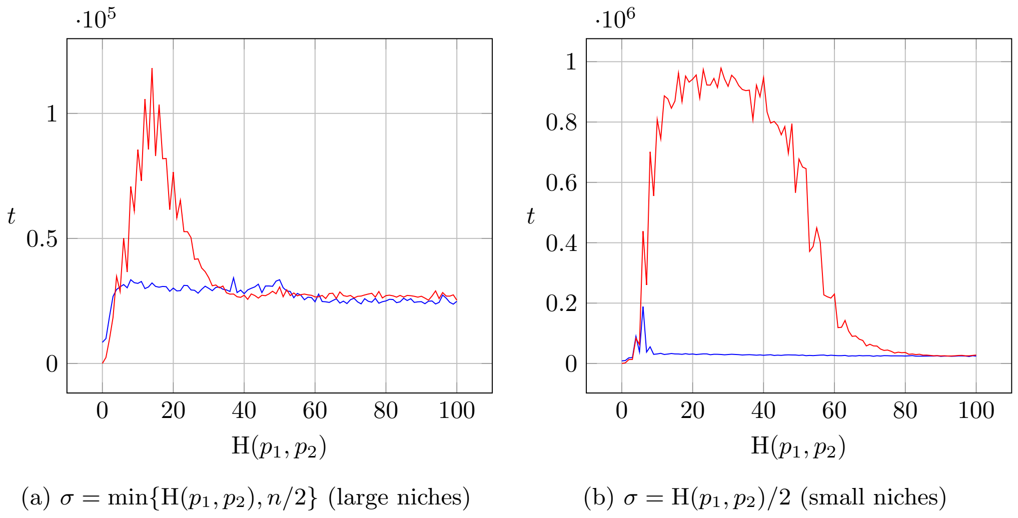
In the following we study two different choices of σ: the maximum value $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$ that satisfied the above constraints, and the minimum feasible value, $\sigma = \mathrm{H}(p_1, p_2)/2$. These two choices allow us to investigate the effect of choosing large or small niches in this setting.

We use genotypic clearing with $\kappa = 1$ and we make use of the results from Section 7.2 to define $\mu = 32$ as a population size able to optimise TwoMax for large $n = 100$. We report the average number of generations of 100 runs, with the same stopping criterion: both optima have been found or the algorithm has reached a maximum of 1 million generations.

For the case of large *clearing radius*, $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$, Figure 6a shows that it is possible to find both optima efficiently across the whole range of $\mathrm{H}(p_1, p_2)$. For random initialisation there are hardly any performance differences, except for a drop in the runtime when the two peaks get very close. For the case of biased populations we see differences by a small constant factor: the closer the peaks, the more difficult it is to escape (or find the new niche) since it requires to flip a specific number of bits to find the other optima. But as the two peaks move away both initialisation methods seem to behave the same indicating that the arguments used in Sections 5.1 and 6 reflect correctly how the algorithm behaves.

Figure 6b shows that with the smallest feasible clearing radius $\sigma = \mathrm{H}(p_1, p_2)/2$ the algorithm is still able to find both optima for all $\mathrm{H}(p_1, p_2)$, but the average runtime for biased initialisation is much higher compared to $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$. From observing the actual population dynamics during a run, it seems that the reason for this high number of generations is because several niches are created around both peaks, i. e., once a peak has been found (and a niche is formed around the peak), the population spreads out by forming many niches between p_1 and p_2.

In the case of biased initialisation it is necessary to jump between specific niches to reach the opposite peak, or make several jumps between different niches in order to escape from its basin of attraction, which leads to this high number of generations.



(a) $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$ (large niches)

(b) $\sigma = \mathrm{H}(p_1, p_2)/2$ (small niches)

Figure 6: The average number of generations measured among 100 runs at the time both peaks $p_1 = 0^n$ and $p_2 = \{0^n, 0^{n-1}1, 0^{n-2}1^2, \ldots, 1^n\}$ were found with $a_1 = a_2 = 1$ and $b_1 = b_2 = 0$ on the fitness landscape defined by f_2 or have reached $t = 1$ million generations for $n = 100$, with genotypic clearing with $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$ and $\sigma = \mathrm{H}(p_1, p_2)/2$, $\kappa = 1$ and $\mu = 32$ for populations with randomised (blue line) and biased (red line) initialisation.

8 Conclusions

The presented theoretical and empirical investigation has shown that *clearing* possesses desirable and powerful characteristics. We have used rigorous theoretical analysis related to its ability to explore the landscape in two cases, small and large niches, and provide an insight into the behaviour of this diversity-preserving mechanism.

In the case of small niches, we have proved that *clearing* can exhaustively explore the landscape when the proper distance and parameters like *clearing radius*, *niche capacity* and population size μ are set. Also, we have proved that *clearing* is powerful enough to optimise all functions of unitation. In the case of large niches, *clearing* has been proved to be as strong as other diversity-preserving mechanisms like *deterministic crowding* and *fitness sharing* since it is able to find both optima of the test function TwoMax.

The analysis made has shown that our results can be easily extended to more general classes of examples landscapes. The analysis done for TwoMax can easily be applied to different classes of bimodal problems using arguments based on how to escape the basin of attraction of one local optimum. We demonstrated this for functions with two complementary peaks and asymmetric variants of TwoMax, consisting of a suboptimal peak with a smaller basin of attraction and an optimal peak with a larger basin of attraction.

Our experimental results suggest that the same efficient performance also applies to bimodal functions where the two peaks have varying Hamming distances. Here *clearing* is able to escape from local optima with different basins of attractions by moving/jumping between niches formed by the *clearing radius*. Defining σ as the smallest possible value that allows to distinguish between peaks creates several small niches, forcing the individuals in the population to make several jumps between niches until an individual can reach the basin of attraction of the other peak. This means that the algorithm requires more generations to find both peaks. But if σ is defined as the maximum feasible value, $\sigma = \min\{\mathrm{H}(p_1, p_2), n/2\}$, the ($\mu$+1) EA is faster and remarkably robust with respect to the Hamming distances between the two peaks. Nevertheless, both approaches allow the population to escape from different basin of attractions.

It remains an open problem to theoretically analyse the population dynamics of *clearing* with more than 2 niches and to prove rigorously that *clearing* is effective across a much broader range of problems, including problems with more than 2 peaks. This involves obtaining more detailed insights into the dynamics of the population, including the distribution and evolution of the losers across multiple niches.

Acknowledgements

The authors would like to thank Carsten Witt for his advice and comments on the analysis of the choice of the population size for Twomax, the anonymous reviewers of this manuscript and the previous FOGA publication for their many valuable suggestions. We also thank the Consejo Nacional de Ciencia y Tecnología — CONACYT (the Mexican National Council for Science and Technology) for the financial support under the grant no. 409151 and registration no. 264342. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 618091 (SAGE).

References

Abramowitz, M. (1974). *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables.* Dover Publications, Incorporated.

Chaiyaratana, N., Piroonratana, T., and Sangkawelert, N. (2007). Effects of diversity control in single-objective and multi-objective genetic algorithms. *Journal of Heuristics*, 13(1):1–34.

Covantes Osuna, E., Gao, W., Neumann, F., and Sudholt, D. (2017). Speeding up evolutionary multi-objective optimisation through diversity-based parent selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 553–560. ACM.

Covantes Osuna, E. and Sudholt, D. (2017). Analysis of the clearing diversity-preserving mechanism. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '17, pages 55–63. ACM.

Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2016). Emergence of diversity and its benefits for crossover in genetic algorithms. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Parallel Problem Solving from Nature – PPSN XIV*, pages 890–900. Springer International Publishing.

Doerr, B., Gao, W., and Neumann, F. (2016). Runtime analysis of evolutionary diversity maximization for oneminmax. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 557–564. ACM.

Ewens, W. (2004). *Mathematical Population Genetics 1: Theoretical Introduction.* Interdisciplinary Applied Mathematics. Springer New York.

Friedrich, T., Hebbinghaus, N., and Neumann, F. (2007). Rigorous analyses of simple diversity mechanisms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '07, pages 1219–1225. ACM.

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

Gao, W. and Neumann, F. (2014). Runtime analysis for maximizing population diversity in single-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '14, pages 777–784. ACM.

Gendreau, M. and Potvin, J.-Y. (2010). Tabu search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, pages 41–59. Springer US.

Glibovets, N. N. and Gulayeva, N. M. (2013). A review of niching genetic algorithms for multimodal function optimization. *Cybernetics and Systems Analysis*, 49(6):815–820.

Jansen, T. (2013). *Analyzing Evolutionary Algorithms: The Computer Science Perspective.* Natural Computing Series. Springer-Verlag Berlin Heidelberg.

Jansen, T. and Wegener, I. (2005). Real royal road functions—where crossover provably is essential. *Discrete Applied Mathematics*, 149(1):111–125.

Jansen, T. and Zarges, C. (2016). Example landscapes to support analysis of multimodal optimisation. In Handl, J., Hart, E., Lewis, P. R., López-Ibáñez, M., Ochoa, G., and Paechter, B., editors, *Proceedings of the Parallel Problem Solving from Nature – PPSN XIV*, pages 792–802. Springer International Publishing.

Johannsen, D. (2010). *Random Combinatorial Structures and Randomized Search Heuristics.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany and the Max-Planck-Institut für Informatik.

Lehre, P. K. and Witt, C. (2012). Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642.

Lozano, M. and García-Martínez, C. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481–497.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms.* PhD thesis, University of Illinois at Urbana-Champaign.

Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms.* Cambridge University Press.

Oliveto, P. S. and Sudholt, D. (2014). On the runtime analysis of stochastic ageing mechanisms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '14, pages 113–120. ACM.

Oliveto, P. S., Sudholt, D., and Zarges, C. (2014). On the runtime analysis of fitness sharing mechanisms. In Bartz-Beielstein, T., Branke, J., Filipič, B., and Smith, J., editors, *Parallel Problem Solving from Nature – PPSN XIII*, pages 932–941. Springer International Publishing.

Oliveto, P. S. and Zarges, C. (2015). Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. *Theoretical Computer Science*, 561:37–56.

Pétrowski, A. (1996). A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 798–803.

Pétrowski, A. (1997a). An efficient hierarchical clustering technique for speciation. In *Artificielle-1997*, pages 22–29.

Pétrowski, A. (1997b). A new selection operator dedicated to speciation. In *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 144–151. Morgan Kaufmann.

Rowe, J. E. and Sudholt, D. (2014). The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38.

Sareni, B. and Krahenbuhl, L. (1998). Fitness sharing and niching methods revisited. *IEEE Transactions on Evolutionary Computation*, 2(3):97–106.

Singh, G. and Deb, K. (2006). Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '06, pages 1305–1312. ACM.

Sudholt, D. (2009). The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410(26):2511–2528.

Sudholt, D. and Witt, C. (2016). Update strength in EDAs and ACO: How to avoid genetic drift. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '16, pages 61–68. ACM.

Ursem, R. K. (2002). Diversity-guided evolutionary algorithms. In Guervós, J. J. M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., and Fernández-Villacañas, J.-L., editors, *Parallel Problem Solving from Nature – PPSN VII*, pages 462–471. Springer Berlin Heidelberg.

Wegener, I. (2002). Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In *Evolutionary Optimization*, pages 349–369. Springer US.

Witt, C. (2006). Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-boolean functions. *Evolutionary Computation*, 14(1):65–86.