ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Impact of Duplicate Avoidance for the $(2+1)$-EA on Monotone Functions

Bachelor Thesis

S. Micheletti

16th July 2020

Advisors: Dr. J. Lengler, Prof. Dr. A. Steger
Department of Computer Science, ETH Zürich

Abstract

Evolutionary algorithms are optimisation heuristics that aim to optimize functions $f : \{0,1\}^n \to \mathbb{R}$. They iteratively evolve a (multi)set of search points $P \subseteq \{0,1\}^n$ using the mutation operator under the Darwin's principle of "survival of the fittest".

In this Bachelor thesis, we study the impact of duplicate avoidance (a simple mechanism to promote diversity in the population P) for the $(2+1)$-EA on HotTopic functions, a class of monotone functions that has proved itself as a successful benchmark in the literature. In multiple situations, promoting diversity in the population is a goal that should be pursued, because the (minor) overhead that they introduce is largely compensated by a performance boost in the optimization time. Here we show that this is not always the case: we will prove that there are instances of HotTopic where the duplicate avoidance mechanism changes the behaviour of the classical $(2+1)$-EA, such that its runtime shifts from quasi-linear to exponential.

Contents

Contents **ii**

1 Introduction **1**
1.1 Theoretical Study of EAs 2
1.2 The Algorithms 3
1.3 Diversity Preserving Mechanisms 5
1.4 HotTopic Functions 8
1.5 Related Work 9
1.6 Our Results 11
1.7 Tools 12

2 Analysis of $(2+1)$-EA with duplicate avoidance **14**
2.1 Preliminaries 15
2.2 Drift of X_i 18
2.3 Analysis of the Drift 34
2.4 Proof of the Runtime 38

3 Conclusions and Future Work **44**

Bibliography **46**

Chapter 1

Introduction

Evolutionary algorithms (EAs) are optimisation heuristics inspired by the natural evolution of species. Countless applications as well as theoretical results have demonstrated that these algorithms are effective on many hard optimization problems. Moreover, they have the advantage that they can be used also when the optimization problem is not well understood and the design of problem-specific algorithms cannot be performed due to the nature of the problem. For the scope of this thesis, they are exploited to maximize a function $f : \{0,1\}^n \to \mathbb{R}$ by iteratively *evolving* a (multi)set of candidate solutions, the *population*. We point out that using EAs for this optimization problem is not always the best choice. In facts, if one has an analytical representation of f or has the possibility to estimate its gradient, then other approaches might be preferable. The EAs that we study here work also if we have access to f only via a black-box evaluation.

We will formally define the structure of EAs in the next section of this chapter, but we already state the general framework for the scope of this thesis. The algorithm starts with a population of μ search points $\{x_1, \dots, x_\mu\} \subseteq \{0,1\}^n$, where each x_i is chosen uniformly at random. The algorithm then proceeds in rounds. In each round, the population is evolved with some mathematical operators under the Darwin's principle of "survival of the fittest". Concretely, in each round, the algorithm chooses some elements from the population (the parents) and uses them to *generate offspring*. Offspring is generated via *mutation*, *i.e.* a new search point is generated from each parent by independently mutating each of its bits with probability $\frac{c}{n}$, where c is called mutation parameter. After the offspring generation, the algorithm evolves the population by keeping the μ fittest search points according to f from the population and the generated offspring. If an element of the offspring is fitter than an element of the starting population, then this new fitter search point will be included in the new population. The ultimate hope is that this artificial evolution will explore the regions of the

search space that have higher fitness in order to find optimal solutions as efficiently as possible. Here our measure for efficiency is given by the number of rounds until the first optimal solution enters in the population.

1.1 Theoretical Study of EAs

EAs can be studied both empirically and theoretically, and both approaches have their advantages and disadvantages. Since in this thesis we will approach the topic from a theoretical perspective, we want to list some arguments on why this is useful. Theoretical studies in this area mostly consider the runtime of algorithms on a pool of benchmarks functions. Both testing a single algorithm on a large number of benchmark functions and testing several algorithms on a single benchmark function is interesting: the former approach allows to detect the strengths and weaknesses of a given algorithm, the latter approach is more useful to create a portfolio of algorithms that are efficient in specific scenarios. Examples of interesting benchmark functions include dynamic functions such as dynamic OneMax (presented in [7]) and static functions such as leading ones, OneMax and jump functions. A (perhaps) surprising observation is the fact that many of those functions such as OneMax (that simply counts the number of one-bits in a bit-string) are much simpler than the functions that one would like to optimize in practice. This is done not because we are not interested in "wild" functions, quite the opposite: by studying simple functions on a large pool of EAs, it is possible to understand strength and weaknesses of a particular EA. Finding weaknesses of an EA on a certain benchmark function is very interesting, because, by making hypothesis on the nature of the limitation, it is possible to design a new algorithm that hopefully works better. This new algorithm is then tested again and, if it works better than the previous one, then it won't fall in the same trap also when it will be used to optimize functions in the real world. This approach of iteratively designing new algorithms is particularly suitable for heuristics that aim to optimize arbitrary functions, and hence a theoretical approach to EAs can lead to precious insights that are useful also in practical contexts.

Monotone functions [1] have been a very successful benchmark in the analysis of evolutionary and genetic algorithms. We stress the fact that monotone functions are trivial to optimize: the maximum is at the string that has only one-bits (in fact, this string dominates all other elements of $\{0,1\}^n$) and there are no local optima (because we can always improve the fitness by flipping an arbitrary zero-bit). Moreover, random local search (*i.e.* the algorithm that flips in each round exactly one bit) optimizes all monotone functions

[1] In the context of functions from the hypercube to real numbers, we say that a function f is monotone if for every $x, y \in \{0,1\}^n$ with $x \neq y$ for which $x_i \geq y_i$ for all $1 \leq i \leq n$ it holds $f(x) > f(y)$.

in $\mathcal{O}(n \log n)$, which can be seen with a coupon collector argument. Although these properties may suggest that every hillclimber optimizes monotone functions efficiently, this is not the case: in 2010 Doerr, Jansen, Sudholt, Winzen and Zarges showed in [2] that the $(1+1)$-EA [2], although it is efficient for all $c < 1$, needs exponential time to optimize some monotone functions for large mutation rates ($c > 16$). Further research (for more details, see Section 1.5) provided some interesting classes of monotone functions, *e.g.* HotTopic functions [3] and noisy linear functions. In this thesis we will use monotone functions, particularly HotTopic functions, to investigate the impact on the mutation rate of a variant of the classical $(2+1)$-EA that has been shown to be useful in other contexts: particularly, we will show that this variant is not useful for HotTopic functions, and hence we will provide a scenario where a technique that was shown to be useful in other situations performs poorly.

1.2 The Algorithms

Now that we have introduced the topic and we have discussed the importance of theoretical analysis in this context, we need to be more precise in the definition of the algorithms. In this section we formally describe the general structure of the classical $(\mu + \lambda)$-EAs and the variant that we will study in Chapter 2.

In their most typical and general form - at each iteration i - EAs manage a population $P_i \subseteq \{0,1\}^n$ of size μ. New offspring is created in every iteration by mutating existing individuals in P_i. Mutation is typically performed by selecting λ elements from P_i uniformly at random and flipping each of their bits independently with probability $\frac{c}{n}$. For $c = 1$, we will further refer to this operator as standard bit mutation. We denote with Q_i the multiset of the λ elements obtained at iteration i with the mutation process. The population P_{i+1} for the next iteration of the algorithm is given by the μ fittest elements (according to f) of the multiset $P_i \cup Q_i$, where here $\cup$ denotes the union of multisets. The general pattern of the $(\mu + \lambda)$-EA is shown in Algorithm 1. An interesting question, both from the theoretical and practical perspective, is the stopping criterion. For the analysis of the runtime, we run the algorithm until the first optimum of f enters the population. This, of course, makes sense only in theory and it is somewhat artificial. More practical stopping criteria could be running the algorithm for a fixed number

[2] The $(1+1)$-EA will be presented as a special case of the $(\mu + \lambda)$-EA in Section 1.2. The algorithm simply keeps a search point x and, at each iteration, it creates x' by flipping each bit of x with a certain probability. After this process, the fittest element between x and x' is kept.

[3] This functions will be the subject of our study and hence will be presented in detail in Section 1.3.

of iterations or until the algorithm converges to some solution. The latter solution is of course sensible to finding local optima and hence requires a clever initialization and/ or multiple restarts.

Algorithm 1: $(\mu + \lambda)$-EA

```
Multiset P_1 ← ∅;
for i = 1, ..., μ do
    Sample x^(i) uniformly at random from {0,1}^n;
    P_1 ← P_1 ∪ {x^(i)};
end
for t = 1, 2, 3, ... do
    Multiset Q_t ← ∅;
    for i = 1, 2, ..., λ do
        Choose x ∈ P_t uniformly at random ;
        Create y by flipping each bit in x independently with
         probability c/n;
        Q_t ← Q_t ∪ {y};
    end
    Multiset P_{t+1} ← ∅ ;
    for i = 1, 2, ..., μ do
        Select x ∈ arg max{f(x)|x ∈ P_t ∪ Q_t} (break ties randomly)
         and update P_{t+1} ← P_{t+1} ∪ {x} ;
    end
end
```

The $(\mu + \lambda)$-EA has multiple parameters, and these have a strong impact on performance. As already mentioned, in this thesis we will focus on the value of the mutation parameter c. As shown by Lengler in [8], the value of c is a matter that critically influences the performance of an EA and it is subject to conflicting goals. On the one hand, if the mutation strength is too low then the progress is also slow, and the algorithm will be susceptible to local optima. On the other hand, if the mutation rate is too high and the parent is close to a global optimum then typically the offspring, even if it has a "good" mutation in it, will also have a large number of detrimental mutations. An interesting result that provides evidence for this argument was shown in [17] by Witt. They showed that the expected runtime for the $(1 + 1)$-EA is $(1 \pm o(1))\frac{e^c}{c} n \ln n$. We note that the runtime goes to infinity either if c approaches zero (*i.e.* if the mutation rate is too low) or if it approaches infinity (*i.e.* if the mutation rate is too high). Another, stronger, result was proved by Lengler and Steger in [10]. They showed that for the $(1 + 1)$-EA, there exists a particularly hard family of functions that has exponential

runtime for $c > c_0 \approx 2.14$, but quasi-linear for $c < c_0$. [4] More generally, a number of results about the impact of c are presented in the literature. We will give an overview of such results later in this chapter.

In this thesis, we will study the impact of c on a variant of Algorithm 1 that is applied in order to promote diversity. We will present diversity preserving mechanism in greater detail in the next section, but we anticipate to the reader that it is widely observed that promoting diversity in the population can have a considerable impact on performance. In general, diversity in the population is needed to explore different regions of the search space and they introduce a trade-off between exploration (*i.e.* considering multiple different regions of the search space) and exploitation (*i.e.* converging quickly to some solution). In the remainder of this section, we just present the algorithm that we will study in Chapter 2: the $(2+1)$-EA with duplicate avoidance (see Algorithm 2). The algorithm very similar to the classical $(2+1)$-EA. The only case that leads to a different behaviour between the two algorithms is the following: when using duplicate avoidance, if the mutation process creates an offspring with the same genotype as one of the elements in population that generated it, the offspring will *always* be discarded, even if it is fitter than one of the elements in the starting population.

Algorithm 2: $(2+1)$-EA with duplicate avoidance

Sample $x^{(1)}$ and $x^{(2)}$ uniformly at random from $\{0,1\}^n$;
$P_1 \leftarrow \{x^{(1)}, x^{(2)}\}$;
for *t=1, 2, 3, ...* **do**
  Choose $x \in P_t$ uniformly at random ;
  Create y by flipping each bit in x independently with probability c/n;
  if *P_t does not contain any element with the same genotype as y* **then**
    $P_{t+1} \leftarrow P_t \cup \{y\}$;
    Select $k \in \arg\min\{f(x) | x \in P_i \cup y\}$ (break ties randomly) and update $P_{t+1} \leftarrow P_{t+1} \backslash \{k\}$;
  end
end

1.3 Diversity Preserving Mechanisms

In this section we motivate why diversity preserving mechanism might help. In order to do that, we take a long tour that continues the discussion on the

[4] Such family of functions is the class of HotTopic functions that we briefly mentioned in Section 1.1 and that we will formally define in Section 1.4.

choice of the parameters that we introduced in the previous section. This allows us on the one hand to terminate our discussion on the importance of the parameters, on the other hand to give a strong argument on why those mechanisms are often useful.

In the previous sections, we said that the mutation parameter c plays a big role, however it is only one of the parameters that have to be chosen. Another choice that we have to do when selecting the most appropriate EA for our purposes, is the choice of the population size μ. It is clear that managing a larger population introduces overhead, but it can also bring benefits: Witt stated in [16] that both theoretical and practical analysis of population-based algorithms have showed a general trend that larger populations are often better. This belief was challenged by Lengler and Zou in [11]. They showed that, in certain situations, increasing the value of μ from one to a larger constant can increase the runtime from quasi-linear to exponential.

Summarising, a larger population can have both positive and negative effects on the efficiency of our algorithm. In this thesis we will study mechanisms that allow to keep some advantages of a larger population and, at the same time, to alleviate its drawbacks. In order to understand such mechanisms, we first need insights about the possible benefits brought by a larger population. A key observation is the fact that a large population may be helpful because it allows for diversity in the algorithm's states. Such diversity may be helpful to explore different areas of the search space, it facilitates global exploration and it allows to escape from local optima. A natural alternative to keep diversity in the population without increasing its size is the introduction of *diversity preserving mechanisms*. Such mechanisms ensure that, in each iteration, the algorithm will have a diverse population that contains dissimilar individuals to promote exploration. Of course, in this scenario, we have a trade-off between *exploration* and *exploitation* : when we use diversity preserving mechanisms we give up a little bit on exploitation to promote exploration. Although this might not always be the best choice (as we will show later), diversity preserving mechanisms have shown to be useful in many cases.

As a first example of benefit brought by diversity preserving mechanisms, consider a large population that collapses to copies of the same genotype before the search space has been properly explored. In this case we still pay the overhead of maintaining a large population, but we can not exploit any benefit from having it. This example might seem quite extreme, but it is not so unlikely: for example, the exponential slowdown given by a larger value of μ studied in [11] contains a similar argument. More generally, the benefits of diversity preserving mechanisms are manifold: they facilitate global exploration, they reduce the risk that the whole population converges to local optima of low fitness, they are better suited for multi-objective optimization

and they are more robust for situations where the function f changes dynamically. Moreover, they are a precious ingredient of genetic algorithms, a variant of EAs which uses the crossover operator between two search points. Although genetic algorithms are a fundamental topic of evolutionary computation and they are often studied together with EAs, we will not further elaborate on them.

In the long history of evolutionary computation, many variants that can be introduced in Algorithm 1 have been proposed to maintain or promote diversity. Many of those techniques work either on the genotypic level, *i.e.* they try to create a diverse set of bit-strings, or on a phenotypic level, *i.e.* they try to obtain different phenotypes, taking into consideration some mapping from genotypes to phenotypes. The duplicate avoidance mechanism, *i.e.* the variant that we proposed in Algorithm 2 that considers the population as a set instead of a multiset, is an example of diversity preserving mechanism that works on the genotypic level. In general, as stated in [14], there is a plethora of mechanisms that can be applied as a variant of Algorithm 1, and it is often not clear what the best strategy is. A theoretical approach to the topic is interested in determining which diversity mechanisms work well for a given problem, which do not, and most importantly in understanding the reasons that lead to such behaviours. This would allow to get precious insights in order to design new diversity mechanisms that work well in an hopefully broad context. We conclude this section by listing some diversity preserving mechanisms that have been studied. For a more extensive list we refer to [13], while we list some other relevant results in Section 1.5.

- Duplicate avoidance, *i.e.* the algorithm that we study in this thesis.
- Fitness diversity, that does not allow the population to contain two search points with the same fitness.
- Deterministic crowding, where the offspring competes only with its parent and replaces it if and only if it has at least the same fitness.
- Fitness sharing, that derates the real fitness of a search point x based by its similarity to other individuals in the population. Concretely, an offspring y enters the population if it is better, considering a linear combination that considers fitness and similarity to other points, than the element z with the worst fitness in the population. For more details we refer to [12].
- Clearing, a niching mechanism similar to fitness sharing. It uses the same idea of similarity to create a new fitness function f', where some "winners" element keep their original fitness, and the other elements have their fitness decreases to zero. The EA then uses f' instead of f to select the fittest individuals.

- Ageing, where search points with a certain age are discarded after they have reached a maximal age to promote diversity.

1.4 HotTopic Functions

In this section we present HotTopic functions, a particularly hard class of monotone functions introduced by Lengler and Steger in [10] that will be the objective function studied in this thesis. HotTopic functions can easily be optimized by some algorithms, but other algorithms fail miserably on them and need exponential time. The high discriminative power of these functions is their main advantage and results suggest that the choice of the mutation parameter plays a crucial role in their performance.

The intuition on HotTopic is the following. There are different parts of the search space which we call levels. Locally, in the ℓ-th level, the fitness is given by the linear function $f_\ell(x) = \sum_{i \in \ell\text{-th level}} x_i \cdot n + \sum_{i \notin \ell\text{-th level}} x_i$. An additional term ensures that any search point of level ℓ is fitter than any search point of level $\ell - 1$. The regions ("levels") are constructed in such a way that flipping zero-bits into one-bits can only increase the level. The string 1...1 lies in the region that corresponds to the highest level, and thus it is the global optimum. This construction implies that there is always a "hot topic", *i.e.* a subset of the search point (corresponding to the level) that has a much greater influence on the fitness than the rest. Focusing too much on this hot topic will lead to a behaviour that is very good at optimising this particular bits, but the rest of the string will deteriorate because it is out of focus. For example, a mutation that flips a single zero bit in the hot topic and several one bits outside the hot topic will always be accepted, although the Hamming distance with respect to the optimum increases[5]. Thus if the hot topic is sufficiently narrow and changes often, then flips of zero bits in the hot topic will be overcompensated by flips of one bits in the neglected parts, which leads to overall stagnation.

We now formally define HotTopic functions by closely following [11]. The function is defined relatively to five parameters: $n \in \mathbb{N}$, $0 < \beta < \alpha < 1$, $0 < \varepsilon < 1$ and $L \in \mathbb{N}$. Throughout this thesis we assume that α and c (but not ε) are constants, while L is exponentially large on n. We denote with HT the function $\text{HotTopic}_{n,\alpha,\beta,\varepsilon,L}$. For $1 \leq i \leq L$ we choose sets $A_i \subseteq [n]$ of size αn independently and uniformly at random, and we choose subsets $B_i \subseteq A_i$ of size βn uniformly at random. We define the level $\ell(x)$ of a search point $x \in \{0,1\}^n$ by

[5]Here we mention the Hamming distance because is a natural way to quantify the proximity to the optimum. In the next chapters we will generalize this measure with the potential, a crucial concept in the analysis of EAs.

$$\ell(x) := \max_{\ell' \in [L]} \{ |\{j \in B_{\ell'} : x_j = 0\}| \leq \varepsilon\beta n \}$$

where we set $\ell(x) = 0$ if no such ℓ' exists. Then we define $f : \{0,1\}^n \to \mathbb{R}$ as follows:

$$HT(x) := \ell(x) \cdot n^2 + \sum_{i \in A_{\ell(x)+1}} x_i \cdot n + \sum_{i \in R_{\ell(x)+1}} x_i$$

where $R_{\ell(x)+1} := [n] \setminus A_{\ell(x)+1}$, and where we set $A_{L+1} := B_{L+1} := \emptyset$. So the set $A_{\ell(x)+1}$ defines the hot topic while the algorithm is at level ℓ. We see that this function give a high weight coefficient to the current hot topic, while all other bits have coefficient one. The initial term $\ell(x)n^2$ is introduced for two reasons. First, this term makes the function non-linear: this is necessary because, as shown in [10], any linear function has quasi-linear time for all constants c. Moreover, with this construction, the level always increases and when the top level is reached, then the string will converge to the optimal solution in $\mathcal{O}(n \log n)$ by a coupon collector argument. [6] We observe that the level ℓ increases if the density of zero-bits in $B_{\ell'}$ drops below ε for some $\ell' > \ell$. With high probability this only happens if the density of one bits in $A_{\ell+1}$ and in the whole string is also roughly ε, up to some constant factors. Hence, the parameter ε determines how far away the algorithm is from the optimum when the level changes. One can check that this function is monotone. Indeed, assume that x is dominated by y, *i.e.* $x_i \leq y_i$ for all $1 \leq i \leq n$. Then $\ell(x) \leq \ell(y)$. If $\ell(x) = \ell(y)$ then it is obvious that $f(x) \leq f(y)$ and the inequality is strict if $x \neq y$. On the other hand, if $\ell(x) < \ell(y)$, then $f(x) < \ell(x)n^2 + n^2 \leq \ell(y)n^2 \leq f(y)$ as desired.

1.5 Related Work

The analysis of EAs on monotone functions started in 2010 by the work of Doerr, Jansen, Sudholt, Winzen and Zarges [2][3]. They showed that the difficulty of optimising monotone functions depends on the value of the mutation parameter c. Particularly, they showed that the $(1+1)$-EA, needs time $\mathcal{O}(n \log n)$ on all monotone functions if $c < 1$. They also showed that, for large mutation rates ($c > 16$), there are monotone functions for which the $(1+1)$-EA needs exponential time. Their construction of hard monotone functions was later simplified by Lengler and Steger in [8], who improved

[6] Throughout this thesis, whenever we use the asymptotic notation such as in $x = \mathcal{O}(y)$, we mean that for every α and c there is a constant C (that may depend on α and c, but not on ε) such that $x \leq Cy$.

the range for c from $c > 16$ to $c > 2.14$ (in order to do this, they introduced the HotTopic functions). For a long time, it was an open question whether $c = 1$ is a threshold at which the runtime switches from polynomial to exponential. Lengler, Martinsson and Steger showed in [9] that $c = 1$ is not a threshold, showing by an information compression argument an $\mathcal{O}(n \log^2 n)$ bound for all $c \in [1, 1 + \varepsilon]$ for some $\varepsilon > 0$. Recently, the limits of our understanding of monotone functions were pushed significantly by Lengler in [8], who analyzed monotone functions for a mainfold of evolutionary and genetics algorithms. By analysing the algorithms on HotTopic functions, he found sharps thresholds on the values of parameter c, such that on one side of the threshold the runtime was $\mathcal{O}(n \log n)$, while on the other side of the threshold it was exponential. These algorithm include the $(1 + 1)$-EA, the $(1 + \lambda)$-EA and the $(\mu + 1)$-EA, for which the threshold was $c_0 \approx 2.14$. He also generalized the argument of the seminal papers on EAs on monotone functions by showing that for mutation parameter $c < 1$ and for every constant $\lambda \in \mathbb{N}$, with high probability the $(1 + \lambda)$-EA optimizes every monotone function in $\mathcal{O}(n \log n)$ steps.

Several results about diversity preserving mechanisms have been shown, both theoretically and empirically. Regarding empirical studies, we mention [1] (that studied a diversity control mechanism that implicitly implied the avoidance of duplicates in the population and observed that such a mechanisms brought benefits to both single-objective benchmark functions including OneMax and multi-objective benchmarks) and [15], that observed that promoting diversity in the population brought considerable advantages in the number of evaluations of the fitness function, which is highly desirable for real-world applications, because the evaluation is often the time-critical factor in such applications. Many of those studies observed that the right use of those diversity strategies can play a key role for the success of an EA. As a first example, we mention the use of diversity mechanisms in a multi-modal setting, *e.g.* on TwoMax, a simple bi-modal function where the fitness of a search point x is defined as $\max\{\sum_{i=1}^{n} x_i, n - \sum_{i=1}^{n} x_i\}$: here the goal is not finding a single optimum, but ending in a population that contains both the all-zero string and the all-one string. For this function, the runtime of the $(\mu + 1)$-EA with constant population size needs exponential time to find both optima with high probability and in expectation. In [5], it was shown that using duplicate avoidance does not help. On the other hand, in the same paper, it was shown that deterministic crowding, fitness sharing and clearing are helpful since they decrease the runtime to find both optima from exponential to quasi-linear for constant population sizes. Particularly relevant to us is what is known as the first rigorous theoretical study about the impact on runtime of diversity preserving mechanisms: the work presented by Friedrich, Hebbinghaus and Neumann in [4], who compared a genotypic and phenotypic diversity mechanism on an artificially

constructed uni-modal problem. An interesting result that they proved is the introduction of a benchmark function (called PL function): PL can be efficiently optimized by the $(2+1)$-EA with duplicate avoidance and standard bit mutation, but leads to exponential runtime of the $(2+1)$-EA with fitness diversity and standard bit mutation. This suggests that implementing duplicate avoidance on the $(2+1)$-EA can be helpful since it makes the runtime go from exponential to quasi-linear in some cases. However, in this paper we show an example where the trend is exactly the opposite.

1.6 Our Results

We have introduced EAs and the importance of a rigorous theoretical analysis. We want to get insights about possible traps, such that we can develop better algorithms or chose a good-performing algorithm in a specific practical context. We also mentioned that the general trend in the literature suggests that promoting diversity in the population is a goal that should be pursued: often the (minor) overhead necessary to promote diversity can be largely compensated by the performance boost in the optimization time.

In this thesis we argue that this is not always the case. Concretely, we present an example where implementing a diversity preserving mechanism hurts: on some instance of HotTopic close to the optimum, the $(2+1)$-EA with duplicate avoidance (see Algorithm 2) is less efficient than the classical $(2+1)$-EA for all $c \in [1.66, 2.14]$. The intuitive explanation of such a behaviour is that if Algorithm 2 considers a population $\{x, y\}$ with $f(x) > f(y)$, then it won't accept any offspring that is a duplicate of x. This is bad because, in the context of a monotone benchmark function, the population $\{x, x\}$ is better than $\{x, y\}$: this follows from the fact that the total amount of one-bits is larger in the former population than in the latter. Moreover, the classical $(2+1)$-EA allows to create a population $\{x, x\}$ starting from $\{x, y\}$ with constant probability: it is sufficient to choose x as parent and not touching any of its bits (which happens with probability $\approx e^{-c}$). Duplicate avoidance forbids this shortcut: in order to create a population with the same total amount of one-bits of $\{x, x\}$ starting from $\{x, y\}$, it needs to flip at least a zero-bit which, close to the optimum, happens rarely. In Chapter 2 we provide more arguments for this result, together with a rigorous proof based on drift analysis. We stress that the fact that the classical $(2+1)$-EA is efficient for a larger interval of mutation parameter than the $(2+1)$-EA with duplicate avoidance is a good certificate for its superiority: this is not because we want to be efficient in optimizing HotTopic (after all, this is a monotone function), but because this result gives us insights on why duplicate avoidance hurts. Moreover, it is desirable to get algorithms that are efficient for a broad choice of parameters, such that the choice of those parameters becomes less of a concern for practical purposes.

1.7 Tools

Before we begin our journey in the analysis of the $(2+1)$-EA with duplicate avoidance on HotTopic functions close to the optimum, we report some relevant tools that we will use throughout the proofs. This is not strictly necessary because we just state some results without any further explanation or formal proof, but we prefer to make the thesis self-contained for the sake of completeness.

We begin with the Chernoff bounds, that we often use to obtain good tail bounds.

Theorem 1.1 (Chernoff Bound, Theorem 1 in [11]) *Let $X_1, \dots, X_n$ be independent random variables (not necessarily i.i.d) that take values in $[0,1]$. Let $S = \sum_{i=1}^{n}$ and $\mu = \mathbb{E}[S]$. Then for all $0 \leq \delta \leq 1$,*

$$Pr[S \leq (1-\delta)\mu] \leq e^{-\delta^2 \frac{\mu}{2}}$$

and for all $\delta \geq 0$,

$$Pr[S \geq (1+\delta)\mu] \leq e^{-\min(\delta^2,\delta)\frac{\mu}{3}}$$

Then we report two important theorems in drift analysis: we will use them to prove them the runtime of our algorithm.

Theorem 1.2 (Negative Drift Theorem, Theorem 12 in [6]) *Let $(X_t)_{t\geq 0}$ be a sequence of random variables and let $d \in \mathbb{R}$. If, for all $t \geq 0$,*

$$\mathbb{E}[X_{t+1} - X_t \mid X_0, \dots, X_t] \leq d,$$

then $(X_t - dt)_{t\geq 0}$ is a supermantigale. If further $(X_t - dt)_{t\geq 0}$ is (c,δ)-sub-Gaussian, the, for all $t \geq 0$ and all $x > 0$,

$$Pr\left[\max_{0\leq j\leq t}(X_j - X_0) \geq dt + x\right] \leq e^{-\frac{x}{2}\min(\delta, \frac{x}{ct})}$$

Theorem 1.3 (Theorem 4 in [10]) *Let $(X_t)_{t\in\mathbb{N}_0}$ be a Markov chain with state space $\mathcal{S} \subseteq \{0\} \cup [1,\infty)$ and with $X_0 = n$. Let T be the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose furthermore that there is a positive increasing function $h : [1,\infty) \to \mathbb{R}_{>0}$ such that for all $x \in \mathcal{S}$, $x > 0$ we have for all $t \geq 0$*

$$\mathbb{E}[X_{t+1} \mid X_t = x] \leq x - h(x).$$

Then

$$\mathbb{E}[T] \leq \frac{1}{h(1)} + \int_1^n \frac{1}{h(u)} du$$

Chapter 2

Analysis of $(2+1)$-EA with duplicate avoidance

In this chapter we study the impact of the duplicate avoidance mechanism on the $(2+1)$-EA (see Algorithm 2). Recall that the $(2+1)$-EA with duplicate avoidance is very similar to the classical $(2+1)$-EA, but it avoids that the population collapses to two identical search points. The goal of this chapter is studying the impact of the duplicate avoidance mechanism on the mutation parameter c. This question, of course with different algorithms and/ or fitness functions, has been the subject of a number of other papers. Particularly relevant to us is the work [8] by Lengler. Lengler showed that, for the classical $(2+1)$-EA when it operates sufficiently close to the optimum, there is a threshold for HotTopic[1] $c_0 \approx 2.14$: for $c \leq c_0$ the algorithm has expected runtime $\Theta(n \log n)$, for $c > c_0$ the algorithm has expected exponential runtime. Here we show that the $(2+1)$-EA with duplicate avoidance has expected exponential runtime for all $c > 1.66$ on some HotTopic functions, and hence using this mechanism does not help. We have not been able to formally prove that there is a threshold as Lengler showed for the classical $(2+1)$-EA, however we will give an informal argument that suggests that there is a threshold $c_0 \in [1.64, 1.66]$ such that Algorithm 2 has expected runtime $\mathcal{O}(n \log n)$ for all $c < c_0$.

Note that throughout the chapter we choose the parameter ε such that the algorithm operates sufficiently close to the optimum. This means that, for all parameters $\alpha \in (0,1)$ and $c > 0$, there exists an ε_0 such that our statements hold for all $\varepsilon \leq \varepsilon_0$. We point out that HotTopic functions are constructed in such a way that the $(2+1)$-EA easily reaches (and stays in) a regime in which all search points have at most $C\varepsilon n$ zero-bits, where C is a constant that depends only on α and c. This justifies our intuitive formulation that

[1] Since we consider a situation close to the optimum, the parameter ε of the HotTopic function has to be small enough.

the parameter ε forces the algorithm to operate in an $\mathcal{O}(\varepsilon n)$ neighbourhood of the optimum. Although one might intuitively think that optimising a function far away from the optimum is easier (after all, there are more possible "good" mutations), we stress the fact that considering only the situation close to the optimum is a restriction: a surprising example that shows a situation where optimising a population far away from the optimum is more difficult than optimising a population close to the optimum is presented in [11]. This result is particularly relevant to this thesis because we consider the same fitness function HotTopic, but with a different parameter ε.

In order to study the impact of the mutation parameter on the runtime of Algorithm 2, we make the following plan. In Section 2.1 we define a *potential function* for some states of the algorithm and we prove a tail bound; in Section 2.2 we compute the drift of the algorithm with respect to the potential defined in Section 2.1. The drift is a function that depends on the parameters α, c and ε. For a fixed c' it holds that, if we find an α_0 such that we have a positive drift at $(c', \alpha_0, \varepsilon)$ with ε small enough, Algorithm 2 with mutation parameter c' has expected exponential runtime on all HotTopic functions with parameters α_0, ε and n large enough. In Section 2.3 we show that, for all $c > 1.66$, such an α_0 exists and in Section 2.4 we formally prove that this actually implies an expected exponential runtime. This would already be enough to state that implementing a duplicate avoidance mechanism on a $(2+1)$-EA is not helpful on some instances of HotTopic functions. However, we additionally give an informal argument that justifies our belief that, similarly as in the algorithms discussed by Lengler in [8], there is a threshold $c_0 \in [1.64, 1.66]$ such that Algorithm 2 has expected runtime $\Theta(n \log n)$ for all $c < c_0$.

2.1 Preliminaries

Drift analysis is a very useful tool in the analysis of EAs. Informally, we have that a drift towards the optimum on a certain EA implies that it has expected polynomial runtime, while a drift in the opposite direction implies that it has expected exponential runtime. There are multiple proofs of this fact in the literature (*e.g.* in [11] and in [10]), but for the sake of completeness we will give an argument for that in Section 2.4. The goal of this chapter is finding for which values of c the drift goes in the opposite direction of the optimum: in fact, by comparing this values with the threshold $c_0 \approx 2.14$ found by Lengler for the classical version of the algorithm, we can determine whether the duplicate avoidance mechanism is useful for the $(2+1)$-EA on HotTopic functions or not. Now that we have defined the goals of this chapter, we can start with some technical aspects. In general, the drift is always defined with respect to a potential function, and the choice of a "good" potential is crucial: defining a "bad" potential can make the necessary computations

prohibitively difficult or even lead us to situations where answering to our initial question becomes impossible.

Before we explain our choice of potential, we introduce the preliminary concept of *special population*. This is useful because we define the potential only in such populations. We say that a population with elements $x^{(1)}$ and $x^{(2)}$ is *special* if the two elements have the same bits in the hot topic of the current level and differ exactly in a single bit in the rest of the search point. In other words, if we compute the bit-wise XOR between the bits outside the hot topic of the current level of two elements of a special population, we obtain a string with exactly a single one-bit. We remind that we define the potential only in states that represent a special population. Concretely, we say that the potential X_i is the number of one-bits in the fittest element of the i-th special population. For convenience we will note with $\mathrm{OM}(x)$ the number of one bits in search point x, so we have $X_i = \mathrm{OM}(\tilde{x})$, where $\tilde{x}$ is the fittest element in the i-th special population. Moreover, we denote by $d(I, x) := |\{i \in I \mid x_i = 0\}|/|I|$ the density of zero bits in I. In particular, $d([n], x) = 1 - \textsc{Om}(x)/n$. The drift of the algorithm with respect to this potential is formally defined as

$$\mathbb{E}\left[X_i - X_{i+1}\right].$$

Before we begin our investigation on what mutation parameters make the drift positive/ negative (the goal of this chapter), we prove a tail bound lemma. This lemma will be useful to prove a lemma in the next section (Lemma 2.2) and to prove the runtime of the algorithm.

Lemma 2.1 *For every $\varepsilon > 0$ there exists a constant $C > 0$ such that the following holds. Let x and y be two search points with $a := \mathrm{OM}(x) \geq \mathrm{OM}(y) =: b$ with $a \geq C\varepsilon n$ and $b \geq \varepsilon n$. Let X be the number of one-bits in the first special population that we reach starting from the population $\{x, y\}$. Then $Pr[X > a + d] \leq e^{-Cd}$ and $Pr[X < b - d] \leq e^{-Cd}$ for all $d \geq 0$.*

Proof Starting from the population $\{x, y\}$, we define a random variable T as the number of iterations until the next special population. The only exception to this definition is that, whenever we get to a population with two elements with the same fitness, we consider only iterations that flip at least one zero-bit. We want to compute an upper bound for $Pr[T > t]$. In order to do that, we compute the probability p_1 of getting to a special population in the next iteration when we consider a non-special population $\{x, y\}$ with $f(x) > f(y)$ and the probability p_2 of reaching a special population in the next iteration when we have two elements of the same fitness in the population (this time we condition the probability of flipping at least one zero-bit).

In order to compute p_1 we observe that from a population with $f(x) > f(y)$ we reach a special population in the next iteration if

1. we mutate a single one-bit outside the hot topic,
2. no other bit in x is flipped
3. and the offspring is accepted in the new population.

If n is sufficiently large and ε is chosen small enough (*e.g.* $\varepsilon < \frac{1}{3}$), a lower bound for this probability is

$$p_1 \geq \frac{1}{4}(1-\alpha)(1-\varepsilon)c(1-\frac{c}{n})^{n-1} \geq \frac{1}{8}(1-\alpha)ce^{-c}$$

In order to compute p_2 we observe that we reach a new special population in a single iteration starting from a population $\{x, y\}$ with $f(x) = f(y)$ if

1. the only bit we flip is a zero-bit outside the hot topic either in x or in y
2. and we keep the parent and the offspring in the new population.

If n is large enough, a lower bound for p_2 is given by

$$p_2 \geq \frac{1}{2}(1-\alpha)c(1-\frac{c}{n})^{n-1} \geq \frac{1}{4}(1-\alpha)ce^{-c}$$

We observe that $p_1, p_2 \in \Omega(1)$ and that they both do not depend neither on n nor on ε. Now we are ready to compute an upper bound for $Pr[T > t]$. We define $p := \min\{p_1, p_2\}$. We have shown that $Pr[T = t \mid T > t-1] \geq p$. We can apply an inductive argument to this result and we get that $Pr[T > t] \leq (1-p)^t \leq e^{-pt}$.

Now we define a new random variable Y that denotes the total number of bits flipped in any of the iterations that count towards T. Since $|a - X| \leq Y$, this random variable will be useful to prove the statement of the lemma. We first compute $p_t := Pr[Y > (2c+1)t \mid T = t]$. For each of the t iterations we distinguish two cases:

1. The two individuals have different fitness. Since in this case we don't condition on flipping at least one zero-bit, each of the n bit has probability $\frac{c}{n}$ to be flipped, independent of each other.
2. The two individuals have the same fitness. In this case we condition on at least one zero-bit being flipped. For the one-bits, the situation is the same as in the previous case: each of them still has probability $\frac{c}{n}$ to be flipped, independent of each other. For the zero-bits, we enumerate them from 1 to k, and we define i as the index of the first zero-bit that is flipped. Then none of the zero bits with index $1, \ldots, i-1$ is flipped, and each of the zero-bits with index $i+1, \ldots, k$ is flipped independently with probability $\frac{c}{n}$. In total, we have an upper bound on the

number of flipped bits in this iteration by $1 + B$, where $B \sim Bin(n, \frac{c}{n})$. This upper bound naturally holds also for the first case.

Hence, for each iteration i, the random variable $Y_i = 1 + B$ is an upper bound on the number of flipped bits. Since $Y = \sum_{i=1}^{t} Y_i$, we have that $Y = t + B'$, where $B' \sim Bin(t \cdot n, \frac{c}{n})$. We now have

$$p_t = Pr\left[Y > (2c+1)t \mid T = t\right] \leq Pr\left[B' > 2ct \mid T = t\right] \leq e^{-\frac{1}{3}ct}$$

where in the last step we used the Chernoff bound. And, with the same argument, for $Pr\left[B' > 2ct \mid T = t'\right]$ and every $t' \leq t$ we get the same upper bound. Now we have everything we need to bound $Pr\left[Y > 2ct\right]$. We get

$$\begin{aligned} Pr\left[Y > 2ct\right] &= Pr\left[T > t \text{ and } Y > 2ct\right] + Pr\left[T \leq t \text{ and } Y > 2ct\right] \\ &\leq Pr\left[T > t\right] + Pr\left[Y > 2ct \mid T \leq t\right] \cdot Pr\left[T \leq t\right] \\ &\leq Pr\left[T > t\right] + Pr\left[Y > 2ct \mid T \leq t\right] \\ &= e^{-\Omega(t)} \end{aligned}$$

By substituting t with $\frac{k}{2c}$ we get $Pr\left[Y > k\right] = e^{-\Omega(k)}$. Since $|a - X| \leq Y$ by definition of Y, we get

$$Pr\left[X < b - d\right] = Pr\left[-(b - X) < -d\right] = Pr\left[b - X > d\right] \leq Pr\left[a - X > d\right] = e^{-\Omega(d)}$$

and

$$Pr\left[X > a + d\right] = Pr\left[-(a - X) > d\right] \leq Pr\left[|a - X| > d\right] = e^{-\Omega(d)}$$

which proves the lemma. □

2.2 Drift of X_i

Now we are finally ready to compute $\mathbb{E}\left[X_i - X_{i+1}\right]$. Since some computations will come out often in the computation of the drift, it is useful to prove two *ad hoc* lemmas such that we can introduce some modularity in the proof by just referring to them later. Lemma 2.2 computes an expected value analogous to the drift (*i.e.* a kind of drift that extends also to non-special populations) in the case that one of the two search points in the population dominates over the other one; Lemma 2.3 does the same in the case that the population contains two search points with the same fitness. These lemmas will be very helpful to compute the drift in a more compact and elegant way.

Lemma 2.2 *Consider a non-special population $\{x^{(1)}, x^{(2)}\}$ with $f(x^{(1)}) > f(x^{(2)})$. We define $a :=$OM $(x^{(1)})$ and X as the random variable denoting* OM (x)*, where x is the fittest element of the next special population that will be visited by Algorithm 2. Then*

$$\mathbb{E}\left[|a - X|\right] \in \mathcal{O}(\varepsilon)$$

Proof In order to prove the statement of Lemma 2.2 we start from the transition state diagram in Figure 2.1. Starting from the non-special population described by the lemma, Algorithm 2 has an infinite number of paths that lead to a special population. We group them in two categories.

1. Leaving the initial state with population $\{x^{(1)}, x^{(2)}\}$ and getting (possibly after an arbitrary number of loops in the initial state) to the special population $\{x^{(1)}, x^{(3)}\}$. This transition happens if Algorithm 2 mutates $x^{(1)}$ and flips a single one-bit outside the corrent hot topic. The probability p_1 of taking this transition is constant: in fact we have to choose $x^{(1)}$ from the initial population, flipping a single of its one bits outside the current hot topic and not changing the other bits. A lower bound for this probability is given by

$$\frac{1}{2}(1-\alpha)(1-\varepsilon)c(1-\frac{c}{n})^{n-1} \in \Omega(1)$$

   In this case, the value of $a - X = 0$, since the fittest element of the special population is $x^{(1)}$.

2. Leaving the initial state with population $\{x^{(1)}, x^{(2)}\}$ to get (either directly or through other states), to a special population where the fittest element is $x_i^{(4)}$ with $f(x_i^{(4)}) > f(x^{(1)})$. In general, there is a large number of those paths. Note that in general each of this path may have a different probability. In order to take one of those paths, we need to flip at least one zero-bit. This naturally follows from the fact that the fittest element of the final special population is fitter than the fittest element of the initial population.

Since the first case contributes zero to the value of $a - X$, we only consider the second one, which has an infinite number of paths that bring to a special population with $a - X \neq 0$. We denote with p the probability of looping in the initial state, with p'_i the probability of taking the i-th path conditioned on leaving the initial state and with Δ_i its corresponding value of $|a - X|$. In order to show the lemma, we have to prove that

$$\sum_{i=1}^{\infty}(\sum_{j=1}^{\infty} p^j p'_i \Delta_i) = \sum_{i=1}^{\infty} \frac{p'_i}{1-p}\Delta_i \in \mathcal{O}(\varepsilon) \tag{2.1}$$

In order to prove this, we first split p'_i in two parts: a first step with probability q_i that indicates the probability of taking the first (and possibly unique) step towards the special population, again conditioned on leaving the initial state. In other words, we want to single out the first iteration which flips a zero-bit. After this first step, the population will be $\{x, x^{(1)}\}$. We

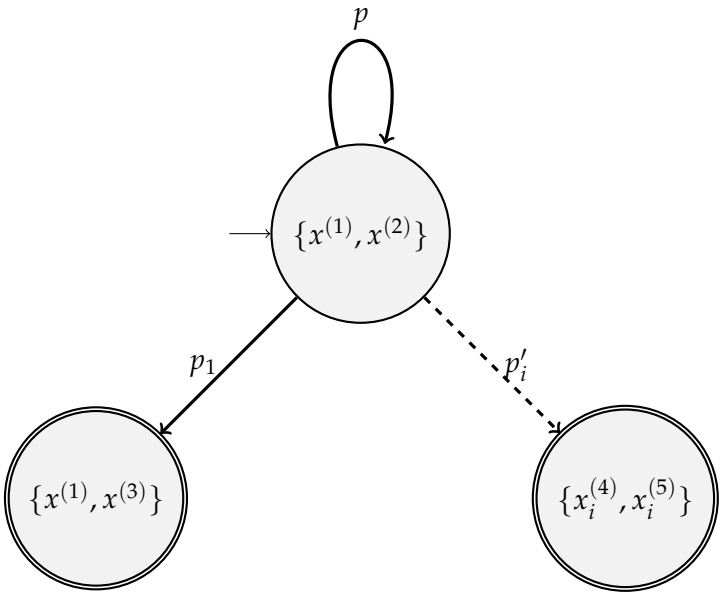

Figure 2.1: Transition state diagram starting from a population $\{x^{(1)}, x^{(2)}\}$ such that $f(x^{(1)}) > f(x^{(2)})$. The final states represent special populations. The special population on the left represents the case where the fittest element of the population is $x^{(1)}$. The special population on the right represents the i-th possible special population with fittest element $x_i^{(4)}$ such that $f(x_i^{(4)}) > f(x^{(1)})$. We observe that nothing prevents $x_i^{(5)}$ being equal to $x^{(1)}$ for some i. The arrow with probability p_i' is dashed because, in some cases, the path to -special situations before arriving to the final special population.

define $K_i := \mathrm{OM}(x^{(1)}) - \max\{\mathrm{OM}(x^{(1)}, \mathrm{OM}(x)\}$. Then we consider also the expected contribution of $G_i := \max\{\mathrm{OM}(x^{(1)}, \mathrm{OM}(x)\} - \mathrm{OM}(x_i^{(4)})$ to the value of $a - X$. Observe that $\Delta_i = |K_i + \mathbb{E}(G_i)|$. In order to show Equation 2.1 we can just show

$$\sum_{i=1}^{\infty} \frac{p_i'}{1-p} \Delta_i = \frac{1}{1-p} \sum_{i=1}^{\infty} q_i |(K_i + \mathbb{E}(G_i))| \in \mathcal{O}(\varepsilon)$$

In order to approximate the expectation of G_i, we apply Lemma 2.1 (recall that we in are in a situation arbitrarily close to the optimum and hence the conditions of the lemma hold) . Since $\sum_{k=0}^{\infty} e^{-Ck} k \in \mathcal{O}(1)$ (this can be seen in different ways, *e.g.* using $k \leq e^{\frac{C}{2}k}$ for k large enough and the geometric series formula), we get that the expected value of G_i is $|C'|$, where $C' \in \mathcal{O}(1)$. Hence, the initial goal of Equation 2.1 reduces to

$$\sum_{i=1}^{\infty} \frac{q_i}{1-p} |(K_i + C')| \in \mathcal{O}(\varepsilon)$$

We now rewrite the sum by substituting the infinite number of paths that have a first step of probability q_i with paths that have a first step of probability $p_{j,k}$. $p_{j,k}$ indicates the probability of flipping j zero-bits and k one bits. The probability of flipping j zero bits is $\mathcal{O}(\varepsilon^j)$. An upper bound to the probability of flipping k one bits can be approximated with a Chernoff bound. We have

$$Pr\left[Bin(n, \frac{c}{n}) \geq x\right] \leq e^{-\frac{(x-c)^2}{3c}}$$

Moreover, we can merge the constants $1-p$ and the leading constant of $\mathcal{O}(\varepsilon^j)$ into a single constant C'''. By applying this result and the approximation with the Chernoff bound, we have that Equation 2.1 reduces to

$$\sum_{j=1}^{\infty}\sum_{k=0}^{\infty} C''\varepsilon^j e^{-\frac{(k-c)^2}{3c}}|K_{j,k}+C'| \in \mathcal{O}(\varepsilon)$$

By introducing a natural upper bound of $j+k$ for $K_{j,k}$ and observing that $|j+k+C'| \leq j+k+|C'|$ we get

$$\sum_{j=1}^{\infty}\sum_{k=0}^{\infty} C''\varepsilon^j e^{\frac{-(k-c)^2}{3c}}(j+k+|C'|) \leq C^{(3)}\varepsilon \tag{2.2}$$

At this point it is useful to introduce an upper bound for $e^{-\frac{(k-c)^2}{3c}}$. For sufficiently large k we have

$$\begin{aligned} e^{-\frac{(k-c)^2}{3c}} &= e^{\frac{-k^2+2kc-c^2}{3c}} \\ &\leq e^{\frac{-k^2+\frac{k^2}{2}}{3c}} \\ &\leq \tilde{C}e^{-\frac{k^2}{6c}} \end{aligned}$$

We hence have an upper bound (by incorporating $\tilde{C}$ in the previous undetermined constant C''') for equation 2.2 of

$$\sum_{j=1}^{\infty}\sum_{k=0}^{\infty} C''\varepsilon^j e^{-\frac{k^2}{6c}}(j+k+|C'|)$$

It is clear that $\sum_{k=0}^{\infty} e^{-\frac{k^2}{6c}}$ converges to a constant: we can reduce it to a form where we can apply the geometric sum formula. Similarly, since $k \leq \frac{k^2}{4}$ for k large enough, we have that also $\sum_{k=0}^{\infty} e^{-\frac{k^2}{6c}} k$ converges to a constant. By appropriately taking into accounts these new constants in a new constant $\bar{C}$, we have

$$\sum_{j=1}^{\infty} \bar{C} \varepsilon^j j \in \mathcal{O}(\varepsilon)$$

We have shown that Equation 2.1 is in $\mathcal{O}(\varepsilon)$, which proves the Lemma. □

Lemma 2.3 *Consider a non-special population $\{x^{(1)}, x^{(2)}\}$ with $f(x^{(1)}) = f(x^{(2)})$. We define a :=*OM $(x^{(1)})$ *such that $a \geq (1-\varepsilon)n$ and X as the random variable denoting* OM (x)*, where x is the fittest element of the next special population that will be visited by Algorithm 2. Then*

$$\mathbb{E}[a - X] = \frac{1}{\alpha(e^{(1-\alpha)c} - 1) + 1} \left[\alpha c(1-\alpha)e^{(1-\alpha)c} - \alpha(e^{(1-\alpha)c} - 1) - 1\right] + \mathcal{O}(\varepsilon + \frac{\varepsilon}{n^{\frac{1}{6}} - \frac{1}{n}})$$

Proof In order to prove Lemma 2.3 we compute both an upper and a lower bound for the expectation of $a - X$. We will show that those expectations are equal up to minor terms. In order to do that we consider the possible paths in the transition state diagram, starting from the non-special population described by the lemma. Algorithm 2 has an infinite number of paths that lead to a special population. We group them in three categories.

1. Flipping a single zero-bit in either $x^{(1)}$ or in $x^{(2)}$. When studying such a transition, we have also to consider all loops in the initial state.
2. Flipping a single zero-bit in the hot topic of either $x^{(1)}$ or $x^{(2)}$ and $t > 1$ one-bits outside of the hot topic. Similarly to case 1, also here we have to consider all loops in the initial state.
3. Other paths that flip more than a single zero-bit. Similarly to how we did in the proof of Lemma 2.2, we can show that the sum of the probability of those paths, conditioned on leaving the initial state, is in the order of $\mathcal{O}(\varepsilon^2)$. Briefly, each path flips $j \geq 2$ zero-bits and k one-bits: by using a Chernoff bound to approximate the probability of flipping k bits and reducing it to a multiplicative constant we have the result. We will argue later on that the sum of those paths has a contribution of order $\mathcal{O}(\varepsilon)$ to the drift.

We compute the contributions to $a - X$ in each case. The lemma will follow by summing the three contributions. Before diving into the analysis of the single cases, we compute some terms that will be useful in the analysis.

- Let $p_1(t)$ be the probability of flipping a single zero-bit in the hot topic and $t > 1$ one-bits outside of it. Recall that α is the proportion of elements in the hot topic of the string and ε the density of zero-bits. We get

$$p_1(t) = \alpha n \varepsilon \frac{c}{n} \binom{n(1-\alpha)(1-\varepsilon)}{t} \left(\frac{c}{n}\right)^t (1-\frac{c}{n})^{n-t-1}$$

Now we give both a lower and an upper bound for $p_1(t)$. In order to do that we compute the bounds for the binomial coefficient and we give an approximation for $(1-\frac{c}{n})^{n-t-1}$. For the upper bound of the binomial coefficient we have

$$\begin{aligned}
\binom{n(1-\alpha)(1-\varepsilon)}{t} &= \frac{\prod_{i=0}^{t-1} n(1-\alpha)(1-\varepsilon) - i}{t!} \\
&\leq \frac{n^t (1-\alpha)^t (1-\varepsilon)^t}{t!} \\
&\leq \frac{n^t (1-\alpha)^t}{t!}
\end{aligned}$$

For the lower bound we restrict ourselves to the cases where $t \leq \min\{n^{\frac{1}{3}}, \varepsilon^{-\frac{1}{2}}\}$. We have

$$\begin{aligned}
\binom{n(1-\alpha)(1-\varepsilon)}{t} &= \frac{\prod_{i=0}^{t-1} n(1-\alpha)(1-\varepsilon) - i}{t!} \\
&\geq \frac{(n(1-\alpha)(1-\varepsilon) - t)^t}{t!} \\
&\geq \frac{\left(n(1-\alpha)(1-\varepsilon) - n^{\frac{1}{3}}\right)^t}{t!} \\
&\geq \frac{\left(n(1-\alpha)(1-\varepsilon)(1-n^{-\frac{1}{2}})\right)^t}{t!} \\
&\geq \frac{n^t (1-\alpha)^t (1-\varepsilon)^{\varepsilon^{-\frac{1}{2}}} (1-n^{-\frac{1}{2}})^{n^{\frac{1}{3}}}}{t!} \\
&\geq \frac{n^t (1-\alpha)^t e^{-2\varepsilon^{\frac{1}{2}}} e^{-2n^{-\frac{1}{6}}}}{t!} \\
&\geq \frac{n^t (1-\alpha)^t (1-2\varepsilon^{\frac{1}{2}})(1-2n^{-\frac{1}{6}})}{t!} \\
&= \frac{n^t (1-\alpha)^t (1-\mathcal{O}(\sqrt{\varepsilon} + n^{-\frac{1}{6}}))}{t!}
\end{aligned}$$

Where we used $e^{-2x} \leq 1 - x \leq e^{-x}$, and the first inequality holds for x sufficiently small. Moreover we approximate $(1-\frac{c}{n})^{n-t-1}$ with

$e^{-c}(1-\mathcal{O}(\frac{t}{n}))$. In order to see why the last approximation holds, we use the Taylor expansions $e^{-x} = 1 - x \pm \mathcal{O}(x^2)$ for $x \in [-1,1]$ and $1 - x = e^{-x \pm \mathcal{O}(x^2)}$ for $x \in [-0.5, 0.5]$. Then, for n large enough, we get

$$\begin{aligned}\left(1-\frac{c}{n}\right)^{n-t-1} &= \left(e^{-\frac{c}{n} \pm \mathcal{O}(\frac{c^2}{n^2})}\right)^{n-t-1} \\ &= e^{-c+\frac{ct}{n}+\frac{c}{n} \pm \mathcal{O}(\frac{t}{n})} \\ &= e^{-c}(1 - \frac{ct}{n} - \frac{c}{n} \pm \mathcal{O}(\frac{t}{n}) \pm \mathcal{O}(\frac{t^2}{n^2})) \\ &= e^{-c}(1-\mathcal{O}(\frac{t}{n}))\end{aligned}$$

Hence an upper bound for $p_1(t)$ is $\alpha\varepsilon c\frac{(1-\alpha)^t}{t!}c^t e^{-c}(1-\mathcal{O}(\frac{t}{n}))$, while a lower bound is $\alpha\varepsilon c\frac{(1-\alpha)^t}{t!}c^t e^{-c}(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}})$.

- We can use the previous upper and lower bounds to get the sum of the probabilities of $p_1(t)$ over the range of possible t. We get an upper bound of

$$\begin{aligned}\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t) &\leq \sum_{t=1}^{\infty} p_1(t) \\ &= \sum_{t=1}^{\infty} \alpha\varepsilon c \binom{n(1-\alpha)(1-\varepsilon)}{t} c^t (1-\frac{c}{n})^{n-t-1} \\ &\leq \sum_{t=1}^{\infty} \alpha\varepsilon c \frac{(1-\alpha)^t}{t!} c^t e^{-c}\left(1-\mathcal{O}(\frac{t}{n})\right) \\ &= \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) - \alpha\varepsilon e^{-c}\frac{c}{n}\sum_{t=1}^{\infty}\frac{(1-\alpha)^t c^t}{(t-1)!} \\ &= \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) - \mathcal{O}(\frac{\varepsilon}{n})\end{aligned}$$

And a lower bound of

$$\begin{aligned}\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t) &= \sum_{t=1}^{\infty} p_1(t) - \sum_{t=n(1-\alpha)(1-\varepsilon)}^{\infty} p_1(t) \\ &\geq \sum_{t=1}^{\infty} \alpha\varepsilon c \frac{(1-\alpha)^t}{t!} c^t e^{-c}(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}})) - \sum_{n(1-\alpha)(1-\varepsilon)}^{\infty}\frac{c^t}{t!} \\ &\geq \alpha\varepsilon c e^{-c}\sum_{t=1}^{\infty}\frac{(1-\alpha)^t c^t}{t!}(1-\mathcal{O}(\sqrt{\varepsilon}+n^{-\frac{1}{6}})) - \frac{\alpha\varepsilon c e^{-c}}{n}\sum_{t=1}^{\infty}\frac{(1-\alpha)^t c^t}{(t-1)!} - \frac{1}{n^K} \\ &= \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) - \mathcal{O}(\varepsilon^{\frac{3}{2}} + \frac{\varepsilon}{n^{\frac{1}{6}}})\end{aligned}$$

where we used that $\frac{c^t}{t!}$ is an upper bound for $p_1(t)$ and that $\sum_{t=n(1-\alpha)(1-\varepsilon)}^{\infty} \frac{c^t}{t!} \leq n^{-K}$ for every fixed positive constant K (the latter upper bound can be seen with the Stirling bound).

- Let p_2 be the probability of flipping a single zero-bit and no other bits in the string. Since ε is the density of zero-bits in the elements of the starting population, we get

$$p_2 = \varepsilon n \frac{c}{n}(1-\frac{c}{n})^{n-1} = \varepsilon c e^{-c}(1-\mathcal{O}(\frac{1}{n}))$$

- Let p be the probability of staying in the initial state after an iteration of the algorithm. We now compute $\frac{1}{1-p}$ (again, in form of a lower and an upper bound). This term is useful to consider the loops in the initial state. Since $1-p = \sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t) + p_2 + \mathcal{O}(\varepsilon^2)$, we get a lower bound of

$$\begin{aligned}\frac{1}{1-p} &= \frac{1}{\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t) + p_2 + \mathcal{O}(\varepsilon^2)} \\ &\geq \frac{1}{\alpha\varepsilon c e^{-c}(e^{(1-\alpha)c}-1) - \varepsilon c e^{-c} - \mathcal{O}(\frac{\varepsilon}{n})} \\ &\geq \frac{1}{\alpha\varepsilon c e^{-c}(e^{(1-\alpha)c}-1) + \varepsilon c e^{-c}} + \mathcal{O}(\frac{\varepsilon}{n})\end{aligned}$$

and an upper bound of

$$\begin{aligned}\frac{1}{1-p} &= \frac{1}{\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t) + p_2 + \mathcal{O}(\varepsilon^2)} \\ &\leq \frac{1}{\alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) + \varepsilon c e^{-c} - \mathcal{O}(\varepsilon^{\frac{3}{2}} + \frac{\varepsilon}{n^{\frac{1}{6}}})} \\ &= \frac{1}{\alpha\varepsilon c e^{-c}(e^{(1-\alpha)c}-1) + \varepsilon c e^{-c}} + \mathcal{O}(\varepsilon^{\frac{3}{2}} + \frac{\varepsilon}{n^{\frac{1}{6}}})\end{aligned}$$

Now we compute the drifts in the three cases stated at the beginning of the proof.

1. We have some additional possibilities: either we mutate $x^{(1)}$ and we throw away $x^{(2)}$ or we mutate $x^{(1)}$ and we throw away $x^{(1)}$ (analogously by exchanging $x^{(1)}$ and $x^{(2)}$). In the first case we directly get to a special population, in the latter we get to the situation of Lemma 2.2. For this reason, the total drift here is $-1+\mathcal{O}(\varepsilon)$. Hence the drift is given by

$$\sum_{i=0}^{\infty} p^i p_2(-1+\mathcal{O}(\varepsilon)) = \frac{\varepsilon c e^{-c}(1+\mathcal{O}(\frac{1}{n}))}{1-p}(-1+\mathcal{O}(\varepsilon)) = -\frac{\varepsilon c e^{-c}}{1-p} + \mathcal{O}(\varepsilon + \frac{1}{n})$$

2. After the first mutation we get to a state where we can apply Lemma 2.2. Given this construction, we have a drift of $t-1$. By computing the probability of such a path and multiplying by the drift we get

$$\begin{aligned}&\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}\sum_{i=0}^{\infty}p^i p_1(t)(t-1)\\&=\frac{1}{1-p}\left(\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}p_1(t)t-\sum_{t=1}^{\infty}p_1(t)\right)\end{aligned}$$

From the previous equation we observe that we need to compute bounds for $\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} p_1(t)t$. We get a lower bound of

$$\begin{aligned}\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}p_1(t)t&=\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}\alpha\varepsilon ce^{-c}\frac{(1-\alpha)^t}{(t-1)!}c^t\left(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}})\right)\\&=\alpha\varepsilon c^2e^{-c}(1-\alpha)\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}\frac{(1-\alpha)^{t-1}}{(t-1)!}c^{t-1}\left(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}})\right)\\&\geq\alpha\varepsilon c^2e^{-c}(1-\alpha)e^{(1-\alpha)c}-\mathcal{O}(\varepsilon^{\frac{3}{2}}+\frac{\varepsilon}{n^{\frac{1}{6}}})\end{aligned}$$

and an upper bound of

$$\begin{aligned}\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}p_1(t)t&\leq\sum_{t=1}^{\infty}\alpha\varepsilon ce^{-c}\frac{(1-\alpha)^t}{(t-1)!}c^t\left(1-\mathcal{O}(\frac{t}{n})\right)\\&=\alpha\varepsilon c^2e^{-c}(1-\alpha)e^{(1-\alpha)c}-\mathcal{O}(\frac{\varepsilon}{n})\end{aligned}$$

We now sum all three cases and we get that the value of $\mathbb{E}[a-X]$ is

$$\begin{aligned}&\frac{1}{1-p}\left(\sum_{t=1}^{\infty}p_1(t)t-\sum_{t=1}^{\infty}p_1(t)+\varepsilon ce^{-c}(1+\mathcal{O}(\frac{1}{n}))(-1+\mathcal{O}(\varepsilon))\right)\\&=\frac{1}{1-p}\left(\sum_{t=1}^{\infty}p_1(t)t-\sum_{t=1}^{\infty}p_1(t)+\varepsilon ce^{-c}(-1+\mathcal{O}(\varepsilon+\frac{1}{n}))\right)\end{aligned}$$

By introducing the terms that we have previously computed we get that lower and upper bound coincide and are equal to

$$\frac{\alpha c(1-\alpha)e^{(1-\alpha)c} - \alpha(e^{c(1-\alpha)}-1) - 1}{\alpha(e^{(1-\alpha)c}-1)+1} + \mathcal{O}(\varepsilon + \frac{\varepsilon}{n^{\frac{1}{6}}} - \frac{1}{n})$$

which proves the lemma. □

Before we continue our journey and actually compute the drift, we make an important observation about the result of the lemma we have just proven. Although this new lemma will be useful for the first time only in Section 2.3, we prefer to introduce it here because it describes a property of the result we just proved in Lemma 2.3.

Lemma 2.4 *Fix an arbitrary $\alpha_0 \in [0,1]$. Then, for all $c > 0$, the function*

$$L(c) = \frac{1}{\alpha_0(e^{(1-\alpha_0)c}-1)+1}\left[\alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} - \alpha_0(e^{(1-\alpha_0)c}-1) - 1\right]$$

is monotonic increasing.

Proof We simply compute the derivative of $L(c)$ with respect to c and we show that it is non-negative for any choice of $\alpha_0 \in [0,1]$. The derivative is given by

$$\begin{aligned} L'(c) = \frac{1}{\left(\alpha_0(e^{(1-\alpha_0)c}-1)+1\right)^2}[&(\alpha_0(1-\alpha_0)e^{(1-\alpha_0)c} + \alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c} \\ &- \alpha_0(1-\alpha_0)e^{(1-\alpha_0)c})(\alpha_0(e^{(1-\alpha_0)c}-1)+1) - (\alpha_0(1-\alpha_0)e^{(1-\alpha_0)c}) \\ &(\alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} - \alpha_0(e^{(1-\alpha_0)c}-1) - 1)] \end{aligned}$$

By removing the denominator (Which it is a square and hence non-negative) and simplifying the expression above we get

$$\begin{aligned} &\alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c}(\alpha_0(e^{(1-\alpha_0)c}-1)+1) - \alpha_0^2 c(1-\alpha_0)^2 e^{2(1-\alpha_0)c} \\ &+ \underbrace{\alpha_0^2(1-\alpha_0)(e^{(1-\alpha_0)c}-1)e^{(1-\alpha_0)c} + \alpha_0(1-\alpha_0)e^{(1-\alpha_0)c}}_{\geq 0} \\ &\geq \alpha_0^2 c(1-\alpha_0)^2 e^{2(1-\alpha_0)c} + \alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c}(1-\alpha_0) \\ &- \alpha_0^2 c(1-\alpha_0)^2 e^{2(1-\alpha_0)c} \\ &= \alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c}(1-\alpha_0) \geq 0 \end{aligned}$$

and hence $L(c)$ is monotonic increasing. □

After this intermezzo, we go on with the plan we proposed at the beginning of the chapter and we compute the drift of Algorithm 2 with respect to α, c and ε. In order to do that, the statements of Lemma 2.2 and Lemma 2.3 will be very useful.

Lemma 2.5 *Let*

$$L(\alpha,c,\varepsilon) := \frac{1}{\alpha(e^{(1-\alpha)c}-1)+1}\left[\alpha c(1-\alpha)e^{(1-\alpha)c} - \alpha(e^{(1-\alpha)c}-1) - 1\right] + \mathcal{O}(\varepsilon)$$

$$\begin{aligned} D_1(\alpha,c,\varepsilon) :=& -\varepsilon c e^{-c} + (1-\alpha)^2\varepsilon c^2 e^{-c} L(\alpha,c,\varepsilon) + \alpha\varepsilon c^2(1-\alpha)e^{-\alpha c} \\ & - \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) + \alpha^2\varepsilon c^2 e^{-c} L(\alpha,c,\varepsilon) \pm \mathcal{O}(\varepsilon^2) \end{aligned}$$

$$D_2(\alpha,c,\varepsilon) := (1-\alpha)\varepsilon c e^{-c} L(\alpha,c,\varepsilon) + \alpha\varepsilon c^2(1-\alpha)e^{-\alpha c} \pm \mathcal{O}(\varepsilon^2)$$

$$\begin{aligned} p(\alpha,c,\varepsilon) :=& 1 - \frac{1}{2}\Big(\varepsilon c e^{-c} - (1-\alpha)^2\varepsilon(1-\varepsilon)c^2 e^{-c} - \alpha\varepsilon c(e^{(1-\alpha)c}-1) \\ & - \alpha^2\varepsilon(1-\varepsilon)c^2 e^{-c} - (1-\alpha)\varepsilon c e^{-c} - \alpha\varepsilon c e^{-\alpha c}\Big) \end{aligned}$$

Starting from a special population with elements x and y with $f(x) > f(y)$, the drift of Algorithm 2 is

$$D(\alpha,c,\varepsilon) = \frac{D_1(\alpha,c,\varepsilon) + D_2(\alpha,c,\varepsilon)}{2(1-p(\alpha,c,\varepsilon))}$$

Proof In order to compute the drift $D(\alpha,c,\varepsilon)$, we sum all contributions of the difference of potential brought by the different paths to a next special population multiplied by their probability. In general, we have that the contributions to the drift of paths that leave the initial state by flipping more than a single zero-bit are negligible. This holds, using a similar argument to the one of Lemma 2.2, because we assume that our algorithm operates sufficiently close to the optimum. We compute both an upper and a lower bound for the drift.

It is useful two distinguish two main cases. The first case, summarised in Table 2.1, considers the case that Algorithm 2 mutates x: we denote with $D_1(\alpha,c,\varepsilon)$ the drift conditioned on leaving the initial state under this assumption. The second case, summarised in Table 2.2, considers the case where the algorithm mutates y, and we denote with $D_2(\alpha,c,\varepsilon)$ the drift (again without considering loops on the initial state) under this assumption. We denote with $p(\alpha,c,\varepsilon)$ the probability of staying in the initial situation. Since the two cases come with a factor of 0.5 in the final expression of the drift, we have

$$\mathbb{E}\left[X_i - X_{i+1}\right] = \sum_{i=0}^{\infty}\frac{p^i}{2}(D_1(\alpha,c,\varepsilon) + D_2(\alpha,c,\varepsilon)) = \frac{1}{2(1-p(\alpha,c,\varepsilon))}(D_1(\alpha,c,\varepsilon) + D_2(\alpha,c,\varepsilon))$$

We start by computing an upper bound and a lower bound for $D_1(\alpha,c,\varepsilon)$. Table 2.1 shows all possible cases. The relevant consequences to the drift are the following

1. We get to a special population, where the generated offspring x_1 is the fittest element. $X_i - X_{i+1} = -1$ and the probability of getting to this state is

$$(1-\alpha)\varepsilon n\frac{c}{n}(1-\frac{c}{n})^{n-1} = (1-\alpha)\varepsilon c e^{-c}(1-\mathcal{O}(\frac{1}{n}))$$

2. We get to a population with $\{x, x_1\}$ with $f(x) = f(x_1)$. This state satisfies the conditions of Lemma 2.3. We observe that $L(\alpha, c, \varepsilon)$ is equal to $\mathbb{E}[\mathrm{OM}(x) - X_{i+1}]$ that we computed in Lemma 2.3. The drift of this case is hence given by

$$(1-\alpha)^2\varepsilon(1-\varepsilon)n^2\frac{c^2}{n^2}(1-\frac{c}{n})^{n-2}L(\alpha,c,\varepsilon) = (1-\alpha)^2\varepsilon(1-\varepsilon)c^2e^{-c}L(\alpha,c,\varepsilon)(1-\mathcal{O}(\frac{1}{n}))$$

3. Similarly to case 1, we get directly to a special population and the difference of the potential is -1. The probability is given by

$$\alpha\varepsilon n\frac{c}{n}(1-\frac{c}{n})^{n-1} = \alpha\varepsilon c e^{-c}(1-\mathcal{O}(\frac{1}{n}))$$

4. We create offspring x_1 fitter than x. We can apply Lemma 2.2 and we get that, with high probability, x_1 will be the fittest element of the next initial population. The value of $X_i - X_{i+1}$ is $t-1$. We sum the contributions for all possible values of t and we get

$$\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)}(t-1)\alpha\varepsilon c\binom{n(1-\alpha)(1-\varepsilon)}{t}\left(\frac{c}{n}\right)^t(1-\frac{c}{n})^{n-t-1}$$

And, by applying the same computations of the proof of Lemma 2.3, we get an upper bound of

$$(\alpha\varepsilon c^2(1-\alpha)e^{-c}e^{(1-\alpha)c} - \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1))(1-\mathcal{O}(\frac{t}{n}))$$

and a lower bound of

$$(\alpha\varepsilon c^2(1-\alpha)e^{-c}e^{c(1-\alpha)} - \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1))(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}}))$$

5. We create an offspring with the same fitness of x, hence we apply the statement of Lemma 2.3. The drift of this case is given by $\alpha^2\varepsilon(1-\varepsilon)c^2e^{-c}L(\alpha,c,\varepsilon)$.

Hence, by summing the five contributions and the $\pm\mathcal{O}(\varepsilon^2)$ error terms, we get that the contribution to the drift of the first case, for n large enough is

$$\begin{aligned}D_1(\alpha,c,\varepsilon) = &-\varepsilon c e^{-c} + (1-\alpha)^2\varepsilon(1-\varepsilon)c^2e^{-c}L(\alpha,c,\varepsilon) + \alpha\varepsilon c^2(1-\alpha)e^{-\alpha c}\\ &- \alpha\varepsilon c e^{-c}(e^{c(1-\alpha)}-1) + \alpha^2\varepsilon(1-\varepsilon)c^2e^{-c}L(\alpha,c,\varepsilon) \pm \mathcal{O}(\varepsilon^2)\end{aligned}$$

We continue by computing $D_2(\alpha,c,\varepsilon)$. Table 2.2 shows all possible cases. The relevant consequences to the drift are the following

1. We get to a population with $\{x, x_1\}$ with $f(x) = f(x_1)$. This state satisfies the conditions of Lemma 2.3. The drift of this case is hence given by $(1-\alpha)\varepsilon c e^{-c}L(\alpha,c,\varepsilon)$.
2. We create offspring y_1 fitter than x. We can apply Lemma 2.2 and we get that, with high probability, y_1 will be the fittest element of the next initial population. The value of $X_i - X_{i+1}$ is t. We sum the contributions for all possible values of t and we get

$$\sum_{t=1}^{n(1-\alpha)(1-\varepsilon)} t\alpha\varepsilon c\binom{n(1-\alpha)(1-\varepsilon)}{t}\left(\frac{c}{n}\right)^t(1-\frac{c}{n})^{n-t-1}$$

   And, by applying the same computations of the proof of Lemma 2.3, we get an upper bound of

$$\alpha\varepsilon c^2(1-\alpha)e^{-c}e^{(1-\alpha)c}(1-\mathcal{O}(\frac{t}{n}))$$

   and a lower bound of

$$\alpha\varepsilon c^2(1-\alpha)e^{-c}e^{(1-\alpha)c}(1-\mathcal{O}(\sqrt{\varepsilon}+\frac{t}{n}+n^{-\frac{1}{6}}))$$

Hence, by summing the five contributions and the $\pm\mathcal{O}(\varepsilon^2)$ error terms, we get for n large enough

$$D_2(\alpha,c,\varepsilon) = (1-\alpha)\varepsilon c e^{-c}L(\alpha,c,\varepsilon) + \alpha\varepsilon c^2(1-\alpha)e^{-\alpha c} \pm \mathcal{O}(\varepsilon^2)$$

In order to compute the final expression for $\mathbb{E}[X_i - X_{i+1}]$ we need to compute the probability p of staying in the initial situation. Since p is equal

to one minus the probabilities of leaving the initial state, we get an upper bound of

$$p(\alpha, c, \varepsilon) \leq 1 - \frac{1}{2}\Big(\varepsilon c e^{-c} - (1-\alpha)^2\varepsilon(1-\varepsilon)c^2e^{-c} - \alpha\varepsilon c(e^{(1-\alpha)c} - 1) \\ - \alpha^2\varepsilon(1-\varepsilon)c^2e^{-c} - (1-\alpha)\varepsilon c e^{-c} - \alpha\varepsilon c e^{-\alpha c}\Big)$$

By combining the results we obtain the drift

$$\frac{D_1(\alpha, c, \varepsilon) + D_2(\alpha, c, \varepsilon)}{2(1 - p(\alpha, c, \varepsilon))}$$ □

which proves the lemma.

0-flips HT	1-flips HT	0-flips out	1-flips out	Consequences
0	0	0	0	Stay in initial state
0	0	0	1	We create offspring with the same fitness as y. With probability 0.5 we stay in the initial population, otherwise, by Lemma 2.2, we have a drift of $\pm\mathcal{O}(\varepsilon)$ with high probability
0	0	0	$t > 1$	We discard the offspring since it is less fit than y
0	0	1	0	See 1
0	0	1	1	See 2
0	0	1	$t > 1$	We discard the offspring since it is less fit than y
0	0	$t > 1$	$t' \in [0, \infty)$	The contribution is negligible since we flip more than a single zero-bit
0	$t \geq 1$	$t' \in [0, \infty)$	$t'' \in [0, \infty)$	We discard the offspring since it is less fit than y
1	0	0	0	See 3
1	0	0	1	We generate offspring x_1 fitter than x but such that OM (x_1)=OM (x). According to Lemma 2.2, the drift from this state will be $\pm\mathcal{O}(\varepsilon)$ with high probability
1	0	0	$t > 1$	See 4
1	0	$t \geq 1$	0	The contribution is negligible since we flip more than a single zero-bit
1	1	0	0	See 5
1	1	0	1	We create offspring with the same fitness as y. With probability 0.5 we stay in the initial population, otherwise, by Lemma 2.2, we have a drift of $\pm\mathcal{O}(\varepsilon)$ with high probability
1	1	0	$t > 1$	We discard the offspring since it is less fit than y
1	1	$t \geq 1$	$t' \in [0, \infty)$	The contribution is negligible since we flip more than a single zero-bit
1	$t > 1$	$t' \in [0, \infty)$	$t'' \in [0, \infty)$	We discard the offspring since it is less fit than y

Table 2.1: Case distinction considering that Algorithm 2 mutates the fittest element of the initial population.

0-flips HT	**1-flips HT**	**0-flips out**	**1-flips out**	**Consequences**
0	0	0	$t \in [0, \infty)$	We stay in initial state
0	0	1	0	See 1
0	0	1	1	We create offspring with the same fitness as y. With probability 0.5 we stay in the initial population, otherwise, by Lemma 2.2, we have a drift of $\pm\mathcal{O}(\varepsilon)$ with high probability
0	0	0 1	$t > 1$	We discard the offspring since it is less fit than y
0	0	$t > 1$	$t' \in [0, \infty)$	The contribution is negligible since we flip more than a single zero-bit
0	$t \geq 1$	$t' \in [0, \infty)$	$t'' \in [0, \infty)$	We discard the offspring since it is less fit than y
1	0	0	0	We create an offspring y_1 that, according to Lemma 2.2, will be the fittest element of the next special population with high probability. Since OM (x) =OM (y_1) we have a contribution to the drift of zero with high probability.
1	0	0	$t \geq 1$	See 2
1	0	$t \geq 1$	$t' \in [0, \infty)$	The contribution is negligible since we flip more than a single zero-bit
1	1	0	0	We create an offspring with the same fitness as y. According to Lemma 2.2, we will have a drift of $\pm\mathcal{O}(\varepsilon)$ with high probability.
1	1	0	$t \geq 1$	We discard the offspring since it is less fit than y
1	1	$t \geq 1$	$t' \in [0, \infty)$	The contribution is negligible since we flip more than a single zero-bit
1	$t > 1$	$t' \in [0, \infty)$	$t'' \in [0, \infty)$	We discard the offspring since it is less fit than y

Table 2.2: Case distinction considering that Algorithm 2 mutates the less fit element of the initial population.

2.3 Analysis of the Drift

In the previous section we proved Lemma 2.5, which expresses the drift $D(\alpha, c, \varepsilon)$. As we will argue in Section 2.4, if we find an $\alpha_0 \in [0,1]$ for a given c_0 such that $D(c_0, \alpha_0, \varepsilon)$ is positive for ε small enough, then Algorithm 2 with mutation parameter c_0 has expected exponential runtime on HotTopic functions with parameters α_0, ε and n large enough. Lemma 2.6 gives us a strong result in this sense: it formally proves that there is an instance of HotTopic function such that the $(2+1)$-EA, with mutation parameter larger than 1.66, has expected exponential runtime.

Lemma 2.6 *Let $\alpha_0 = 0.3$. Then, for all $c > 1.66$, we have $D(\alpha_0, c, \varepsilon) > 0$.*

Proof We observe that $D(\alpha_0, c, \varepsilon) = \frac{D_1(\alpha_0,c,\varepsilon)+D_2(\alpha_0,c,\varepsilon)}{2(1-p(\alpha_0,c,\varepsilon)}$. Since both the constant factor two and $1-p$ are positive (the latter one because it is the sum of the probabilities of the outgoing arrows), they don't influence the sign of $D(\alpha_0, c, \varepsilon)$. Hence it is sufficient to analyse the sign of $D_1(\alpha, c, \varepsilon) + D_2(\alpha, c, \varepsilon)$. We get the following

$$
\begin{aligned}
&- \varepsilon c e^{-c}(1 - (1-\alpha_0)^2(1-\varepsilon)cL(\alpha_0, c) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} + \alpha_0(e^{(1-\alpha_0)c} - 1) \\
&- \alpha_0^2(1-\varepsilon)cL(\alpha, c, \varepsilon) - (1-\alpha_0)L(\alpha_0, c) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c})
\end{aligned}
$$

and hence it is sufficient to show

$$
\begin{aligned}
&1 - (1-\alpha_0)^2(1-\varepsilon)cL(\alpha_0, c) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} + \alpha_0(e^{(1-\alpha_0)c} - 1) \\
&- \alpha_0^2(1-\varepsilon)cL(\alpha_0, c) - (1-\alpha_0)L(\alpha_0, c) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} < 0
\end{aligned}
$$

In order to do that we need a lower bound for $L(\alpha_0, c)$. Since Lemma 2.4 sates that, for any fixed value of α, the function $L(\alpha, c)$ is monotonic increasing in c, we easily get a lower bound by substituting $c = 1.66$ in our expression. We get a lower bound of $L(\alpha_0, c) = -0.33$. By setting this value in the previous inequality we get

$$
\begin{aligned}
&1 + 0.33(1-\alpha_0)^2 c - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} + \alpha_0(e^{(1-\alpha_0)c} - 1) \\
&+ 0.33\alpha_0^2 c + 0.33(1-\alpha_0) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} < 0
\end{aligned}
$$

In order to prove that $f(c) := 1 + 0.33(1-\alpha_0)^2 c - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} + \alpha_0(e^{(1-\alpha_0)c} - 1) + 0.33\alpha_0^2 c + 0.33(1-\alpha_0) - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} < 0$ for all $c > 1.66$ we do the following: we observe that $f(1.66) = -0.02 < 0$ and we observe that $f(c)$ is monotonic decreasing in c. We hence just have to show

that $f'(c) < 0$ for all $c > 1.66$. We have

$$\begin{aligned} f'(c) &= 0.33(1-\alpha_0)^2 - \alpha_0 c(1-\alpha_0)e^{(1-\alpha_0)c} - 2\alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c} \\ &\quad + \alpha_0(1-\alpha_0)e^{(1-\alpha_0)c} + 0.33\alpha_0^2 \\ &= 0.33(1-\alpha_0)^2 + 0.33\alpha_0^2 - 2\alpha_0 c(1-\alpha_0)^2 e^{(1-\alpha_0)c} + \alpha_0(1-\alpha_0)e^{(1-\alpha_0)c}(1-c) \\ &\leq 0.17 + 0.03 - 1.5 - 0.44 < 0 \end{aligned}$$

where in the last step we used that $f'(c)$ is monotonically decreasing in c, which naturally follows from its form. This concludes the proof. □

We observe that the proof of the lemma does not work for other values of $c < 1.66$: for example $f(1.64) > 0$ and hence the proof would not hold for $c = 1.64$.

The statement of Lemma 2.6, together with the formal arguments that we present in Section 2.4, gives an answer to the question of this chapter: using the duplicate avoidance mechanisms on the $(2+1)$-EA hurts. In fact, for all $c \in [1.66, 2.14)$, Algorithm 2 has expected exponential runtime, while the classical $(2+1)$-EA has expected runtime $\Theta(n \log n)$. A scenario that gives intuitive insights on why this hold is the following: consider that in a population $\{x^{(1)}, x^{(2)}\}$ with $f(x^{(1)}) > f(x^{(2)})$ the algorithm chooses $x^{(1)}$ for the mutation operator. If it doesn't flip any bit in $x^{(1)}$ (which happens with constant probability $q \approx e^{-c}$), then the classical $(2+1)$-EA accepts the offspring in the population and discards the less fit element $x^{(2)}$, while the $(2+1)$-EA with duplicate avoidance discards the offspring because it is a copy of $x^{(1)}$. The choice of the classical variant of the algorithm is better for two reasons: first it generates a population with an higher overall fitness, second it provides an additional possibility (which has constant probability) to improve the fitness of an element in the population without flipping any zero-bit (and flipping a zero-bit has probability $\mathcal{O}(\varepsilon)$, which is small in the situation close to the optimum). We recall that this result indicates the superiority of the classical algorithm because it suggests that it is more robust and in general we prefer algorithms that work for a broad choice of parameters.

In [8], Lengler showed that for the classical $(2+1)$-EA there is a threshold $c_0 \approx 2.14$ such that the expected runtime of the algorithm is exponential for $c > 2.14$ and polynomial for $c < 2.14$. Until now we have only showed that for the variant of the algorithm with duplicate avoidance, we have expected exponential runtime for all $c > 1.66$. Although we have not been able to prove if formally[2], we believe that for all $\alpha \in [0,1]$, for all $c < 1.64$ and

[2]The problem is that both the equation to find the maximum value of the drift in function of c for a fixed α and the equation to find the maximum value of the drift in function of α for a fixed c don't have an analytical solution.

for ε small enough, we have $D(\alpha, c, \varepsilon) < 0$. If this last statement holds, then we would be able to prove that for Algorithm 2 there is a threshold $c_0 \in [1.64, 1.66]$ analogous to the $c_0 \approx 2.14$ found by Lengler.

In the remainder of this section we justify our belief that, for all $\alpha \in [0, 1]$ and for all $c < 1.64$, the value of $D(\alpha, c, \varepsilon)$ for ε small enough is negative. The informal arguments that we list now reflect what we did in order to "guess" the value $c = 1.66$ in Lemma 2.6. The first thing we tried to do was finding the value of α (as a function of c) that maximizes the drift. After computing the derivative we got a trascendental equation that does not have analytical solutions. In order to find the value of α that maximizes the drift for some c, we plot the functions shown in Figure 2.2-2.5. They show the value of the drift (the red function) with respect to α (the variable on the x-axis) for a fixed value $\varepsilon = 0.0000001$ and multiple values of c.

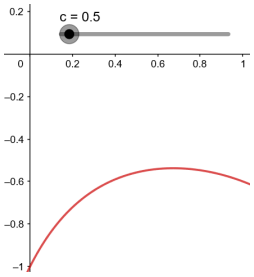

Figure 2.2: Value of the drift (y-axis) w.r.t. α (x-axis) with fixed $\varepsilon = 0.0000001$ and $c = 0.5$

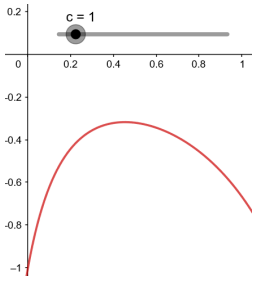

Figure 2.3: Value of the drift (y-axis) w.r.t. α (x-axis) with fixed $\varepsilon = 0.0000001$ and $c = 1$

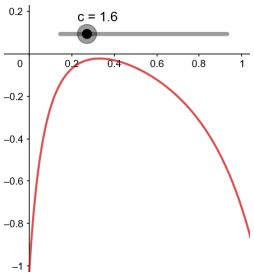

Figure 2.4: Value of the drift (y-axis) w.r.t. α (x-axis) with fixed $\varepsilon = 0.0000001$ and $c = 1.6$

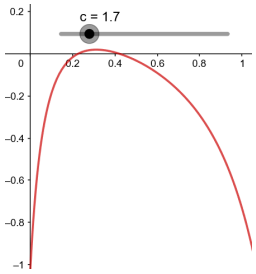

Figure 2.5: Value of the drift (y-axis) w.r.t. α (x-axis) with fixed $\varepsilon = 0.0000001$ and $c = 1.7$

From the plots above we observe that, apparently, the maximum value of the drift increases together with the mutation parameter c. For $c = 1.6$ the drift is negative for all $\alpha \in [0, 1]$, but for $c = 1.7$ we have a positive drift for example at $\alpha = 0.3$. This suggest that there is a threshold in this interval that make the runtime of Algorithm 2 going from quasi-linear to exponential. In

general, the value of α that maximizes the drift for a fixed value of c changes and it seems decreasing with higher values of c. In Lemma 2.6 we chosed $\alpha_0 = 0.3$ because this is a value that makes the drift positive with $c = 1.7$. The main utility of this plots is showing that considering $\alpha_0 = 0.3$ as the value that maximizes the drift close to 1.66 is a good choice.

So far we justified our choice of $\alpha_0 = 0.3$ and that there could be a threshold in the interval $[1.6, 1.7]$. In order to get a more precise threshold, we used MATLAB. The following MATLAB snippet suggests that, since $D(\alpha, c, \varepsilon)$ is continuous, the drift is negative for all $c < 1.64$ and has a zero at $c \approx 1.6506$.

```
>>syms a c eps
>>L_NUMERATOR = - a*(exp(-c*(a - 1)) - 1) - >>a*c*exp(-c*(a - 1))*(a - 1) - 1
>>L_DENOMINATOR = a*(exp(-c*(a - 1)) - 1) + 1
>>L = L_NUMERATOR/L_DENOMINATOR
>>D1 = -eps*c*exp(-c)+(1-a)^2*eps*(1-eps)*c^2*exp(-c)L
     +a*eps*c^2*(1-a)*exp(-a*c)-a*eps*c*exp(-c)*(exp(c*(1-a))-1)
     +a^2*eps*(1-eps)*c^2*exp(-c)*L
>>D2 = (1-a)*eps*c*exp(-c)*L+a*eps*c^2*(1-a)*exp(-a*c)
>>p = eps*c*exp(-c)+(1-a)^2*eps*(1-eps)*c^2*exp(-c)
    +a*eps*c*(exp((1-a)*c)-1)+a^2*eps*(1-eps)*c^2*exp(-c)
    +(1-a)*eps*c*exp(-c)+a*eps*c*exp(-a*c)
>>D = (D1+D2)/p

>>vpasolve(subs(D,{a,eps},{0.33,0.000000001})==0,c,[0,1.64])
    OUTPUT: NO SOLUTION FOUND
>>if(subs(D,{a,c,eps},{0.33,1.6,0.000000001})<0)disp('Negative drift');end
    OUTPUT: Negative drift
>>vpasolve(subs(D,{a,eps},{0.3,0.000000001}==0,c,[0,1.66])
    OUTPUT: ans = 1.650605930990846032l441965861794
```

Summarising, in this section we formally proved that for all $c > 1.66$ there is an instance of HotTopic function such that Algorithm 2 has expected exponential runtime. Moreover we believe that there is a threshold $c_0 \approx 1.6506$ such that for all $c < c_0$ the $(2+1)$-EA has expected runtime $\Theta(n \log n)$. This belief is justified by the following facts:

- A value of $\alpha_0 = 0.3$ is close to the value of α that maximizes $D(\alpha, c, \varepsilon)$ for ε small enough and c close enough to 1.65.
- $D(0.3, c, 0.0000001)$ is negative for all values of $c < 1.64$ and is equal to zero for $c \approx 1.6506$.

2.4 Proof of the Runtime

In the previous sections we have proved that, for all $c > 1.66$, there is an instance of HotTopic with ε small enough that has positive drift, *i.e.* with a drift far away from the optimum. This is an important result, because it implies that the runtime of Algorithm 2 with such a mutation parameter needs exponential time to optimize such a function. Now we want to formally prove this fact, by closely following the exposition in [11].

Lemma 2.7 *For every constant $0 < \delta < \frac{2}{7}$ the following holds. Let $\ell \in [L]$ and consider the $(2+1)$-EA with duplicate avoidance under the following assumptions*

- $d([n], x) \geq \varepsilon(1+2\delta)$
- $\varepsilon(1+\frac{\delta}{2}) \leq d(A_{\ell+1}, x) \leq d([n], x) + \delta\varepsilon$

hold for both search points in the initial population. For $t \geq 0$, let $x^{(t)}$ be the offspring in iteration t. Then, with probability $1 - e^{-\Omega(\varepsilon n)}$, the following holds for all $t \leq L$.

1. $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$
2. $d(A_{\ell+1}, x^{(t)}) \geq \varepsilon(1+\frac{\delta}{4})$ *or* $d([n], x^{(t)}) \geq \varepsilon(1+2\delta)$

Proof First, we recall that X_i is the random variable for the number of one-bits in the fittest element of the i-th special population. We stucture the proof by doing a case distinction on the number of steps.

- For a number of steps $i < B := \frac{\alpha n \delta}{8c}$ we prove that $X_i \leq X_0 + \frac{\varepsilon\delta n}{2}$ and that this implies $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$. Moreover, we show that $d(A_{\ell+1}, x) \geq \varepsilon(1+\frac{\delta}{4})$ holds with probability $1 - e^{-\Omega(\varepsilon n)}$.
- For a number of steps $i > B$, we can apply the negative drift theorem to show that $X_i \leq X_0$. This implies $d([n], x^{(t)}) \geq \varepsilon(1+2\delta)$ and, since this value is larger that $\varepsilon(1+\delta)$, it implies both statements of the lemma.

We begin with the case of a number of steps $i < B$. In the first part of the proof, we consider a maximum number of $B := \frac{\alpha n \delta}{8c}$ steps. We first show that, with probability $1 - e^{-\Omega(\varepsilon n)}$, we have

$$d(A_{\ell+1}, x^{(t)}) \geq \varepsilon(1+\frac{\delta}{4})$$

In order to do that we observe that we the density of $d(A_{\ell+1})$ drops below $\varepsilon(1+\frac{\delta}{4})$ if we flip at least $\frac{\varepsilon\delta\alpha n}{4}$ zero-bits in the hot topic of the current level. We show that this happens with low probability. In order to do that we use a Chernoff bound on a random variable X that denotes the number of zero-bits flipped in the current hot topic. Since $\mathbb{E}[X] = \frac{\alpha^2 n\varepsilon\delta}{8}$, we get an upper bound of

$$e^{-\frac{1}{24}(\alpha^2-4\alpha+4)\delta\varepsilon n}$$

which is exponentially small in n, and hence the second statement of the lemma is proven. We use a similar argument to show that, with probability $1 - e^{-\Omega(\varepsilon n)}$, $X_i \leq X_0 + \frac{\varepsilon\delta n}{2}$ for $i < B$. We define again another random variable X that counts the number of bits flipped in B steps. It holds $\mathbb{E}[X] = \frac{\alpha n \varepsilon \delta}{8}$ and an upper bound for the probability of flipping more that $\frac{\varepsilon\delta n}{2}$ zero-bits is

$$e^{-\frac{1}{24}\frac{(\alpha^2-8\alpha+16)}{\alpha}\varepsilon\delta n}$$

which is exponentially small in n. We now have to show that $X_i \leq X_0 + \frac{\varepsilon\delta n}{2}$, together with the assumption $d([n], x) \geq \varepsilon(1+2\delta)$, implies $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$ for at most B steps. We have

$$\begin{aligned}
d([n], x^{(t)}) &= \frac{n - X_i}{n} \\
&\geq \frac{n - X_0 - \frac{\varepsilon\delta n}{2}}{n} \\
&\geq \frac{n - (n - \varepsilon(1+2\delta)n) - \frac{\varepsilon\delta n}{2}}{n} \\
&= \frac{n\varepsilon(1+2\delta) - \frac{\varepsilon\delta n}{2}}{n} \\
&= \varepsilon(1 + \frac{3}{2}\delta) \\
&\geq \varepsilon(1+\delta)
\end{aligned}$$

which proves the first statement of the lemma.

Now that we have proved the lemma for $i < B$, we prove that both statements also hold for $i > B$. In particular, we need B to be linear in n. We first show that $X_i \leq X_0$ with high probability. In order to do that we just need to show how to apply the negative drift theorem, in the version presented in [6] and reported as Theorem 1.2 in Chapter 1. We can replace the variable X_i of the theorem with our variable X_i, and d is the negative drift we have found in Section 2.2 (note that previously we have shown that $\mathbb{E}[X_i - X_{i+1}] > 0$, hence we can apply the argument by flipping the order of the variables and the direction of the inequality). By choosing $x = -di$ (which is positive since d is negative), we get that an upper bound for the probability of $X_i - X_0$ being larger than 0 is

$$e^{\frac{di}{2}\min(\delta, \frac{-di}{cdi})} \in e^{-\Omega(n)}$$

because $d < 0$ and $i > B$ (and hence linear in n). We now just have to prove that $X_i \leq X_0$ implies that $d([n], x^{(t)}) \geq \varepsilon(1+2\delta)$ (and automatically

also $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$). Hence we can kill two birds with one stone by proving both statements at the same time. We have that $X_0 = n - n \cdot d([n], x) \leq n - n\varepsilon(1+2\delta)$, where we used the assumption $d([n], x) \geq \varepsilon(1+2\delta)$. We get

$$\begin{aligned} d([n], x^{(t)}) &= \frac{n - X_i}{n} \\ &\geq \frac{n - (n - \varepsilon(1+2\delta))}{n} \\ &= \varepsilon(1+2\delta) \end{aligned}$$

which completes the proof also for the case $i > B$. □

Lemma 2.7 was rather technical, but it will be useful to prove Lemma 2.8, a fundamental lemma that will allow us to conclude that the runtime of the $(2+1)$-EA with duplicate avoidance and $c > 1.66$ has exponential optimization time on some instances of HotTopic with ε small enough. Our proof of the runtime will consider an auxiliary process. The auxiliary process considers the behaviour of Algorithm 2 on a function $\tilde{f}$, which is completely equivalent to the HotTopic function apart from a single detail: in HotTopic, the level increases to the maximum $\ell' \in L$ such that the number of zero-bits in $B_{\ell'}$ is at most $\varepsilon\beta n$; in $\tilde{f}$ the level can increase at most by one. Formally, we define the level as $\tilde{\ell}(0) = 0$, and if an offspring $y^{(t)}$ of $x^{(t)}$ enters the population in round t, then we set $\tilde{\ell}(y^{(t)}) := \max\{\ell' \in \left[\min\{\ell'(x^{(t)}) + 1, L\}\right] : |\{j \in B_\ell : x_j = 0\}| \leq \varepsilon\beta n\}$. Then the fitness of a search point x with respect to $\tilde{f}$ is

$$\tilde{f}(x) = \tilde{\ell}(x) \cdot n^2 + \sum_{i \in A_{\ell(x)+1}} x_i \cdot n + \sum_{i \notin A_{\ell(x)+1}} x_i.$$

.

In Lemma 2.8 we show a useful property of the auxiliary process, and in Lemma 2.9 we will show that the auxiliary process is equivalent to the process we investigate in this chapter.

Lemma 2.8 *There is a constant $\delta > 0$ such that the auxiliary process satisfies $d([n], x^{(t)}) > \varepsilon \cdot (1+\delta)$ for all $t \leq L$.*

Proof The advantage of the auxiliary process is that we may postpone drawing $A_{\ell+1}$ until we reach level $\tilde{\ell} = \ell$. In particular, since $A_{\ell+1} \subseteq [n]$ is a uniformly random subset, we may use the same assumptions of Lemma 2.7 to observe that $|d(A_{\ell+1}, x) - d([n], x)| \leq \delta\varepsilon$ holds with probability $1 - e^{-\Omega(\varepsilon n)}$

for both members x of the population when we reach level ℓ. The exponentially small error probability allows us to use a union bound and conclude that with high probability the same holds for all ℓ. We want to show that the auxiliary process, if running on level ℓ and starting with a population that initially satisfies $|d(A_{\ell+1}, x) - d([n], x)| < \delta\varepsilon$ for $\delta < \frac{2}{7}$, maintains $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$ for all new search points $x^{(t)}$ until $t > L$. By the first conclusion of Lemma 2.7, $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$ holds as long as the level remains to be ℓ and $t \leq L$. When a search points x reaches level $\ell+1$, by definition, we have $d(B_{\ell+1}, x) < \varepsilon$. Since $B_{\ell+1}$ is a uniformly random subset of $A_{\ell+1}$, by Chernoff bound, we get that

$$Pr\left[d(A_{\ell+1}, x) \geq \varepsilon(1+\frac{\delta}{4})\right] \leq e^{-\Omega(\varepsilon n)}.$$

So we apply the second conclusion of Lemma 2.7 and conclude that $d([n], x) \geq \varepsilon(1+2\delta)$. With high probability, it holds that $d(A_{\ell+2}, x) \geq \varepsilon(1+2\delta) - \varepsilon\delta$ and the conditions of Lemma 2.7 are satisfied again for level $\ell+1$. By induction we obtain $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$ for all $t \leq L$. As the choice of ℓ is arbitrary, we start with level $\ell = 0$ and $d([n], x^{(t)}) \geq \varepsilon(1+\delta)$ holds for all $t \leq L$, which concludes the proof. □

Lemma 2.8 is the final tool that we need to prove the exponential runtime of our algorithm with $c > 1.66$. The next lemma formally proves this statement.

Lemma 2.9 *For every $c > 1.66$ there is a parameter α of the* HotTopic *function such that the $(2+1)$-EA with mutation parameter c visits each level of the* HotTopic *function at least once. Since the number of levels is exponential, the algorithm needs an exponential number of steps with high probability and in expectation. In particular, with high probability, the optimization time is exponential.*

Proof Let $L = e^{\rho n}$ be the number of levels. In order to prove the statement of the lemma, we show that the $(2+1)$-EA with duplicate avoidance on HotTopic behaves in the same way as on the auxiliary process $\tilde{f}$.

We want to show that Algorithm 2 has the same behaviour on HotTopic and on $\tilde{f}$: this fact, together with the observation that $\tilde{f}$ visits an exponential number of levels, proves the expected exponential runtime of the $(2+1)$-EA with duplicate avoidance and sufficiently large c for some instance of HotTopic. For the auxiliary process, *i.e.* the algorithm on $\tilde{f}$, we need to uncover A_{i+1} and B_{i+1} when a search point in the population reaches level i. As we have already argued, we have $d([n], x^t) \geq \varepsilon(1+\delta)$ for a suitable constant $\delta > 0$. It is useful to assign every round t to a level $\ell(t)$. Since we will need to uncover $B_{\ell(t)+2}$ at some point after time t (*i.e.* when its density of zero-bits will be sufficiently small), its choice does not influence the behavior of the auxiliary process until time t. Hence,we can first let the auxiliary

process run until time t, and afterwards uncover the set $B_{\ell(t)+2}$. Since $B_{\ell(t)+2}$ is a uniformly random subset of size βn and $d([n], x^t) \geq \varepsilon(1+\delta)$, it contains at least $\beta\varepsilon(1+\delta)n$ zero-bits in expectation. Moreover, by Chernoff bound, the probability that $B_{\ell(t)+2}$ contains at most $\beta\varepsilon n$ zero-bits is

$$e^{-\frac{\delta^2}{2(1+\delta)}\beta\varepsilon n} = e^{-\Omega(\beta\varepsilon n)}$$

The same argument also holds for $B_{\ell(t)+3}, \ldots, B_L$. Since $L = e^{\rho n}$ with desirably small $\rho > 0$, we can afford a union bound over all such sets and all $t \leq L$, which is a union bound over less than $L^2 = e^{2\rho n}$ terms. Hence, with high probability we have

$$d(B_i, x^{(t)}) > \varepsilon \qquad \forall 1 \leq t \leq T \text{ and } \tilde{\ell}(t) + 2 \leq i \leq L \tag{2.3}$$

We now show by induction that $\ell(t) = \ell(\tilde{t})$. The base case $\ell(0) = \ell(\tilde{0})$ follows from definition. For the induction step we have, using 2.3,

$$\ell(i+1) = \max\{\ell' \in [L] : d(B_{\ell'}, x) \leq \varepsilon\} = \tilde{\ell}(i+1) \qquad \square$$

This implies that Algorithm 2 behaves in the same way in HotTopic and in $\tilde{f}$ and this concludes the proof of the exponential optimization time.

We have finally proven the main result of the thesis: Algorithm 2 with mutation rate $c > 1.66$ needs exponential runtime to optimize some instance of HotTopic functions. This means that that we have an example that shows that using a diversity preserving mechanism such as duplicate avoidance hurts. In fact, the classical $(2+1)$-EA with mutation rate $c \in [1.66, 2.14)$ has quasi-linear runtime.

Although we have not been able to prove it formally, in Section 2.3 we provided insights on why we believe that for all $c < 1.64$ the $(2+1)$-EA with duplicate avoidance has drift towards the optimum for all HotTopic functions. Both the results hold for a sufficiently small value of ε, *i.e.* close to the optimum. For the sake of completeness we now prove that if, as we believe, the drift for all $c < 1.64$ is towards the optimum, then the optimization time is quasi-linear in n.

Lemma 2.10 *If for all $c < 1.64$ and all $\alpha \in [0,1]$ we have $\mathbb{E}[X_i - X_{i+1}] < 0$, then Algorithm 2 optimizes all instances of* HotTopic *functions in expected runtime $\Theta(n \log n)$.*

Proof For the lower bound, we use a Chernoff bound to observe that the probability that the initial strings have at least $\frac{n}{3}$ zero-bits is at least $1 - e^{-\frac{1}{54}n}$. and that each of this $\frac{n}{3}$ bits needs to be flipped at least one in order to reach the optimum. Since this is essentially a coupon collector process, we need to flip at least $\frac{n}{3}H_{\frac{n}{3}} + \mathcal{O}(n)$ bits in expectation and with high probability, where H_n denotes the $n-$th armonic number. Thus we also need $\Omega(n \log n)$ rounds to achieve so many flips, since in each round we expect to flip a constant number of bits with $c < 1.64$.

Now we analyse the upper bound and we show that it is $\mathcal{O}(n \log n)$. We define $Y_i = n - X_i$, *i.e.* Y_i denotes the number of zero-bits in the i-th special population. Let ε_0 be the parameter of the HotTopic function. We can show that $Y_i \leq C\varepsilon_0 n$ for a constant $C > 0$. In the previous section we showed that Algorithm 2 with $c < 1.64$ satisfies $\mathbb{E}[X_i - X_{i+1}] < 0$ for all instances of HotTopic. We can reformulate this result and we get $\mathbb{E}[Y_i - Y_{i+1}] > \frac{C'}{2} \cdot \varepsilon$. By defining $h(Y_i) := \frac{C'}{2n} Y_i$ we can apply the additive drift theorem (see Theorem 1.3 in Chapter 1) and we get

$$\mathbb{E}[T] = \frac{1}{h(1)} + \int_1^n \frac{2n}{c'} \cdot \frac{1}{y} dy = \mathcal{O}(1) + \frac{2n}{c'}(\log(n) - \log(1)) = \mathcal{O}(n \log n)$$

which proves that the value of the upper bound coincides with the lower bound and hence the lemma is proven. □

Chapter 3

Conclusions and Future Work

We have studied the impact of the duplicate avoidance mechanism for the $(2+1)$-EA on HotTopic close to the optimum, and we showed that it hurts. In fact, for all $c \in [1.66, 2.14)$, the classical $(2+1)$-EA has an expected optimization time $\Theta(n \log n)$, while the variant that promotes genotypic diversity in the population needs exponential time to optimize the function with high probability and in expectation. The reason for this is that the classical $(2+1)$-EA, starting from a population $\{x, y\}$ with $f(x) > f(y)$, allows to create a population $\{x, x\}$ with constant probability: it is sufficient to choose x as parent and not touching any of its bits (which happens with probability $\approx e^{-c}$). Duplicate avoidance forbids this shortcut: in order to create a population with the same total amount of one-bits of $\{x, x\}$ starting from $\{x, y\}$, it needs to flip at least a zero-bit which, close to the optimum, happens rarely.

Many questions about the impact of diversity preserving mechanisms for evolutionary and genetic algorithms remain open. For example, is there a difference between the $(2+1)$-EA and the $(\mu+1)$-EA with $\mu \in \Omega(1)$? The scenario we explained above generalizes also to larger populations, and hence it could be the case that the diversity preserving mechanism hurts also in the general case: however, it is desirable to prove it formally. And what if μ is not a constant, but an arbitrarily large parameter? The size of the population is also a delicate matter, as presented in [11]. The same mechanism could also be applied to genetic algorithms, a variant of evolutionary algorithms that uses non only mutation, but also the crossover operator in order to not "throw away" good genes of search points with lower fitness. The general trend is that crossover helps (particularly in a function as HotTopic, where we want to avoid to flip a zero-bit in the hot topic and many one-bits outside of it), but it would be interesting to study the impact of duplicate avoidance also in that scenario. These questions are valid both to the situation close to the optimum and the general situation with a larger value of ε. Moreover, duplicate avoidance is only one example of diversity pre-

serving mechanism: other diversity preserving mechanisms such as fitness diversity, deterministic crowding, fitness sharing and clearing are of interest and their impact on HotTopic has not been investigated yet. There are still many possibilities.

Bibliography

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

[2] Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges. Optimizing monotone functions can be difficult. In *International Conference on Parallel Problem Solving from Nature*, pages 42–51. Springer, 2010.

[3] Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges. Mutation rate matters even when optimizing monotonic functions. *Evolutionary computation*, 21(1):1–27, 2013.

[4] Tobias Friedrich, Nils Hebbinghaus, and Frank Neumann. Rigorous analyses of simple diversity mechanisms. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1219–1225, 2007.

[5] Tobias Friedrich, Pietro S Oliveto, Dirk Sudholt, and Carsten Witt. Analysis of diversity-preserving mechanisms for global exploration. *Evolutionary Computation*, 17(4):455–476, 2009.

[6] Timo Kötzing. Concentration of first hitting times under additive drift. *Algorithmica*, 75(3):490–506, 2016.

[7] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. (1+ 1)-EA on generalized dynamic onemax. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 40–51, 2015.

[8] Johannes Lengler. A general dichotomy of evolutionary algorithms on monotone functions. *IEEE Transactions on Evolutionary Computation*, 2019.

[9] Johannes Lengler, Anders Martinsson, and Angelika Steger. When does hillclimbing fail on monotone functions: An entropy compression argument. In *2019 Proceedings of the Sixteenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 94–102. SIAM, 2019.

[10] Johannes Lengler and Angelika Steger. Drift analysis and evolutionary algorithms revisited. *Combinatorics, Probability and Computing*, 27(4):643–666, 2018.

[11] Johannes Lengler and Xun Zou. Exponential slowdown for larger populations: the (μ+ 1)-EA on monotone functions. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, pages 87–101, 2019.

[12] S Mahfoud. Handbook of evolutionary computation, chapter niching methods, 1997.

[13] Giovanni Squillero and Alberto Tonda. Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. *Information Sciences*, 329:782–799, 2016.

[14] Dirk Sudholt. The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. In *Theory of Evolutionary Computation*, pages 359–404. Springer, 2020.

[15] Rasmus K Ursem. Diversity-guided evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 462–471. Springer, 2002.

[16] Carsten Witt. Runtime analysis of the (μ+ 1)-EA on simple pseudo-boolean functions. *Evolutionary Computation*, 14(1):65–86, 2006.

[17] Carsten Witt. Tight bounds on the optimization time of the (1+ 1)-EA on linear functions. Technical report, 2011.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

IMPACT OF DUPLICATE AVOIDANCE FOR THE (2+1)-EA ON MONOTONE FUNCTIONS

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Micheletti	Joel Andrea

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Minusio, 13th July 2020

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.