Artificial Immune Systems for Combinatorial Optimisation

A Theoretical Investigation

The University Of Sheffield.

Donya Yazdani

Department of Computer Science
The University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2020

To my childhood.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Donya Yazdani
February 2020

Acknowledgements

I would like to express my gratitude to my supervisor Pietro Oliveto, from whom I have learned immensely, even when he was not intending to teach me directly. I want to thank him for giving me the opportunity to do this PhD and for always actively being part of the projects. I am also extremely grateful to Dogan Corus for all his help and support during my studies.

I should thank Graham Birtwistle and David White for being so encouraging and kind, helping me to polish my thesis. Also to Pietro, Andrei, Dogan and George for proof reading my thesis and my examiners John Clark and Thomas Jansen for insightful comments to improve this manuscript. I am also thankful to my second supervisor Dirk Sudholt for all his suggestions and discussions.

Special thanks to my friends in the Algorithms group who have been great colleagues: George, Edgar, Sam, Alasdair, Mario, Jorge, Andrei, Pan, Sebastian, Marios, and Daniel. I also want to thank all my friends in the DCS who made my PhD experience a joyful journey: August, Fariba, Chunchao, Juan, Lili, Abdullah, Anna, Will, Nasser, Nada, Mina, and Alison.

My PhD journey turned out to be a challenging, yet rewarding, period of my life. Hereby, I want to expand my special thanks to everyone who helped me during this time. After this PhD, I can easily survive a pack of twelve wolves (maybe not bare handed). This journey would not have been possible without the help of stunning people I have met: my dear friends Nabi, George, Sahar, Sina, and Helga. Thank you.

Some big heart-shaped thanks to Danial, Delaram, and Hadi, whose love has always lightened my heart. Thank you for pretending that your sister is doing such a life-changing thing. Some greater thanks to Mum, for teaching me how to love unconditionally, and to Dad, for showing me how to be endlessly generous. It happened that I indeed needed these in every step of my life, including my PhD.

Finally, I am so thankful to the University of Sheffield and the Computer Science Department for supporting my studies, and to the helpful staff always coming up with solutions for every problem. Thank you for giving extra support during the Corona pandemic.

Abstract

We focus on the clonal selection inspired computational models of the immune system developed for general-purpose optimisation. Our aim is to highlight when these artificial immune systems (AIS) are more efficient than evolutionary algorithms (EAs). Compared to traditional EAs, AIS use considerably higher mutation rates (hypermutations) for variation, give higher selection probabilities to more recent solutions and lower selection probabilities to older ones (ageing). We consider the standard Opt-IA that includes both of the AIS distinguishing features and argue why it is of greater applicability than other popular AIS. Our first result is the proof that the *stop at first constructive mutation* version of its hypermutation operator is essential. Without it, the hypermutations cannot optimise any function with an arbitrary polynomial number of optima. Afterwards we show that the hypermutations are exponentially faster than the standard bit mutation operator used in traditional EAs at escaping from local optima of standard benchmark function classes and of the NP-hard PARTITION problem. If the basin of attraction of the local optima is not too large, then ageing allows even greater speed-ups. For the CLIFF benchmark function this can make the difference between exponential and quasi-linear expected time. If the basin of attraction is too large, then ageing can implicitly detect the local optimum and escape it by automatically restarting the search process. The described power of hypermutations and ageing allows us to prove that they guarantee $(1+\varepsilon)$ approximations for PARTITION in expected polynomial time for any constant ε. These features come at the expense of the hypermutations being a linear factor slower than EAs for standard unimodal benchmark functions and of eliminating the power of ageing at escaping local optima in the complete Opt-IA. We show that hypermutating with inversely proportional rates mitigates such drawbacks at the expense of losing the explorative advantages of the standard operator. We conclude the thesis by designing fast hypermutation operators that are provably a linear factor faster than the traditional ones for the unimodal benchmark functions and PARTITION, while maintaining explorative power and working in harmony together with ageing.

Table of contents

List of figures **xv**

List of tables **xvii**

Nomenclature **xix**

1 Introduction **1**
1.1 Overview 1
1.2 Structure of the Thesis 5
1.3 Main Contributions 8
1.4 Underlying Publications 9

I Background **11**

2 Artificial Immune Systems **13**
2.1 Introduction 13
2.2 The Natural Immune System of Vertebrates 14
2.2.1 Clonal Selection Theory 14
2.3 Structure of an Artificial Immune System and Immune Inspired Operators . 17
2.3.1 An Evolutionary Algorithmic Scheme 18
2.3.2 Computational Models of Cloning 22
2.3.3 Computational Models of Hypermutation 23
2.3.4 Computational Models of Ageing 25
2.4 Artificial Immune Systems for Optimisation 26
2.4.1 Clonalg 27
2.4.2 B-cell 27
2.4.3 Opt-IA 27
2.5 Conclusion 29

3 Theory of Artificial Immune Systems **33**
3.1 Introduction . . . 33
3.2 Runtime Analysis . . . 33
3.3 Tools for Runtime Analysis . . . 34
3.3.1 Basic Probability Theory . . . 34
3.3.2 Tail Inequalities . . . 35
3.3.3 Artificial Fitness Levels . . . 37
3.3.4 Drift Analysis . . . 37
3.3.5 Ballot Theorem . . . 38
3.3.6 Typical Run Investigations . . . 38
3.3.7 Some Frequently Used Inequalities . . . 38
3.4 Standard Benchmark Functions . . . 39
3.4.1 Unimodal Functions . . . 39
3.4.2 Multimodal Functions . . . 41
3.5 Runtime Analysis of Hypermutation Operators . . . 44
3.5.1 Inversely Fitness Proportional Hypermutation Operators . . . 44
3.5.2 Contiguous Hypermutation Operators . . . 45
3.5.3 Hypermutation Operators with Mutation Potential . . . 46
3.6 Runtime Analysis of Ageing Operators . . . 46
3.6.1 Static Ageing . . . 47
3.6.2 Stochastic Ageing . . . 48
3.6.3 Hybrid Ageing . . . 48
3.7 Runtime Analysis of Complete Artificial Immune Systems . . . 49
3.7.1 The Standard B-cell Algorithm . . . 49
3.8 Conclusion . . . 50

II Standard Opt-IA **51**

4 Analysis for Benchmark Problems **53**
4.1 Introduction . . . 53
4.1.1 Opt-IA Parameters and Settings . . . 54
4.2 Analysis of Hypermutations with Static Mutation Potentials in Isolation . . . 55
4.2.1 Stopping at First Constructive Mutation is Essential . . . 55
4.2.2 Artificial Fitness Levels for Hypermutations . . . 58
4.2.3 Performance Analysis for Unimodal Functions . . . 59
4.2.4 Performance Analysis for Multimodal Functions . . . 60

4.3 Analysis of Ageing in Isolation . . . 61
4.3.1 Ageing Combined with Randomised Local Search . . . 61
4.3.2 Ageing Combined with Standard Bit Mutation . . . 65
4.3.3 Ageing Combined with Hypermutations . . . 67
4.4 Analysis of a Complete Opt-IA . . . 68
4.4.1 A Tight Bound for ONEMAX . . . 69
4.4.2 Where Using Both Ageing and Hypermutations is Essential . . . 70
4.4.3 When Using Ageing and Hypermutations Together is Detrimental . 75
4.4.4 Opt-IA Is Efficient for TRAP Functions . . . 80
4.4.5 Standard Opt-IA with Genotype Diversity . . . 81
4.5 Conclusion . . . 84

5 Analysis for the NP-Hard Partition Problem 87
5.1 Introduction . . . 87
5.2 The Partition Problem . . . 88
5.2.1 State-of-the-Art of Problem-Specific Algorithms . . . 89
5.2.2 State-of-the-Art of General-Purpose Heuristics . . . 90
5.3 General Preliminary Results . . . 90
5.4 Class of Worst Case Instances for Evolutionary Algorithms . . . 93
5.4.1 Standard Bit Mutations Are Inefficient . . . 96
5.4.2 Hypermutations Are Efficient . . . 98
5.4.3 Ageing Is Efficient . . . 102
5.5 $(1+\varepsilon)$ Approximation Ratio Analysis . . . 105
5.5.1 Hypermutations Find $(1+\varepsilon)$ Approximation Ratios . . . 106
5.5.2 Ageing Finds $(1+\varepsilon)$ Approximation Ratios . . . 108
5.6 Conclusion . . . 109

III Fast Opt-IA 111

6 Inversely Proportional Hypermutations 113
6.1 Introduction . . . 113
6.2 Limitations of the Inversely Proportional Hypermutations from the Literature 114
6.3 Inversely Proportional Mutation Potentials with FCM . . . 116
6.3.1 Hamming Distance Based Linear Decrease . . . 116
6.3.2 Fitness Difference Based Exponential Decrease . . . 116
6.3.3 Hamming Distance Based Exponential Decrease . . . 116

6.4 Analysis in Ideal Conditions 117
6.4.1 Behaviour for ONEMAX in Ideal Conditions 118
6.4.2 Behaviour for LEADINGONES in Ideal Conditions 120
6.5 Inefficient Behaviour In Practice 123
6.6 An Efficient Inversely Proportional Hypermutation Operator with Mutation Potentials for Opt-IA in Practice 128
6.7 Conclusion 130

7 Hypermutations with Probabilistic Intermediate Evaluations 133
7.1 Introduction 133
7.2 Parabolic Sampling Distributions 135
7.3 Artificial Fitness Levels for Fast Hypermutations 139
7.4 Fast Hypermutations for Benchmark Functions 140
7.5 Fast Opt-IA for Benchmark Functions 142
7.6 Fast Hypermutations for Classical Combinatorial Optimisation Problems . . 150
7.6.1 Performance Analysis of Fast Hypermutations for PARTITION . . . 151
7.6.2 Performance Analysis of Fast Hypermutations for VERTEX COVER 154
7.7 Conclusion 156

IV Conclusion 157

8 Summary and Final Remarks 159
8.1 Future Work 163

References 165

List of figures

2.1 Binding of an antibody and an antigen 15
2.2 Structure of an antibody 16

3.1 A Boolean hypercube [16] 40
3.2 The ONEMAX function of size $n = 50$ 41
3.3 The CLIFF_d function of size $n = 50$ 42
3.4 The JUMP_k function of size $n = 50$ 43
3.5 The TWOMAX function of size $n = 50$ 43
3.6 The TRAP function of size $n = 50$ 44

4.1 The HIDDENPATH function 71
4.2 The HYPERTRAP_γ function 76

5.1 The $4/3$ approximation local optimum of P^*_ε [90] 93
5.2 A local optimum of G^*_ε 94

6.1 Inversely proportional mutation potentials for ONEMAX 118
6.2 Inversely proportional mutation potentials for LEADINGONES 118
6.3 The geometric representation of Symmetric M_{expoHD} 129

7.1 The parabolic evaluation probabilities from (7.1) 137

List of tables

6.1 The expected runtimes achieved by the $(1+1)$ IA with different MPs and $(1+1)$ EA 117

7.1 The expected runtimes of the $(1+1)$ IA and the Fast $(1+1)$ IA 142

Nomenclature

Acronyms / Abbreviations

AFL Artificial Fitness Levels

AIS Artificial Immune System

AISs Artificial Immune Systems

CDR Complementarity Determining Region

EA Evolutionary Algorithm

EAs Evolutionary Algorithms

FCM Stopping at First Constructive Mutation

GA Genetic Algorithm

GAs Genetic Algorithms

HMP Hypermutation with Mutation Potential

HMPs Hypermutation with Mutation Potentials

MP Mutation Potential

MPs Mutation Potentials

RLS Randomised Local Search

s.t. Such That

SBM Standard Bit Mutation

w.l.o.g. Without Loss of Generality

w.o.p. With Overwhelming Probability

Chapter 1

Introduction

1.1 Overview

Artificial immune systems (AIS) are a general computational paradigm inspired by the working principles of the natural immune system of vertebrates. The learning, classification and memory capabilities of the immune system used to defend the body against threats have found natural applications in pattern recognition [22], virus detection [33, 40] and in solving complex optimisation problems [20]. In this thesis, our focus is on the computational models of the immune system developed for optimisation applications. These models have taken inspiration from different aspects of the *clonal selection process* of the adaptive immune system [45]. During the clonal selection, the best b-cells at recognising an invading pathogen are activated to expand, then undergo *somatic hypermutation* with the hope of creating fitter b-cells. Given the generality of the clonal selection process in evolving responses for different pathogens, AIS models for optimisation are not designed with a specific problem in mind: they rather may be applied to virtually any optimisation problem for which the quality of solutions can be assessed, thus requiring little understanding of the problem. Numerous successful applications of AIS for combinatorial optimisation problems have been reported in the literature. Examples include protein structure prediction [20], graph colouring [18] and hitting set [17].

Due to the similarities of the clonal selection process of the immune system to the natural selection process of Darwinian theory of evolution, the optimisation models of AISs are similar to those of well-established evolutionary algorithms (EAs). While both algorithms operate in similar iterative loops to evolve solutions to optimisation problems [24], their major difference lies in the mutation rate used for variation; EAs tend to have low mutation rates inspired by sexual or asexual reproduction, while AISs apply high mutation rates inspired by the somatic hypermutation of b-cells. What is unclear is on which classes of

problems AISs have better performance compared to that of EAs, hence when they should be preferred to traditional EAs. In this thesis, we aim to provide insights into this matter.

To answer the question of when using AISs is preferable to standard and well-studied EAs from the literature, we first need to decide on a reasonable framework for AISs. By considering different aspects of clonal selection as the source of inspiration, several optimisation models of the immune system have been proposed in the literature. What all these models have in common is the use of a *hypermutation operator* inspired by the somatic hypermutation process. Given a combinatorial search space where each candidate solution to the optimisation problem is a data string of 0-bits and 1-bits with length n (the search space is exponential in the problem size n), the idea behind hypermutation operators is to flip each bit with much higher probability than that of standard EAs. Three main different ways have been proposed in the literature to simulate the somatic hypermutation process for AIS optimisation algorithms. The first operator is the *inversely fitness proportional hypermutation* used by an immune system called Clonalg [23], for which the probability of mutating bits is a function of the fitness of the candidate solution and is decreased with the increase of the fitness. It has been rigorously proven in the literature that such an operator is unable to efficiently optimise even easy unimodal optimisation problems (i.e., problems without any local optima) [93]. The reason is that, the mutation rate (i.e., the probability that each bit flips) at the beginning of the search is so high that it makes it hard for the solutions to improve at all. It has been shown in the literature that if the mutation rate is calculated inversely to the best seen solution instead of an estimate of the optimum, then the operator can solve easy unimodal problems efficiently. However, this results in a very low mutation rate which makes it hard to call the resulting operator *hyper*mutation.

Another approach suggested in the literature to perform hypermutations is based on the observation that somatic hypermutation only happens in some specific areas of the gene sequence [59, 82]. A well-known method using such observation is called *contiguous hypermutation* which is applied by an immune algorithm called B-cell [55]. This operator only flips bits in a randomly selected interval of the bit string. Such a mutation operator is biased in the sense that the order of the bits has an influence on the outcome of the operator [95]. Applying contiguous hypermutation, bits closer to each other have higher probabilities to be flipped simultaneously compared to bits which are further away. Although according to the *No Free Lunch* theorem all randomized search heuristics, both biased and unbiased, have the same performance over all possible problems, being biased is generally not a desirable property for a general purpose algorithm in black box optimisation scenarios where no problem knowledge is assumed to be available.

The third major approach proposed in the literature to perform hypermutations is to consider a *potential* and mutate that number of bits. These hypermutation operators are called *hypermutation with mutation potential* or simply HMP and have been introduced for an immune system called Opt-IA [19]. More formally, a mutation potential (MP) is a function which returns the maximum number of bits that can be flipped within one hypermutation of a bit string. Different methods have been suggested to define this function. Some reasonable ones are the *static MP* where the potential is always fixed, i.e., cn bits are flipped for a positive constant $c \leq 1$, and the *inversely fitness proportional MP* for which the potential depends on the fitness of the solution and decreases with fitness. While the static MP is always linear in n, the inversely fitness proportional MP is dynamic so it may return lower potentials than linear after sufficient progress is made. These MPs are proposed to be used in conjunction with a mechanism called *stopping at first constructive mutation* (FCM) which makes the hypermutation operator stop flipping bits as soon as an improvement is found (i.e., not all the bits returned by the MP are necessarily flipped) [19].

While the theoretical understanding of Clonalg and B-cell has emphasized the considerable drawbacks summarised above to their application for general-purpose optimisation, no similar drawbacks have been pointed out for Opt-IA in the few available results investigating its performance. Since Opt-IA applies a rather reasonable choice of hypermutations (i.e., HMP are unbiased and FCM allows making progress even when the mutation potential is very large) and uses *ageing* operators inspired by the natural death of b-cells, we consider it as an ideal candidate for the analysis of a complete population-based AIS. To examine the performance of AIS, we first analyse its operators in isolation (i.e., simple frameworks with just one AIS operator) for problems with different characteristics and afterwards together as a complete algorithm. Unlike problem tailored algorithms, AISs are not designed to find the optimum as quickly as possible for a single problem but they are designed to find good solutions for a wide range of problems (similarly to other general-purpose randomised search heuristics). Therefore, one obvious question is how much (optimisation) time they need to find the solution for a problem with a given structure. Hence, *runtime analysis* will provide us with insights on which kind of problems AISs perform efficiently and on which poorly.

In this thesis, we rigorously prove that both ageing and static HMP can efficiently escape local optima of both standard multimodal benchmark functions and a classical problem from combinatorial optimisation (i.e., the NP-hard PARTITION problem). While benchmark functions are specifically designed to analyse the performance of randomised search heuristics for search landscapes with specific characteristics, classical combinatorial optimisation problems are more similar to some real-world optimisation applications, hence are more likely to be reflected in practice. Our analyses of ageing and HMP operators in

simple frameworks for optimising benchmark problems reveal that they are generally superior to standard EAs when local optima have to be escaped to identify optimal solutions. We confirm this insight for the more realistic PARTITION problem for which we provide worst-case approximation guarantees for both operators. These results are the first proven time performance guarantees of any AIS for standard combinatorial optimisation problems. While EAs need some problem specific information to guarantee good solutions of approximation ratio $(1+\varepsilon)$ for PARTITION, we show that AISs do not require such information to efficiently guarantee the same approximation ratio.

An important drawback revealed by our analyses is that AISs using static HMPs are not as efficient as standard EAs for hill-climbing (i.e., exploitation - the ability of searching the neighbourhood of a potentially promising search point). The reason is that, as they get closer to the optimum, the probability of improving decreases while the potential remains linear. This in turn implies that, once sufficient progress is made, at each time solutions fail to improve at the beginning of a hypermutation, a linear amount of fitness function evaluations will be wasted. Another drawback that is highlighted by our analyses is that if ageing is used alongside HMPs, as in the standard Opt-IA, then the power of the ageing operator at escaping from local optima vanishes.

One way to overcome the issue of slow exploitation is to set the mutation potential inversely proportional to the fitness. Ideally, if as the optimum is approached, the mutation potential decreases, then the amount of wasted fitness evaluations will decrease in the critical stages of the optimisation process. However, it has been proved for the standard inversely proportional HMPs from the literature, that no speed-up is achieved compared to the static HMPs already for the simple ONEMAX problem [52]. As an attempt to fix this issue, we propose strategies to properly decrease the mutation potential with respect to the distance to the best seen local optimum. We prove that these modifications allow inversely proportional HMPs coupled with ageing to optimise both easy unimodal and multimodal problems faster than static HMPs. Thus, the power of ageing for escaping local optima is restored. However, such modifications create a new issue: due to their low mutation rates on local optima, inversely proportional HMPs now can only escape them if they are combined with ageing. This conflicts with the main advantage of hypermutations which is their good exploration ability (i.e., the ability of searching different regions of the search space).

Therefore, we suggest some modifications to the static HMPs by evaluating the fitness based on some *success* probability after each bit-flip of a hypermutation, instead of using the deterministic evaluation scheme of traditional FCM. The probability distribution we propose is based on the insights gained in our analysis of static HMP for standard benchmark problems. Since static HMP outperforms EAs when many bits have to be flipped and struggle

when only few bit-flips are necessary, our chosen distribution is parabolic: the probability of evaluating after a bit flip is higher at the beginning and at the end of a hypermutation. We prove that such evaluating scheme makes the operator asymptotically as efficient as EAs in the exploitation phases while maintaining its essential characteristics for escaping from local optima, i.e., it still *hyper*mutates. In addition to showing this for benchmark problems, we further highlight the exploitation ability of the proposed method by proving an upper bound on the expected runtime to achieve a $(1+\varepsilon)$ approximation for any PARTITION instance in an expected time which is by a linear factor smaller than the one we can prove for the traditional static HMP using FCM, and that it finds local optima for the NP-hard VERTEX COVER problem in an expected time which is rigorously a linear factor smaller than that of the static version.

1.2 Structure of the Thesis

This thesis is presented in three main parts, Part I: Background, Part II: Opt-IA and Part III: Fast Opt-IA. The first part contains the necessary background information about the subject, the preliminaries needed for the runtime analysis and a literature review of the current theoretical understanding of AISs. The second part includes the theoretical analysis of the behaviour of the standard Opt-IA on both benchmark problems and combinatorial optimisation problems. Finally, the third part contains our proposed modifications to Opt-IA to overcome the drawbacks highlighted by the analyses in the previous chapters. In the following, the subjects covered in each chapter are introduced briefly.

Part I: Background

Chapter 2 introduces artificial immune systems designed for optimisation applications. We start by reviewing the natural immune system of vertebrates and providing an overview of the clonal selection process of b-cells in detail. Then, the commonly used operators by AISs inspired by clonal selection are introduced. These operators are *cloning*, *hypermutation* and *ageing*. The chapter is concluded with descriptions of the three most popular AISs for optimisation, i.e., Clonalg [23], B-cell [55] and Opt-IA [19]. The algorithms which we will analyse are mainly introduced in detail in this chapter.

After having defined how AISs work, in **Chapter 3** we present the basics needed in the rest of the thesis to theoretically investigate the performance of AISs. To this aim, we first define the mathematical tools which will be used in the analyses. Then, we introduce the standard benchmark functions that we will consider and that are widely used for evaluating

the performance of randomised search heuristics in general. The remainder of this chapter provides an overview of the state-of-the-art of the theoretical understanding of AISs. These results cover hypermutation and ageing operators in isolation and complete AISs.

Part II: Standard Opt-IA

In **Chapter 4**, we analyse the performance of Opt-IA for benchmark functions. We first consider HMP and ageing both in isolation, and afterwards as a complete AIS. The aim is to highlight function characteristics for which Opt-IA and its main components are particularly effective, hence when they may be preferable to standard EAs. Our first result related to HMP is that the use of FCM is essential: we prove that a simple single-trajectory algorithm equipped with hypermutations and no FCM requires exponential expected runtime to optimise any function with an arbitrary polynomial number of optima. From then on we will always consider the operator in conjunction with FCM. Our second main contribution is the introduction of a methodology that allows us to derive upper bounds on the expected runtime of the sophisticated hypermutation operators by analysing much simpler randomised local search (RLS) algorithms with an arbitrary neighbourhood size. Thirdly, our analyses show exponential speed-ups of HMP to escape from local optima compared to the standard bit mutations (SBM) used by EAs. This is achieved, though, at the expense of a linear slow-down compared to EAs for simple unimodal hill-climbing problems. Then, we analyse the ability of ageing to escape from local optima of standard multimodal benchmark functions. We prove that using the operator both with RLS and SBM can make a difference between polynomial and exponential runtimes. We then concentrate on the analysis of the complete Opt-IA algorithm. We first show that the algorithm is efficient for all the problems where HMP in isolation was proven to be efficient. Afterwards, we present a class of functions called HIDDENPATH, where it is necessary to use both ageing and hypermutations in conjunction, hence where the use of Opt-IA in its totality is crucial. We conclude the chapter by pointing out some limitations of the algorithm. In particular, we present a class of functions called HYPERTRAP that is deceptive for Opt-IA while standard EAs optimise it efficiently with overwhelming probability.

In **Chapter 5**, we turn our attention to classical combinatorial optimisation. We perform analyses for PARTITION (also known as Number Partitioning) which is well-known to be NP-hard [54, 2] and for which the performance of RLS and EAs is well understood in the literature [90, 69]. It has been shown that RLS and EAs may get stuck on local optima which lead to a worst case approximation ratio of $4/3$. In order to achieve a $(1+\varepsilon)$ approximation for arbitrary ε, a clever restart strategy has to be put in place. Herein, we first show the power of HMPs and ageing at escaping local optima by proving that each of them solve to optimality

instances that are hard for RLS and SBM. Afterwards, we prove that AISs using HMPs guarantee arbitrarily good solutions of approximation ratio $(1+\varepsilon)$ in $O(n^3)$ expected fitness function evaluations for any constant ε without requiring any restarts. On the other hand, we prove that an AIS with SBM and ageing can efficiently achieve the same approximation ratio in $O(n^2)$ expected fitness function evaluations by automatically restarting the optimisation process when it implicitly detects the absence of progress (i.e., when it is stuck on a local optimum). While for EAs and RLS the restart strategy has to be decided in advance for each desired approximation ratio, the considered AISs efficiently identify the same approximation ratios without requiring such information. To the best of our knowledge, this is the first time that either HMP or ageing have been theoretically analysed for a standard problem from combinatorial optimisation and the first time performance guarantees of any AIS are proven for classical combinatorial optimisation.

Part III: Fast Opt-IA

In **Chapter 6**, we analyse whether inversely proportional HMP operators can speed up AIS in the exploitation phase. The idea behind the inversely proportional hypermutations proposed in the literature is to decrease the potential or the mutation probability the closer solutions are to the optimum. Our hope is that the amount of wasted fitness function evaluations decreases automatically when the optimisation process becomes hard (which leads to high wastage of the potential). Since it has been proven that the inversely proportional hypermutation operators proposed in the literature for both Clonalg and Opt-IA do not succeed at this [52, 93], we propose some modifications which provably allow the decrease of the mutation potential as the distance to the best seen local optima diminishes. We rigorously prove for standard multimodal benchmark functions that this operator, coupled with ageing, enables the algorithm to approximate the ideal behaviour of the inversely proportional hypermutations the more the search space is explored. The operator provides asymptotic speed-ups in the exploitation phases compared to static HMP while it efficiently escapes from local optima coupled with ageing operator. A drawback of inversely proportional HMPs is that low mutation rates on local optima make it difficult for hypermutations to escape, thus reducing their explorative capabilities.

In **Chapter 7**, we consider two different strategies to speed up the static HMP operator in the exploitation phase while keeping its general ability to explore the search space. Instead of evaluating the fitness of the candidate solution after each bit-flip using FCM, the proposed alternatives for FCM evaluate the fitness after each bit-flip only with a probability that follows a parabolic distribution. We first provide a fitness level based mathematical method to derive upper bounds on the expected runtime of the new HMP operators by using the

expected runtime of RLS which is easier to analyse (this method is adapted from the one previously proposed for the standard static HMP). Then, we analyse their performance with and without ageing on benchmark functions and also classical combinatorial optimisation problems. We first show that the proposed HMP operator can achieve $(1+\varepsilon)$ approximations for any PARTITION instance in an expected polynomial time which is by a linear factor smaller than that we have provided for the original static HMP using FCM. Then, we prove a tight bound on the expected runtime for finding feasible solutions for the VERTEX COVER problem which is the same as the expected runtime required by EAs previously proven in the literature. This expected runtime (i.e., $\Theta(n \log n)$) represents a linear speed-up compared to the traditional static HMP operator.

Finally, in **Chapter 8**, we will conclude this thesis by providing a summary of the results and discussing promising future work directions.

1.3 Main Contributions

In this thesis we analyse the behaviour of AISs for combinatorial optimisation to identify characteristics of search environments which they can optimise efficiently compared to EAs. Our main contributions to the understanding of AISs are summarised in the following.

1. The importance of using the FCM mechanism with HMP is proven for the efficient optimisation of essentially any problem of interest.

2. A general mathematical framework for the analysis of the expected runtime of HMP operators is provided. This framework transfers runtime analysis results obtained by the artificial fitness levels method for RLS to valid upper bounds on the expected runtime of AISs using static HMP.

3. The ability of ageing at allowing algorithms to escape from local optima by identifying a gradient leading away from it, is proven. Previously, the ability of ageing at escaping local optima of a sophisticated benchmark function from dynamic optimisation was shown when coupled with randomised local search [74]. We prove ageing's effectiveness in a greater generality for standard multimodal benchmark problems and classical problems from combinatorial optimisation using both randomised local search and standard bit mutations.

4. The first time complexity analysis of a complete Opt-IA, i.e., hypermutations and ageing together, is provided. It is shown that although each operator separately enables

the AIS to escape local optima more efficiently than standard elitist EAs, it is not the case when both are used together as proposed in the standard Opt-IA.

5. The first time performance guarantees for any AIS for a combinatorial optimisation problem are provided. It is proved that both ageing and hypermutations (and the complete Opt-IA) achieve arbitrarily good $(1+\varepsilon)$ approximations for any instance of the NP-hard PARTITION problem in expected polynomial time for any constant ε.

6. Inversely proportional HMP operators are presented that provably decrease the mutation rate with the decrease in distance to the optima and lead to lower expected runtimes compared to static HMP for standard unimodal and multimodal benchmark functions.

7. A new probabilistic evaluation scheme is presented to be used with static HMPs rather than FCM. It is proved that using such evaluation scheme, AISs can hill-climb efficiently while still maintaining their capabilities at escaping from local optima.

1.4 Underlying Publications

The results presented throughout this thesis are drawn from the following publications. The authors are ordered alphabetically and the contribution is divided by the number of the authors.

- Chapter 4, *Analysis for benchmark problems*, is derived from [10, 15]:

    1. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2017). On the Runtime Analysis of the Opt-IA Artificial Immune System. Proceedings of The Genetic and Evolutionary Computation Conference, pages 83–90.
    2. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2019). When Hypermutations and Ageing Enable Artificial Immune Systems to Outperform Evolutionary Algorithms. Theoretical Computer Science, in press.

- Chapter 5, *Analysis for the NP-hard* PARTITION *problem*, is derived from [11, 13]:

    3. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2018). Artificial Immune Systems Can Find Arbitrarily Good Approximations for the NP-hard Partition Problem. Proceedings of 15th International Conference on Parallel Problem Solving from Nature, pages 16–28.

4. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2019). Artificial Immune Systems Can Find Arbitrarily Good Approximations for the NP-hard Number Partitioning Problem. Artificial Intelligence, volume 274, pages 180–196.

- Chapter 6, *Inversely proportional hypermutations*, is derived from [14]:

5. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2019). On Inversely Proportional Hypermutations with Mutation Potential. Proceedings of The Genetic and Evolutionary Computation Conference, pages 215–223.

- Chapter 7, *Hypermutations with probabilistic intermediate evaluations*, is derived from [12]:

6. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani (2018). Fast Artificial Immune Systems. Proceedings of 15th International Conference on Parallel Problem Solving from Nature, pages 67–78.

Part I

Background

Chapter 2

Artificial Immune Systems

2.1 Introduction

Artificial immune system (AIS) models for optimisation are inspired by different immunological processes happening in the natural immune system of vertebrates, in particular, the *clonal selection* process [3]. For this reason, they are often called *clonal selection algorithms*. In this chapter, we describe the basic concepts of both natural and artificial immune systems and define the well-known AISs for optimisation. The state-of-the-art in the theoretical investigations of AIS will be covered in the next chapter.

In Section 2.2, an overview of the natural immune system with main focus on the principles of clonal selection theory is provided. The aim is to highlight the biological concepts which have inspired the design of well-known algorithms in the family of immune inspired randomised search heuristics. In Section 2.3, we describe the structure of an AIS. As the general framework of AISs[1] is the same as that of EAs, in Subsection 2.3.1 we introduce a very general evolutionary process which encompasses both classes of algorithms. Two important frameworks which are the basis for many of the algorithms we consider in the future chapters are discussed in this subsection. In Subsections 2.3.2 to 2.3.4 the commonly used operators in AISs including cloning, hypermutation and ageing are explained. Finally, in Section 2.4, well-known examples of AISs, i.e., Clonalg [23], B-cell [55], and Opt-IA [20] are introduced in detail.

[1] From now on by AIS we mean the AISs designed for optimisation problems, i.e., as a randomised search heuristic.

2.2 The Natural Immune System of Vertebrates

The immune system is a communication network of different cells and tissues, which spread throughout the body, with the goal of protecting the host from pathogens, e.g., bacteria, viruses, and fungi. Generally, the immune system is divided into two parts: *the innate immune system* and *the adaptive immune system* [45]. Whenever the immune system encounters a pathogen, it triggers an immune response by either one or both of these parts. The innate immune system is non-specific (i.e., it reacts to all threats in the same way) and does not change through time, while the adaptive immune system produces specific responses to a particular pathogen and learns from past experiences.

As the first line of defence against pathogens, the innate immune system tries to protect the body against invading organisms using tissues like skin, mucus, tears, and stomach enzymes. Blood cells differentiated from a *myeloid* cell including eosinophils, neutrophils, basophils, macrophages, mass cells, and dendritic cells are part of the first line of defence. As the innate immune system rarely changes in response to life experience, it cannot eliminate the newly developed pathogens, herein, the second line of defence is triggered by the adaptive immune system.

The adaptive immune system includes blood cells called t-cells, b-cells, and natural killer cells which all are differentiated from a lymphoid cell and are called *lymphocytes* in general. After information regarding the invading pathogen is passed to b-cells, an adaptive immune response is generated by the *clonal selection* of b-cells to produce proteins called *antibodies* which are intended to bind to the pathogen and eventually remove it from the body [45]. The transferred information includes the structure of *antigens* of the invading pathogen. Antigens are substances on the surface of a pathogen which can be simple (e.g., containing one single molecule) or complex. Figure 2.1 shows how an antibody binds to an antigen.

2.2.1 Clonal Selection Theory

Proposed by Burnet in the 1950s [3], the clonal selection theory states that antibodies are produced in each host's body only in response to antigens to which the body is exposed. Clonal selection is initiated in a lymph node when a dendritic cell carrying information describing an antigen meets a compatible t-cell. A compatible t-cell can recognise a part of the antigen known as *epitope*. Mostly, a t-cell can recognize only one epitope of an antigen. This t-cell activates a b-cell which is also able to recognize (an epitope of) the same antigen. An effective immune response contains different b-cells producing antibodies against one specific antigen [45].

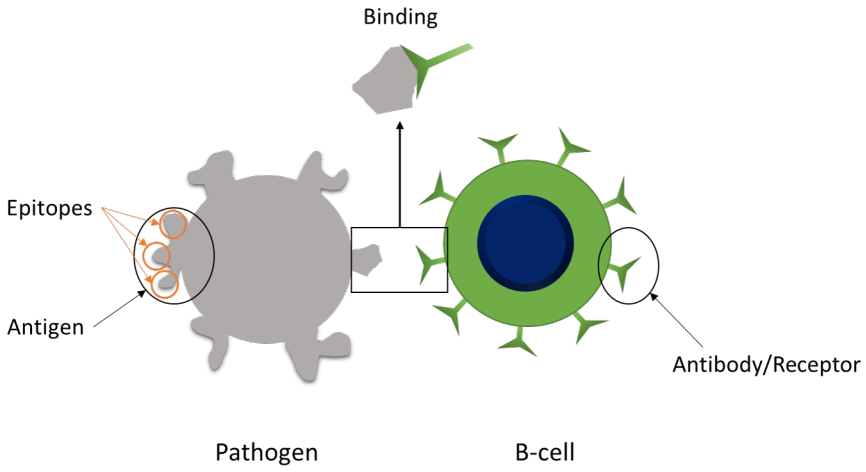

Fig. 2.1 Binding of an antibody and an antigen

Different parts of an antibody are shown in Figure 2.2. Each antibody molecule includes two symmetric twofold axes [45]. It is composed of two identical heavy protein chains and two identical light protein chains. Heavy and light chains both have variable (V) and constant (C) regions. The V regions of a heavy and a light chain combine to form an antigen-binding site, hence both chains contribute to the antigen binding specificity of an antibody molecule [45]. The combination of 320 different light chains with about 11,000 different heavy chains result in almost 10^6 different antigen-binding sites [45].

The clonal selection theory as stated in [3] by Burnet, includes all the important features of the adaptive immune system as is known today. This theory contradicted the *instructive hypothesis* which was the domain idea in the 1940s claiming antigens act as templates for the production of antibodies [45] (i.e., it is opposed to what clonal selection theory states: an antigen gets *selected* by the b-cells that already can recognize it, hence only specific b-cells get activated by an antigen).

The clonal selection procedure may be classified into three phases: (1) clonal expansion, (2) affinity maturation, and (3) programmed death of cells. This classification is made to clarify how different operators in AISs relate to the phases in the clonal selection of b-cells. These phases are described in the following.

Clonal expansion

When an antigen binds to a b-cell, the b-cell becomes activated and generates many identical cells called *clones*. During the next phases, some of these cells differentiate into *plasma b-cells* to secrete antibodies with an identical specificity to the surface receptor of the first b-cell which was activated by the antigen. A small number of these b-cells then turn into

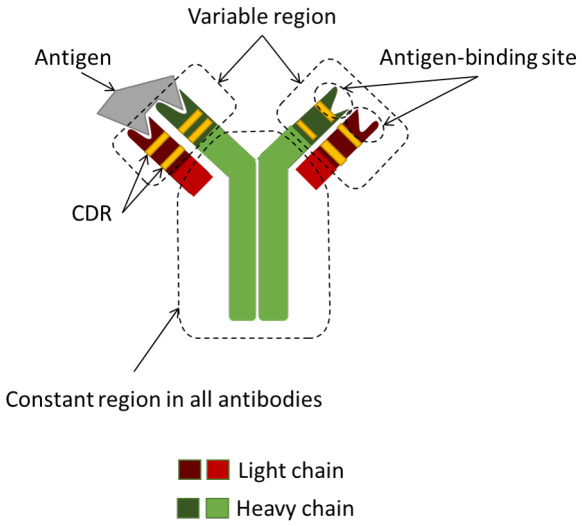

Fig. 2.2 Structure of an antibody

b-memory cells at the end of the clonal selection process to be used for future invasion of the same pathogen. Clonal expansion has inspired the design of cloning operators used in AISs.

Affinity maturation

After the clonal expansion, a process called *affinity maturation* takes place. The aim of this process is to increase the affinity (i.e., binding capacity) of the antibodies of the cloned b-cells with the specific antigen which triggered the clonal selection. This is done by *somatic hypermutation* and then selection of b-cells with the best affinities. Somatic hypermutation is designed to make changes in complementarity-determining regions (CDR) of antibody V parts (Figure 2.2). This process happens in the germinal center of a lymph node and is a "critically timed process" [61]. The term *somatic* means that the mutations introduced in the b-cells of a host cannot be passed on to its offspring.

During the somatic hypermutation, some of the cloned b-cells make point mutations (i.e., a mutation causing a single nucleotide base modification) in the V region of their antibody genes with the aim of increasing the affinity with the antigen. The rate of these point mutations is approximately 10^{-4} to 10^{-3} mutations per base pair[2] per generation (i.e., one new mutation every second generation) [61]. Such a rate is almost six orders of magnitude greater than the average of natural mutations in the structure of a DNA and due to this, it is called *hyper*mutation. These mutations are mostly single base substitutions, and occasionally insertions and deletions (a single change in an amino acid of the V region can increase

[2]Base pairs are the building blocks of a DNA. Each base pair consists of two complimentary nucleotides.

the affinity by a factor of about 10). However, only a small number of these cells are able to increase the affinity. The other mutated cells either do not change the affinity (silent mutations), or they decrease the affinity, break molecule structure, or even create antibodies that react to self molecules. The cells that have a low affinity for the antigen as a result of hypermutation, will most likely not be selected by the antigen and die.

Affinity maturation seems similar to neo-Darwinism which states that *random mutations produce an altered phenotype which leads to survival of the more fit and death of the less fit*. However, this is where the similarity ends. During the past decade, research has revealed that somatic hypermutation is a complicated controlled process which targets genetic changes to specific sites within the V region of the antibodies while protecting the rest of the genome from changes [59, 82]. Therefore, it is not an undirected random process like Darwinian theory of evolution. Some findings have also shown that somatic hypermutation can be cumulative and stops after about 15 generations [61, 83]. Although there are many hypotheses about how the mechanism of somatic hypermutation works, it is not yet fully understood. The variation operators with high mutation rates used in typical AISs are inspired by somatic hypermutation.

Programmed death

After somatic hypermutation is applied, each of the cloned b-cells will be tested by a group of cells in the lymph nodes and spleen. B-cells whose receptors have low affinity with the antigen, are programmed to die with *apoptosis*. Apoptosis or programmed cell death is an internal death program which is activated by the cell itself. Proliferating cells, in particular lymphocytes, frequently undergo apoptosis. The rate of apoptosis is high during the process of development of b-cells and also during the adaptive response [45]. The ageing operators used in AIS are loosely inspired by programmed cell death.

2.3 Structure of an Artificial Immune System and Immune Inspired Operators

Artificial immune system search heuristics are inspired mostly by the processes described in Subsection 2.2.1. In this section, we introduce a general evolutionary scheme originally adopted by EAs which many other randomised search heuristics also fit into (e.g., randomised local search, simulated annealing), including AISs [46]. After introducing this general scheme, we explain in detail the immune inspired operators commonly used by AISs.

EAs iteratively evolve a population of candidate solutions with the aim of creating fitter solutions. As with any reasonable search algorithm, this structure should enable an EA to both *explore* and *exploit* the search space. While exploration is the ability of searching different regions of the search space (i.e., trying unexplored regions), exploitation refers to the ability of searching the neighbourhood of a potentially promising search point (i.e., improving an existing known good solution). The candidate solutions in this structure, which undergo the iterative process, are called *individuals* in an EA and either *b-cells* or *antibodies* in an AIS[3]. The environment that these candidate solution are operating in is called the *search space*. According to the problem at hand, different search spaces are defined. Some frequently used search spaces are $S = \{0,1\}^n$ (a *combinatorial search space*) and $S = \mathbb{R}^n$ (a *continuous search space*). The points in a search space are the candidate solutions. In a combinatorial search space $S = \{0,1\}^n$ which is used throughout this thesis, each search point is a *bit string* $x = x[1]x[2]\cdots x[n] \in \{0,1\}^n$ (i.e., a string of 1-bits and 0-bits). Each search point is assigned with a *fitness value* describing how well it optimises a given problem. This value is determined by a *fitness function* $f : S \to \mathbb{R}$.

2.3.1 An Evolutionary Algorithmic Scheme

After initialising a population of candidate solutions, EAs apply an iterative process of *selection for reproduction*, *variation*, and *selection for replacement* (depicted in Algorithm 2.1) [46]. Every iteration of this loop is called a *generation*. Algorithms differ according to which operators are used in each stage of the loop. As one would prefer this iterative process to end at some point of time, some termination criterion is also needed, e.g., stopping after reaching a predefined number of generations or a fitness above a predefined quality. In this thesis, we will not use a termination criterion and instead evaluate the expected number of fitness function calls (fitness function evaluations) required for an optimal solution (or a given approximation) to be sampled for the first time.

In the following, for each phase, we introduce the operators used in combinatorial search spaces with particular emphasis on those considered throughout this thesis.

Initialisation

An EA starts with an *initialisation* phase. Initialisation refers to the process where the first population of μ candidate solutions is generated before the main loop can start. Usually, the population is initialised with random solutions although problem specific information may

[3]Throughout this thesis, candidate solutions, individuals, search points, b-cells and antibodies are all used equivalently.

Algorithm 2.1: General scheme of an EA

```
1 t := 0;                                          // number of generations
2 initialisation and evaluation of μ individuals;
3 while termination condition is not satisfied do
4     selection for reproduction;      // selecting parents for reproduction
5     variation and evaluation;                        // creating offspring
6     selection for replacement;          // selecting the next generation
7     t := t + 1;
```

also be used if available. For the bit string representation, the initial solutions are generated uniformly at random by setting each bit to 1 independently with probability $1/2$. After initialisation, the fitness of each individual is evaluated by a fitness function.

Selection for reproduction

The goal of the *selection for reproduction* phase is to determine which individuals will reproduce in the current generation, and hence undergo variation. The selected individuals for reproduction are called *parents*. Some popular operators for this phase are *tournament* selection and *ranking* selection [46].

In *generational* EAs, where the new generation completely replaces the previous one, it is crucial that better solutions are selected with high probability than poorer ones [5, 77, 76]. In this thesis, *steady-state* EAs and AISs are considered where *elitism* is used and only a subset of the population is replaced in each generation (i.e., the best individuals are never lost). Hence, it is not crucial that better individuals have a high probability of reproducing. Indeed, it is very common in steady-state algorithms to select parents uniformly at random with replacement.

Variation

After some parents are selected for reproduction, the *variation* process starts. The role of the variation is to create *offspring* by introducing some changes to each parent. While in EAs the variation operators normally aim to make small changes to the parent inspired by natural evolution, in AISs the rate of such changes is high, inspired by the high mutation rates of the b-cells in the natural immune system.

Generally, EAs use two operators to apply variation: *crossover* and *mutation*. The crossover operator imitates sexual reproduction and creates offspring by recombining pieces of two parents. There are various methods of recombination, e.g., single-point crossover for which a random point is chosen in the chromosome of two parents and then the bits at

the right side of this point are swapped resulting in two offspring[4]. Usually, an evolutionary algorithm which uses crossover is called a *genetic algorithm* (GA).

The mutation operator of EAs using a bit string representation usually inverts each bit value independently with mutation probability $p_m = \chi/n$. This is called *standard bit mutation* (SBM). The most common mutation rate is $p_m = 1/n$ where one bit of the bit string is flipped in expectation but many more bits may flip. There is also another mutation operator called k-bit mutation, which inverts the values of exactly k random bits in a bit string. An algorithm that uses such a mutation operator is called *randomised local search* or RLS_k where the choice of k determines the neighbourhood size of the local search operators (usually $k = 1$).

Selection for replacement

After an offspring population is created, a phase called *selection for replacement* starts during which some offspring are chosen to be part of the next generation. For *generational* algorithms, such a selection phase is not applied as the whole population is replaced at the end of each generation by a new population of the same size. The selection mechanism of the *steady-state* algorithms considered throughout this thesis is the *plus selection* which selects the best individuals out of the parent and offspring populations (breaking ties uniformly at random) until a population of the desired size is built for the next generation [46]. This operation is typically referred to as $(\mu + \lambda)$ selection where μ is the size of the parent population and λ that of the offspring population. From now on, we call this stage simply *selection* since this is where the *selection pressure* occurs in the steady-state algorithms we consider, i.e., the best μ individuals are kept while the rest are eliminated.

Basic frameworks

All the algorithms which are either subject to our analyses or are used to compare our results with, fall into the introduced general scheme (i.e., they share the same main phases). The most simple basic framework widely used is the $(1+1)$ framework (Algorithm 2.4) which is obtained from Algorithm 2.1 by setting $\mu = \lambda = 1$. An algorithm using this framework simply starts by initialising one individual at random, then applies mutation (e.g., SBM, local mutation, hypermutation, etc.) to create an offspring. During the selection phase, if the fitness of the offspring is at least as good as that of the parent, it replaces the parent and the next generation starts. In the $(1+1)$ framework, if the mutation operator used is SBM (see Function 2.2), then the simplest and most studied EA, the $(1+1)$ EA, is obtained. If the

[4] As AISs do not apply recombination, the crossover operator is not used in the algorithms analysed in this thesis. For further detail about crossover please see [42].

mutation operator flips exactly k bits (see Function 2.3), then we get the $(1+1)$ RLS$_k$ which is simply called the $(1+1)$ RLS or just RLS for the case of $k=1$. Using this framework, we will analyse typical AIS mutation operators in isolation (as opposed to *in combination* with other immune system operators) to facilitate comparisons with the $(1+1)$ EA and RLS for which the performance is well understood in the literature.

Function 2.2: Standard bit mutation operator of EAs

input : $x \in \{0,1\}^n$;
output : y;

$y := x$;
for $i \in \{1,\cdots,n\}$ **do**
    flip $y[i]$ with probability χ/n (where χ is usually 1);
return y;

Function 2.3: Local mutation operator of RLS$_k$

input : k, $x \in \{0,1\}^n$;
output : y;

$count := 0$;
$y := x$;
repeat
    choose $i \in \{1,\cdots,n\}$ uniformly at random without replacement;
    flip $y[i]$ with probability 1;
    $count := count + 1$;
until $count = k$
return y;

Algorithm 2.4: $(1+1)$ framework

$t := 0$;
initialise $x \in \{0,1\}^n$ uniformly at random;
while *a global optimum is not found* **do**
    mutate x using a mutation operator to produce offspring y;
    if $f(y) \geq f(x)$ **then**
        `// for minimisation problems the above clause is` $f(y) \leq f(x)$`. Additionally, the inequality is strictly greater or less in case of strict selection.`
        $x := y$;
    $t := t + 1$;

While the $(1+1)$ framework is the basis for trajectory-based algorithms (i.e., algorithms with a population of one single individual as opposed to population-based algorithms) we use a $(\mu+1)$ framework for $\mu > 1$ as the basis of many population-based algorithms (see Algorithm 2.5). A $(\mu+1)$ framework starts with a population of size μ, then chooses a random individual as the parent and creates an offspring ($\lambda = 1$) by some variation method. Then, it compares the fitness of this offspring with the least fit individual. If the offspring is at least as fit, then it replaces the least fit individual in the population. Throughout this thesis, when typical AIS operators that rely on populations are considered (such as ageing), the $(\mu+1)$ framework will be used to analyse the effect of the operator in a minimal framework. Some other algorithms we consider in this thesis to compare the performance with are the $(\mu+1)$ EA (Algorithm 2.5 using SBM defined in Function 2.2) which is the most popular steady-state EA, and the $(\mu+1)$ RLS$_k$ (Algorithm 2.5 using the local mutation operator defined in Function 2.3).

Algorithm 2.5: $(\mu+1)$ framework

```
1 t := 0;
2 initialise P = {x_1, ..., x_μ}, a population of μ individuals uniformly at random;
3 while a global optimum is not found do
4     choose x ∈ P uniformly at random;
5     mutate x using a mutation operator to produce offspring y ;
6     choose z ∈ P with the worst fitness, breaking ties uniformly at random;
7     if f(y) ≥ f(z) then
          // for minimisation problems the above clause is f(y) ≤ f(z).
          Additionally, the inequality is strictly greater or less in
          case of strict selection;
8         z := y;
9     t := t + 1;
```

2.3.2 Computational Models of Cloning

Inspired by the clonal expansion phase in the clonal selection process, computational models of cloning create some copies of the parent population. Cloning operators are normally used by AISs in the selection for reproduction phase to generate the clones of the parent population to which the variation operators will be applied to create offspring. Cloning operators from the literature generate clones in either a static manner (i.e., each b-cell gets an equal number of clones) or proportional to the fitness (i.e., the better the fitness of the parent, the greater the number of clones). A static cloning operator makes *dup* copies of

each individual and according to the AIS model may either put them all in one single new population of clones (as in Opt-IA [19]) or in separate clone populations, i.e., one for each b-cell x_i where $i \in \{1, \cdots, \mu\}$ (as in B-cell [55] and Clonalg [23]). Since none of the algorithms in this thesis use fitness proportional cloning, we do not further explain it.

2.3.3 Computational Models of Hypermutation

Usually the only variation operator in AISs is *hypermutation*[5]. Different hypermutation operators have been proposed in the literature. While some use a *mutation probability*, others use a *mutation potential* (a concept similar to k-bit mutation). In the former, every bit is flipped independently with some probability (usually much higher than SBM), while in the latter the operator flips *at most* a number of bits defined by the mutation potential. For both approaches, static and dynamic operators exist. While in the static hypermutations the mutation rate remains the same throughout the run, in the dynamic hypermutation operators the mutation rate usually depends on the fitness of the candidate solution. In the following, we introduce the most popular hypermutation operators.

Inversely fitness proportional hypermutation

The inversely fitness proportional hypermutation operator was originally introduced in the literature to be used by Clonalg [23] which will be fully defined in Subsection 2.4.1. This operator sets the mutation probability to be lower, the higher is the fitness of each b-cell. The probability that each bit is flipped is $p_m = e^{-\rho \cdot v}$ where ρ is the decay parameter and v is the normalised fitness value in $[0,1]$. For maximisation problems, it has been suggested to normalise the fitness by dividing the fitness of a b-cell by the fitness of the best seen individual [23].

Contiguous hypermutation

Inspired by the observation that there exist some specific regions in the gene sequence where hypermutations tend to happen, some hypermutation operators mutate bits in clusters of regions in a bit string. An example of such operators in the literature is the contiguous hypermutation operator [55]. This mutation operator was originally proposed for the B-cell AIS which will be described in Subsection 2.4.2. The contiguous hypermutation operator first chooses a starting point i, and a length l, both at random. Then, it flips each bit in $[i, i+l]$ with probability p_m. Using this operator, the bits at the beginning of the bit string have lower

[5]Note that crossover does not happen in the natural immune system.

probability of being flipped than the bits at the end of the bit string. This makes contiguous hypermutation a *biased* operator as the probability of mutating particular bits is different according to their positions [95]. Versions that wrap around the bit string (i.e., that assume that the bit string is circular) have been proposed to remove the bias towards the beginning of the bit string [50]. Nevertheless, bits that are closer to each other still have higher probability of being mutated than bits that are further away from each other.

It should be noted that although mutating bits in some specific regions of a bit string is inspired by some observations regarding the affinity maturation process, the current related mutation operators are inconvenient to be used in black box optimisation. The inconvenience is due to the need of problem specific knowledge to decide upon the position of bits.

Hypermutation with mutation potential

The last hypermutation operator we introduce uses mutation potentials. Hypermutations with mutation potentials (HMP) were proposed to be used in Opt-IA [19]. A mutation potential (MP) is a function which returns the maximum number of mutation steps that may occur within one hypermutation. Four different mutation potentials are proposed in the literature [19]: *static* MP, *fitness inversely proportional* MP, *fitness proportional* MP, and *hypermacromutation* MP.

The static MP function is $M = cn$ for a constant $0 < c \leq 1$ and is independent of fitness. Hence, the potential is always linear in the length of the bit string n. The other MPs are dynamic so they may return lower potentials than linear. The fitness inversely proportional MP is inversely proportional to the fitness value of the parent, and is obtained by $M(x) = ((1 - \frac{f_{opt}}{f(x)}) \cdot cn)$ where f_{opt} is the minimum value of the fitness function or the best seen (for minimisation problems). The fitness proportional MP is proportional to the fitness value of the parent and is obtained by $M(x) = \min\{(\frac{1}{f(x)-f_{opt}} \cdot cn), cn\}$ where f_{opt} is the minimum value of the fitness function or the best seen and (for minimisation problems). Finally, the hypermacromutation MP uses a rather similar idea to contiguous hypermutations and is independent of both the fitness value and parameter c. It first chooses two integers i and j at random such that $i < j \leq n$, then flips bits only in the range of $[i, j]$ (the mutation potential is $M = j - i + 1$).

A mechanism called *stop at first constructive mutation* (FCM) was introduced to be used together with HMP [19]. Using FCM, the hypermutation operator stops mutating as soon as an improvement is found, hence the fitness is evaluated after each bit-flip. This policy was claimed to be adopted to slow-down (premature) convergence and to improve the exploration capabilities of the HMP operators [19]. Function 2.6 shows how an HMP works.

Function 2.6: Hypermutation with mutation potential with or without FCM

```
input  : M, x ∈ {0,1}^n;
output : y;
y := x;
count := 0;
if FCM is used then
    while count < M do
        count := count + 1;
        choose i ∈ {1, ··· , n} uniformly at random without replacement;
        flip y[i];
        evaluate f(y);
        if f(y) ≥ f(x) then // or f(y) > f(x) depending on how
         constructive mutation is defined
            break;
    return y;
if FCM is not used then
    while count < M do
        count := count + 1;
        choose i ∈ {1, ··· , n} uniformly at random without replacement;
        flip y[i];
    return y;
```

2.3.4 Computational Models of Ageing

Inspired by the natural death of b-cells, some AISs use ageing mechanisms in their search process (e.g., Opt-IA). Some of the most well-known ageing mechanisms from the literature are *static* ageing [19], *stochastic* ageing [19], and *hybrid* ageing [74]. Using any of these ageing operators, each individual is assigned with an $age := 0$ when initialised. At the beginning of each generation, the age of each individual is increased by one. Whenever an offspring is created, it inherits the age of its parent unless it is fitter than the parent, in which case its age is set to 0. The difference among the ageing operators lies in the specific procedure applied to decide on removal of individuals. While static ageing removes each individual deterministically after they reach an age threshold τ, stochastic ageing removes them with a probability p_{die} at the end of each generation (without reaching any specific threshold). Hybrid ageing on the other hand, combines these two mechanisms and removes each individual that has reached an age threshold τ with probability p_{die}. We will later analyse ageing operators in isolation using the minimal $(\mu+1)$ framework introduced in Subsection 2.3.1, in particular in the well-studied $(\mu+1)$ EA$^{\text{ageing}}$ shown in Algorithm 2.7

and (μ+1) RLS$^{\text{ageing}}$ which is identical to $(\mu+1)$ EA$^{\text{ageing}}$ except that it flips one bit rather than using SBM.

Algorithm 2.7: $(\mu+1)$ EA$^{\text{ageing}}$ (for minimisation) [74]

begin
  $t := 0$;
  initialise $P := \{x_1, ..., x_\mu\}$, a population of μ b-cells uniformly at random;
  for *all* $x \in P$ **do**
    $x.age := 0$;
  while *a global optimum is not found* **do**
    `// Ageing (age increment):`
    for *all* $x \in P$ **do**
      $x.age := x.age + 1$;
    for *a randomly chosen* $x \in P$ **do**
      call Function 2.2;
      `// Ageing (age assignment):`
      if $f(y) < f(x)$ **then**
        $y.age := 0$;
      else $y.age := x.age$;
    `// Ageing (removal by static ageing):`
    for *all* $x \in P$ **do**
      if $x.age > \tau$ **then**
        remove x;
    `// Selection:`
    $P := (P \cup y)$;
    if $|P| > \mu$ **then**
      remove the least fit individual, breaking ties uniformly at random;
    if $|P| < \mu$ **then**
      add $(\mu - |P|)$ individuals initialised uniformly at random with $age := 0$;
    $t := t + 1$;

2.4 Artificial Immune Systems for Optimisation

In this section the three most popular AISs for optimisation are described.

2.4.1 Clonalg

Proposed by de Castro and Von Zuben [23], Clonalg is a clonal selection algorithm which views each individual locally, i.e., each individual competes only with its own clones during the selection phase. Clonalg was originally proposed for continuous search spaces to perform machine learning and pattern recognition tasks, however, it has also been adopted for discrete environments and applied for optimisation emphasising multimodal and combinatorial problems.

The pseudo-code of Clonalg is provided in Algorithm 2.8. After initialising a population of μ b-cells x_i for $i \in \{1, \cdots, \mu\}$ and evaluating their fitness, Clonalg then creates a separate clone population for each individual. Each clone population $P_i^{(\text{clo})}$ contains $\text{dup} = (\beta \cdot \mu)$ copies of the parent x_i where β is a user defined parameter. All clone populations will undergo a variation phase in the next step via the inversely proportional hypermutation operator. During selection, the best individual from each hypermutated population $P_i^{(\text{hyp})}$ and x_i will be chosen to be potentially part of the next parent population. Finally, the d individuals with the lowest fitness will be selected to be re-initialised.

2.4.2 B-cell

Proposed by Kelsey and Timmis [55], B-cell is a clonal selection algorithm which uses contiguous somatic hypermutation. Algorithm 2.9 contains a pseudo-code for the B-cell algorithm. After initialising a population of μ b-cells, B-cell creates a separate clone population with size $\text{dup} = \mu$ for each individual. Then, these clone populations will go through a variation phase, which includes two operations of *metadynamic* and contiguous hypermutation. The metadynamic operator is inspired by the receptor editing of those lymphocytes with receptor binding to self-molecules [45]. It is essentially an SBM operator which is applied to a random individual chosen from each clone population. Then, all clones will undergo the contiguous hypermutation process. Finally, the best b-cell is chosen from each hypermutated population and their parent to create the next parent population.

2.4.3 Opt-IA

Proposed by Cutello et al. [19, 20], Opt-IA uses hypermutations with mutation potential (HMP) and ageing operators. This algorithm and its constituent operators are the main focus of this thesis. A pseudo-code for Opt-IA is shown in Algorithm 2.10. This algorithm starts by initialising a population of μ b-cells generated uniformly at random with $age := 0$. In each generation, a new parent population consisting of *dup* copies of each b-cell is

Algorithm 2.8: Clonalg [23]

input : β, d, ρ;
begin
  $t := 0$;
  initialise $P = \{x_1, ..., x_\mu\}$, a population of μ b-cells uniformly at random;
  while *termination condition is not satisfied* **do**
    for $i \in \{1, \cdots, \mu\}$ **do**
      $P_i^{(\text{clo})} := \emptyset$ and $P_i^{(\text{hyp})} := \emptyset$
    `// Cloning:`
    for $i \in \{1, \cdots, \mu\}$ **do**
      make dup $= (\mu\beta)$ copies of x_i and add them to $P_i^{(\text{clo})}$;
    `// Hypermutation:`
    for $i \in \{1, \cdots, \mu\}$ **do**
      for *all* $x \in P_i^{(clo)}$ **do**
        $y := x$;
        flip each bit of y with probability $p_m = e^{-\rho \cdot v}$;
        add y to $P_i^{(\text{hyp})}$;
    `// Selection:`
    $P := \emptyset$;
    for $i \in \{1, \cdots, \mu\}$ **do**
      choose the fittest in $(x_i \cup P_i^{(\text{hyp})})$ and add to P ;
    reinitialise the d individuals with lowest fitness.
    $t := t + 1$;

created (i.e., $|p^{(\text{clo})}| = \text{dup} \cdot \mu$). Afterwards, the population of clones will go through HMP and hypermacromutation independently. In the next step, the selection operator of Opt-IA removes the genotype redundancies (if div $= 1$) and chooses the best μ b-cells out of the parent population and hypermutated b-cells to create the next generation. If needed, new randomly initialised individuals are added by the selection operator to fill up the population. The original selection mechanism of Opt-IA allows no redundancies and randomly generates new b-cells instead of redundant ones. Note that, in contrast with Clonalg and B-Cell, Opt-IA uses one shared clone population for all b-cells.

Algorithm 2.9: B-cell [55]

begin
  $t := 0$;
  initialise $P = \{x_1, ..., x_\mu\}$, a population of μ b-cells uniformly at random;
  while *termination condition is not satisfied* **do**
    for $i \in \{1, \cdots, \mu\}$ **do**
      $P_i^{(\text{clo})} := \emptyset$ and $P_i^{(\text{hyp})} := \emptyset$
    `// Cloning:`
    for $i \in \{1, \cdots, \mu\}$ **do**
      make dup $= \mu$ copies of x_i and add them to $P_i^{(\text{clo})}$;
    `// Metadynamics:`
    for *a randomly chosen* $x \in P$ **do**
      flip every bit with probability $1/n$ (i.e., SBM);
    `// Hypermutation:`
    for $i \in \{1, \cdots, \mu\}$ **do**
      for *all* $x \in P_i^{(clo)}$ **do**
        $y := x$;
        choose $j \in \{0, 1, \cdots, n-1\}$ uniformly at random;
        choose $l \in \{0, 1, \cdots, n\}$;
        if $l > n - j + 1$ **then**
          $l \longleftarrow n - j + 1$;
        for $k \in \{0, \cdots, l\}$ **do**
          flip $y[j + k - 1]$ with probability p_m;
        add y to $P_i^{(\text{hyp})}$;
    `// Selection:`
    $P := \emptyset$;
    for $i \in \{1, \cdots, \mu\}$ **do**
      choose the fittest in $(x_i \cup P_i^{(\text{hyp})})$ and add to P ;
    $t := t + 1$;

2.5 Conclusion

We discussed some important biological background regarding the natural immune system of vertebrates with an emphasis on clonal selection principles. We introduced the main components of an evolutionary algorithm and the immune inspired operators commonly used by AISs. Then, we described how popular AISs work in detail, including the Opt-IA AIS which is the focus of this thesis. In the next chapter, we will define the background needed to

perform runtime analyses of AIS and we will discuss the state-of-the-art in the theoretical understanding of these algorithms.

Algorithm 2.10: Opt-IA* [19]

input : M, div, τ;
begin
  $t := 0$;
  initialise $P := \{x_1, ..., x_\mu\}$, a population of μ b-cells uniformly at random;
  for *all* $x \in P$ **do**
    $x.age := 0$;
  while *termination condition is not satisfied* **do**
    $P^{(\text{clo})} := \emptyset$, $P^{(\text{hyp})} := \emptyset$ and $P^{(\text{macro})} := \emptyset$;
    `// Cloning:`
    for *all* $x \in P$ **do**
      make dup copies of x and add to $P^{(\text{clo})}$;
    `// Ageing (age increment):`
    for *all* $x \in (P^{(clo)} \cup P)$ **do**
      $x.age := x.age + 1$;
    `// Hypermutation:`
    for *all* $x \in P^{(clo)}$ **do**
      call Function 2.6;
      add y to $P^{(\text{hyp})}$;
      `// Ageing (age assignment):`
      if $f(y) > f(x)$ **then** $y.age := 0$;
      else $y.age := x.age$;
    `// Hypermutation:`
    for *all* $x \in P^{(clo)}$ **do**
      create y by applying hypermacromutation (same as the contiguous hypermutation operator of B-cell);
      add y to $P^{(\text{macro})}$;
      `// Ageing (age assignment):`
      if $f(y) > f(x)$ **then**
        $y.age := 0$;
      else $y.age := x.age$;
    `// Ageing (removal for static ageing)`
    for *all* $x \in \left(P \cup P^{(hyp)} \cup P^{(macro)}\right)$ **do**
      if $x.age \geq \tau$ **then** remove x;
    `// Selection:`
    $P := \left(P \cup P^{(\text{hyp})} \cup P^{(\text{macro})}\right)$;
    if $div = 1$ **then** remove any duplicate offspring with the same genotype as individuals in P;
    if $|P| > \mu$ **then** remove the $(|P| - \mu)$ least fit individuals, breaking ties uniformly at random;
    if $|P| < \mu$ **then** add $(\mu - |P|)$ individuals initialised uniformly at random with $age := 0$;
    $t := t + 1$;

Chapter 3

Theory of Artificial Immune Systems

3.1 Introduction

In the previous chapter, we provided an overview of the field of artificial immune systems for optimisation. In this chapter, we discuss the state-of-the-art in the theoretical understanding of the behaviour and performance of AISs.

We start in Section 3.2 by defining the notions of expected runtime of randomised search heuristics and the success probability within a predefined number of steps. A wide range of methodologies are available to perform runtime analyses. In Section 3.3, we define the methodologies that will be used throughout this thesis. As we want to identify problem characteristics that a given search heuristic optimises either well or poorly, we use benchmark functions with significant characteristics to highlight problem features that make optimisation easy or hard for given algorithms. In Section 3.4, we introduce the well-established benchmark problems that will serve this purpose in this thesis. We conclude the chapter by providing an overview of the available theoretical results concerning the performance of AISs. The majority of these results are focused on the commonly used operators of AISs, i.e., hypermutation and ageing operators in isolation (i.e., separately in simple frameworks). These are overviewed in Sections 3.5 and 3.6, respectively. In Section 3.7, we review the available theoretical results concerning complete standard AISs as used in practical applications.

3.2 Runtime Analysis

For any optimisation algorithm, the *optimisation* time T is defined as the first point of time when the optimum is found. For a bit string representation and a maximisation problem,

a global optimum is a search point $x_g \in \{0,1\}^n$ such that $f(x_g) = \max\{f(x) \mid x \in \{0,1\}^n\}$. Since fitness function evaluations are often the most costly operations for a randomised search heuristic, T is defined as the number of times that the fitness function is evaluated and is a random variable (because different runs may very well find the optimum in different times). In runtime analysis, we primarily seek the *expected* value of T, i.e., $E(T)$. To present $E(T)$, we make use of common asymptotic notations (O, o, Ω, ω, and Θ) [4] which allow us to state how the expected runtime grows with respect to the problem size n.

Besides the expected optimisation time, we are often also interested in estimating the success probability. This is the probability that the optimum is found within t steps. We say the optimum is found within t steps with *high probability* if $P(T \leq t) = 1 - o(1)$, and *with overwhelming probability* (w.o.p.) if $P(T \leq t) = 1 - o(1/poly(n))$ where $o(1/poly(n))$ denotes the set of functions asymptotically smaller than the reciprocal of any polynomial function of n. Similarly, *overwhelmingly small probability* denotes any probability in the order of $o(1/poly(n))$. Throughout this thesis, we call an algorithm *inefficient* for optimising a given problem if its (expected, or with some success probability) runtime grows as a super-polynomial function of the problem size n (i.e., $n^{\omega(1)}$), and we call it *efficient* if the (expected, or with some success probability) runtime grows at most as a polynomial function of the problem size n.

3.3 Tools for Runtime Analysis

The analysis of randomised search heuristics borrows most of its tools from probability theory and the theory of stochastic processes [32, 64]. In this section, we introduce the commonly used techniques in our analyses. We will often refer to the theorems and definitions of this section in the rest of this thesis.

3.3.1 Basic Probability Theory

We frequently use the expected value of a random variable X. A definition of expectation of a discrete random variable is provided in Definition 1. An important property of expectation is given in Theorem 1. For X a binomial random variable, the expected value is provided in Theorem 2. If X follows a geometric distribution, then the expected value is often referred to as *the waiting time* (which is frequently used in our analyses) where X often represents the runtime until some event happens, and is given in Theorem 3.

Definition 1 (Expectation [64])**.** *The expectation of a discrete random variable X is defined as $E[X] = \sum_i i \cdot P(X = i)$ where the sum is over all values that X can take.*

Theorem 1 (Linearity of expectation [1])**.** *Let X be a discrete random variable with expectation $E(X)$ and $a_1, \cdots, a_n \in \mathbb{R}$. Then, $E(\sum_i a_i X_i) = \sum_i a_i E(X_i)$.*

Theorem 2 (Expectation of binomial random variables [64])**.** *For X a binomial random variable with probability distribution of $P(X = k) = \binom{n}{k}(1-p)^{n-k}p^k$ on $k = 0, 1, \cdots, n$, $E[X] = np$.*

Theorem 3 (Waiting time argument [64])**.** *Let X be a geometric random variable with probability distribution of $P(X = k) = (1-p)^{k-1}p$ on $k = 1, 2, \cdots$. Then, $E[X] = 1/p$.*

The *union bound* is another basic argument used in probability theory which is commonly applied throughout this thesis. It gives an upper bound for the probability of the union of a set of outcomes by using the sum of their individual probabilities [64]. A formal description is provided below.

Theorem 4 (Union bound [64])**.** *Assume $A_1, \ldots, A_n$ are arbitrary events in a probability space. Then, $P\left(\bigcup_{i=1}^{n} A_i\right) \leq \sum_{i=1}^{n} P(A_i)$.*

We use the *coupon collector problem* in some analyses to derive bounds on the expected time needed until a certain number of bits are chosen at least once by the mutation operator [78]. Assume there are n different types of coupons and at each trial only one coupon is drawn. Knowing that each coupon has the same probability of being drawn, we want to know the expected time to collect all n coupons.

Theorem 5 (Coupon collector problem [25])**.** *The expected time to collect all n coupons is nH_n, where $H_n := \sum_{i=1}^{n} 1/i$ is the nth harmonic number. Since $\ln n < H_n < 1 + \ln n$, the coupon collector needs an expected time of $(1 + o(1))n \ln n$.*

3.3.2 Tail Inequalities

We sometimes want to bound the probability that a random variable deviates from its expectation by a certain value. To do this, tail inequalities are used. These turn the expectation into bounds on the value assumed by the random variables that hold with some probability [70]. We use four different tail inequalities in this thesis. The simplest one is Markov's inequality [1], which gives an upper bound for the probability that a non-negative random variable exceeds some value.

Theorem 6 (Markov's inequality [1])**.** *Let X be a random variable with non-negative values, and $E[X]$ its expectation. For all $t \in \mathbb{R}^+$ it holds that $P(X \geq t) \leq E[X]/t$.*

Given its generality, Markov's inequality is often too weak to yield useful results, but it is a fundamental tool for derivation of more sophisticated tail inequalities.

Markov's inequality, though, can be used iteratively to obtain more useful statement. Let $E(T) = cn\ln n$. Using Markov's inequality, we get $P(T \geq e \cdot cn\ln n) \leq 1/e$. Hence the probability that T exceeds $e \cdot cn\ln n$ is quite high. However, the probability that T exceeds n consecutive phases of length $e \cdot cn\ln n$ is exponentially small: $P(T \geq 2e \cdot cn\ln n) \leq 1/e^2)$, $P(T \geq 3e \cdot cn\ln n) \leq 1/e^3), \cdots, P(T \geq ne \cdot cn\ln n) \leq 1/e^n)$.

The second inequality which is closely related to Markov's inequality, is Chebyshev's inequality which provides much stronger tail bounds by using the variance of the random variables along with the expectation [1].

Theorem 7 (Chebyshev's inequality [1]). *For X a random variable with variance $Var(X) = E((X - E(X))^2)$, it holds that $P(|X - E(X)| \geq \gamma\sqrt{Var(X)}) \leq 1/\gamma^2$ for all $\gamma > 0$.*

Chernoff bounds [26] is another family of tail inequalities which provides us with considerably stronger bounds if the random variable is a sum of the independent Poisson trials.

Theorem 8 (Chernoff bounds). *Let $X_1, \cdots, X_n$ denote independent Poisson trials such that $P(X_i = 1) = p_i$. Let $X = \sum_{i=1}^{n} X_i$. Then:*

1. *For any $\delta > 0$: $P(X \geq (1+\delta)E[X]) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{E[X]}$ [64],*
2. *For any $0 < \delta < 1$: $P(X \leq (1-\delta)E[X]) \leq e^{(-\delta^2 E[X]/2)}$ [64],*
3. *For any $\delta \geq 0$: $P(X \geq (1+\delta)E[X]) \leq e^{(-\min\{\delta,\delta^2\}E[X]/3)}$ [26].*

For the case of sampling without replacement for hypergeometric distributions, we may use another tail bound called Serfling's inequality [86].

Theorem 9 (Serfling's inequality [86]). *Consider a set $C := \{c_1, \ldots, c_n\}$ consisting of n elements, with $c_i \in R$ where c_{min} and c_{max} are the smallest and largest elements in C respectively. Let $\bar{\mu} := (1/n)\sum_{j=1}^{n} c_i$, be the mean of C. Let $1 \leq i \leq k \leq n$ and X_i denote the i th draw without replacement from C and $\bar{X} := (1/k)\sum_{j=1}^{k} X_i$ the sample mean. For $1 \leq k \leq n$, and $\lambda > 0$:*

$$P\left(\sqrt{k}(\bar{X} - \bar{\mu}) \geq \lambda\right) \leq \exp\left(-\frac{2\lambda^2}{(1 - f_k^*)(c_{max} - c_{min})^2}\right),$$

where $f_k^* := \frac{k-1}{n}$.

3.3.3 Artificial Fitness Levels

Artificial fitness levels (AFL) is a method originally developed to derive upper bounds on the expected runtime of EAs without having to track the probability that the algorithm moves from any point in the search space to any other. Instead, the search space is divided into a small number of non-intersecting levels of increasing fitness and only the probability of leaving each level for a better one needs to be estimated. Thus, the analysis is considerably simplified and often tight bounds are even achieved.

Using AFL, we divide the search space into m mutually exclusive partitions $A_1, \cdots, A_m$ such that all the points in A_i have a lower fitness than points which belong to A_j for all $j > i$. The last partition, A_m, only includes the global optima. If p_i is the smallest probability that an individual belonging to A_i mutates to an individual belonging to A_j such that $i < j$, then the expected time to find the optimum is $E(T) \leq \sum_{i=1}^{m-1} 1/p_i$ [78].

3.3.4 Drift Analysis

Drift analysis uses the expected random decrease in the distance to the optimum achieved in one step to make statements about the expected time to reach the global optimum [78, 58]. Assume a given algorithm is at distance d from an optimum and each step takes it closer to the destination in expectation by at least δ (drift). Then, the *additive* drift theorem states that the optimum is reached in at most d/δ expected steps[1]. For any problem at hand, a proper distance function must be defined. This is often trivial if the expected progress of the algorithm remains similar throughout the run. However, if this is not the case, defining a proper distance function for tight bounds on the runtime can be tricky. If the drift is proportional to the current distance to the optimum, then the *multiplicative* drift theorem is usually easier to apply.

Theorem 10 (Multiplicative drift [78])**.** *Let $\{X_t\}_{t\geq}$ be a sequence of random variables taking the values in some set S. Let $g : S \to \{0\} \cup \mathbb{R}_{\geq 1}$ and assume that $g_{max} := \max\{g(x) \mid x \in S\}$ exist. Let $T := min\{t \geq 0 : g(x_t) = 0\}$. If there exists $\delta > 0$ such that $E(g(X_{t+1}) \mid g(X_t)) \leq (1-\delta)g(X_t)$, then $E(T) \leq \frac{1}{\delta}(1+\ln g_{max})$ and for every $c > 0$, $Pr(T > \frac{1}{\delta}(\ln g_{max} + c)) \leq e^{-c}$.*

[1]The same statement holds for lower bounds on the expected runtime: if the drift is at least δ, then the expected runtime is at most d/δ.

3.3.5 Ballot Theorem

The Ballot theorem was first used for analysing hypermutation operators with FCM to derive an upper bound on the probability of picking more 0-bits than 1-bits (or vice versa) in one step [52].

Theorem 11 (Ballot theorem [32])**.** *Suppose that, in a ballot, candidate P scores p votes and candidate Q scores q votes, where $p > q$. The probability that throughout the counting there are always more votes for P than for Q equals $(p-q)/(p+q)$.*

3.3.6 Typical Run Investigations

The typical run investigations method is based on the idea that the global behaviour of processes is often predictable in contrast to the unpredictability of local behaviours. A good example for such predictability is the outcome of tossing a coin [78]. Only for a large number of trials one can give a tight bound on the number of heads (tails) showing up overall with an exponentially small error.

Using typical run investigations, we divide a process into k phases. These phases should be long enough to guarantee the occurrence of an event of interest with probability p_k and its failure with probability $1-p_k$. The last phase leads the algorithm to the global optimum with failure probability of $1-\sum_{i=1}^{k} p_i$. The overall runtime is a sum of the runtime of all phases. This method is normally used together with Chernoff bounds to obtain the failure probability of each phase.

3.3.7 Some Frequently Used Inequalities

The following well-known inequalities are widely used throughout this thesis [64].

$$1+x \leq e^x \tag{3.1}$$

$$\left(1-\frac{1}{n}\right)^n \leq \frac{1}{e} \leq \left(1-\frac{1}{n}\right)^{n-1} \tag{3.2}$$

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{n\cdot e}{k}\right)^k \quad \text{(Binomial coefficients)} \tag{3.3}$$

$$\sum_{i=1}^{n} \frac{1}{i} \leq \ln(n) + O(1) \quad \text{(Harmonic series)} \tag{3.4}$$

3.4 Standard Benchmark Functions

Pseudo-Boolean benchmark functions $f : \{0,1\}^n \to \mathbb{R}$ are widely used to examine the efficiency of search algorithms. Through such examinations, we can highlight problem characteristics that hinder efficient optimisation for a given algorithm or that make an algorithm outperform others. Such understanding on designed benchmark functions also allows researchers to construct general mathematical techniques to simplify the analysis of more complicated algorithms on more sophisticated problems [46]. In this section, the standard benchmark function classes we will use to analyse the algorithms in future chapters are introduced.

Benchmark functions can be generally categorized into two main classes: *unimodal* and *multimodal* functions. A function f is unimodal if for any non-optimal search point x, there is always another point with Hamming distance 1 with better fitness [31]. Otherwise it is multimodal. In Subsections 3.4.1 and 3.4.2, we will respectively introduce the unimodal and multimodal fitness functions we consider in this thesis.

We will plot functions differently depending on the properties they possess. For *unitation* functions, where the function value only depends on the number of 1-bits and not the bit positions, we usually illustrate the function values over the number of 1-bits in the bit string. For non-unitation functions, we normally illustrate the function using the Boolean hypercube. In a Boolean hypercube, which is shown in Figure 3.1, the 2^n bit strings are shown using n levels. Each level i contains all the $\binom{n}{i}$ possible bit strings with exactly i 0-bits. These bit strings are assumed to be ordered in each layer from left to right by the number of consecutive 1-bits at the left side of the bit string. Hence, the left narrow margin shown in the figure, represents bit strings in the form of $1^{n-i}0^i$ and the right margin shows 0^i1^{n-i} [46]. These definitions are needed to understand more complex functions in the upcoming chapters. Later on, we simply use a circle to show this hypercube.

3.4.1 Unimodal Functions

We start by introducing one of the simplest unimodal functions widely used to show whether an algorithm is able to find an optimum by only hill-climbing. This function, called ONEMAX, counts the number of 1-bits in a bit string with 1^n as the global optimum. Two similarly defined functions are ZEROMAX, which maximises the number of 0-bits, and ZEROMIN which is the minimisation version of ZEROMAX. ONEMAX is formally defined in (3.5) and illustrated in Figure 3.2.

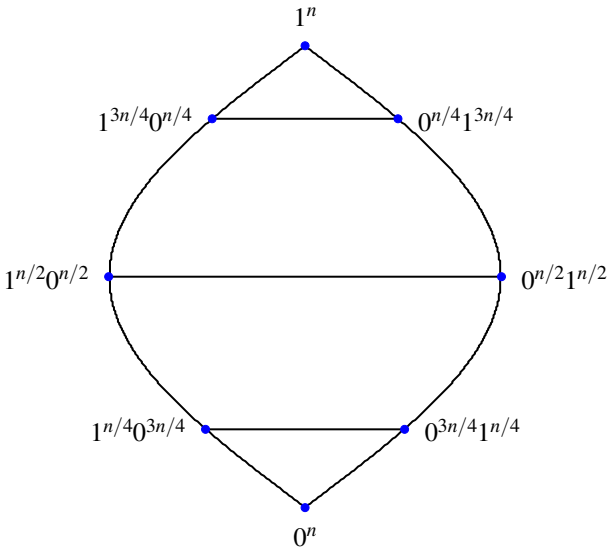

Fig. 3.1 A Boolean hypercube [16]

$$\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i. \tag{3.5}$$

We will refer to the number of 1-bits (ones) in a bit-string using $|x|_1$. Similarly, we use $|x|_0$ for the number of 0-bits (zeros).

ONEMAX has been widely used in the evolutionary computation field to analyse the performance of algorithms. It is unlikely that an algorithm which cannot optimise ONEMAX efficiently would be useful in practical applications. It was shown previously that RLS and the $(1+1)$ EA have expected runtimes of $n\ln n + O(n)$ and $en\ln n + O(n)$, respectively, to optimise ONEMAX [31]. While the first result can be proven to be tight by direct application of coupon collector (Theorem 5, page 34) [78], the second result can be proven via artificial fitness levels [31] and has been shown to be tight as well [87]. For the performance of population-based algorithms, it was shown that the $(\mu+1)$ EA needs in expectation $\Theta(\mu n + n\log n)$ steps to optimise ONEMAX [91]. While in Chapter 4 we show that static HMPs are considerably slower than EAs for optimising this function, in Chapters 6 and 7 we will show how our modified AISs can achieve expected runtimes which match the asymptomatic performance of EAs for hill-climbing. This asymptotic performance is unbeatable by unbiased mutation-only (i.e., unary) black box algorithms [57].

A slightly more complicated unimodal function which we usually use to analyse the hill-climbing performance of randomised search heuristics is LEADINGONES, which counts the consecutive number of 1-bits at the beginning of the bit string. This non-unitation function is formally defined in (3.6).

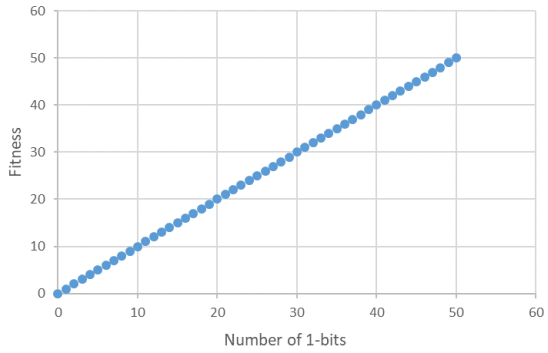

Fig. 3.2 The ONEMAX function of size $n = 50$

$$\text{LEADINGONES}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j. \tag{3.6}$$

LEADINGONES was first introduced to provide an example to show that not all unimodal functions are optimised in expected $O(n \log n)$ steps by EAs [84]. It was shown that the $(1+1)$ EA and RLS both optimise this function in $\Theta(n^2)$ [31, 78]. We will use LEADINGONES as another example in the next chapter, i.e., to show that AISs using static HMP are slower at hill-climbing compared to EAs. We then, in Chapters 6 and 7, suggest some strategies which, without compromising the exploration capabilities of the hypermutation operators, improve their exploitation ability for both ONEMAX and LEADINGONES.

3.4.2 Multimodal Functions

While unimodal functions are commonly used to highlight the exploitation capabilities of algorithms, multimodal benchmark functions are used to analyse their exploration capabilities. The first multimodal function we introduce, is the well-studied CLIFF_d function. It was originally introduced to highlight circumstances where non-elitist EAs may outperform elitist ones [44]. This function is illustrated in Figure 3.3 and is formally defined in (3.7). We sometimes call the local optima of CLIFF, *cliff*.

$$\text{CLIFF}_d(x) = \begin{cases} \sum_{i=1}^{n} x_i & \text{if } \sum_{i=1}^{n} x_i \leq n-d, \\ \sum_{i=1}^{n} x_i - d + 1/2 & \text{otherwise.} \end{cases} \tag{3.7}$$

CLIFF_d consists of a ONEMAX slope with a fitness gap of length d bits between the local optima and the global optimum. The local optima are followed by another ONEMAX slope with a lower fitness than the local optima, leading to the global optimum. Hence,

algorithms that accept inferior solutions (e.g., a move jumping to the bottom of the cliff) can then optimise the following slope and reach the optimum. This effect was originally shown for a non-elitist evolutionary algorithm called $(1,\lambda)$ EA which can optimise the function in approximately $O(n^{25})$ expected fitness function evaluations if the population size λ is neither too large nor too small [44]. Elitist EAs need $\Theta(n^d)$ expected steps to optimise $\textsc{Cliff}_d$ as they do not accept the jump to the bottom of the cliff. A faster expected runtime, but still exponential, was recently shown for the population-genetics-inspired SSWM (Strong Selection Weak Mutation) algorithm with expected runtime of at most $n^d/e^{\Omega(d)}$ [80]. We will show in Chapter 4 how hypermutations lead to exponential speed-ups compared to these algorithms that increase exponentially with the distance d between the cliff and the optimum, and considerably better performance (even quasi-linear) if ageing operators are used.

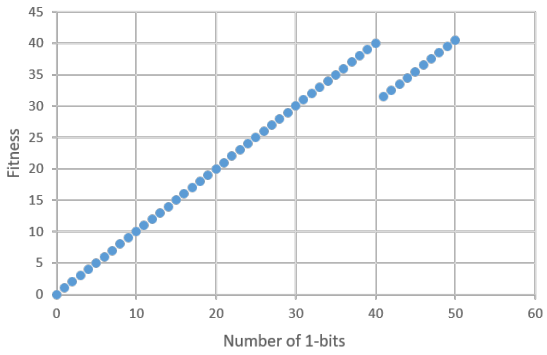

Fig. 3.3 The $\textsc{Cliff}_d$ function of size $n = 50$

A similar function to $\textsc{Cliff}_d$ which we will frequently use is the $\textsc{Jump}_k$ function introduced in [31]. $\textsc{Jump}_k$ consists of a $\textsc{OneMax}$ slope with a gap of length k bits that needs to be overcome for the optimum to be found (Figure 3.4). This function is formally defined in (3.8).

$$\textsc{Jump}_k(x) := \begin{cases} k + \sum_{i=1}^{n} x_i & \text{if } \sum_{i=1}^{n} x_i \leq n-k \\ & \text{or } \sum_{i=1}^{n} x_i = n, \\ n - \sum_{i=1}^{n} x_i & \text{otherwise.} \end{cases} \tag{3.8}$$

Differently from $\textsc{Cliff}$, the slope between the global and local optima leads back to the local ones. Hence, the function is much harder to optimise for non-elitist algorithms. Mutation-based EAs require $\Theta(n^k)$ expected function evaluations to optimise $\textsc{Jump}_k$. Hence their expected runtime grows as the length of the gap increases from super-polynomial to exponential as soon as $k = \omega(1)$. We use the $\textsc{Jump}_k$ benchmark function to show that hypermutations

can achieve considerable speed-ups (an exponential factor of $(e/k)^k$ when the jump is hard to perform) for escaping local optima compared to well-studied EAs.

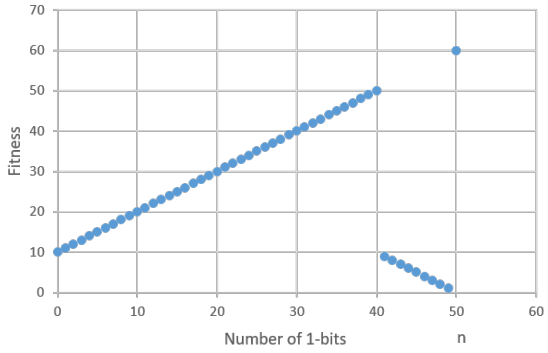

Fig. 3.4 The $\textsc{Jump}_k$ function of size $n = 50$

Another multimodal benchmark function which we will use is the well-studied bi-modal TwoMax function commonly used to evaluate the global exploration capabilities of population-based algorithms, to see, e.g., whether a population can identify both optima of the function. TwoMax is defined in (3.9) and illustrated in Figure 3.5. Standard $(\mu+1)$ EAs cannot identify both optima of TwoMax efficiently. Hence, the function is often used to appreciate the effectiveness of diversity mechanisms for enhancing the global exploration capabilities of populations-based EAs [35, 88, 75, 79]. We will use this function in Chapter 6 to show the effectiveness of inversely proportional HMPs in conjunction with ageing at finding both optima.

$$\textsc{TwoMax}(x) := \max\left\{\sum_{i=1}^{n} x_i,\ n - \sum_{i=1}^{n} x_i\right\}. \tag{3.9}$$

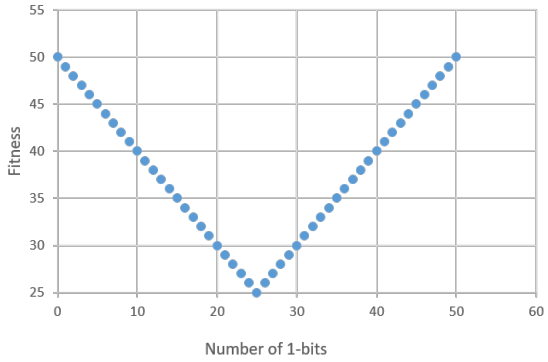

Fig. 3.5 The TwoMax function of size $n = 50$

The last multimodal function we introduce is the TRAP function which is extremely deceptive for EAs [31]. It is formally defined in (3.10) and illustrated in Figure 3.6. TRAP has a ONEMAX slope which directs the search towards the 1^n search point. However, the global optimum is exactly at the opposite side, i.e., 0^n. It was shown that w.o.p., EAs require $n^{\Theta(n)}$ steps to optimise TRAP [31]. We will show that AISs are efficient for a class of TRAP functions where the optimum has $O(1)$ 1-bits.

$$\text{TRAP}(x) := \sum_{i=1}^{n} x_i + (n+1) \cdot \prod_{i=1}^{n} (1 - x_i). \tag{3.10}$$

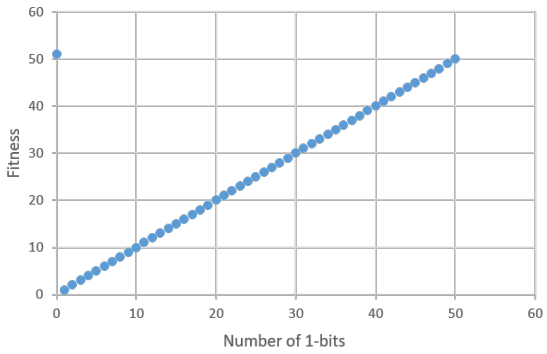

Fig. 3.6 The TRAP function of size $n = 50$

3.5 Runtime Analysis of Hypermutation Operators

As already discussed in Chapter 2, different hypermutation operators have been proposed in the literature inspired by affinity maturation in the natural clonal selection of b-cells. Inversely fitness proportional hypermutations used by Clonalg [23], contiguous hypermutations used by B-cell [55], and hypermutation with mutation potentials used by Opt-IA [20] are the most popular ones. In the following, we review the available runtime analyses results for these classes of hypermutation operators.

3.5.1 Inversely Fitness Proportional Hypermutation Operators

In the inversely fitness proportional hypermutations of Clonalg, the mutation probability $p_m = e^{-\rho \cdot v}$ is a function giving the probability of flipping each bit [23]. This function value depends on the fitness of the candidate solution: the better the fitness, the lower the mutation rate. For different values of ρ (i.e., constant, linear, and logarithmic) it was proved that a $(1+1)$ algorithm applying this mutation operator cannot even optimise the simple ONEMAX

problem in polynomial time with high probability [93]. The investigation was done assuming the optimum is known and using it as the best seen individual to achieve the ideal mutation rate.

However, it was shown that the use of a population may make the operator efficient for ONEMAX. In particular, a $(\mu+1)$ algorithm using the inversely proportional hypermutation operator with $\rho = \ln n$ has an expected runtime of $O(\mu n + n \log n)$ to optimise ONEMAX [94]. However, the reason behind the efficient performance using a population is that the current best seen solution is always mutated with probability $p_m = 1/n$ (just like SBM with default mutation rate). While advantages may still be gained from the high mutation rates used by individuals of lower quality, no such advantages may be noticed from the analysis. In Chapter 6 we will show how inversely proportional hypermutations can be efficient even in single-trajectory $(1+1)$ algorithms if coupled with FCM and show how inversely proportional hypermutations may be efficient also if the best individual uses high mutation rates.

3.5.2 Contiguous Hypermutation Operators

Applied by the B-cell algorithm [55], contiguous hypermutation flips bits in a specific region of the bit string with some probability and leaves the rest of the bit string untouched. It is known as a *biased* operator as the order of the bits affects the outcome of the operator [95]. Applying contiguous hypermutations, the probability of simultaneously only flipping bits that are closer to each other is much higher than flipping bits that are further away.

Contiguous hypermutation operators were analysed for $p_m = 1$ (i.e., all bits in the chosen region will be flipped with probability 1) [50]. Such a mutation operator has been shown to be slow at hill-climbing as it hardly allows any single bit-flips (the probability of flipping exactly one identified bit is $\Theta(1/n^2)$) [50]. Some alternative versions with different approaches to choose the mutation region were also considered in [50]. In particular, it was suggested that the selected region wraps around (i.e., it assumes the bit string is circular) which improves the performance. The efficiency of contiguous hypermutations in a $(1+1)$ algorithm was also proven for TRAP functions and a family of graphs for the MAX-CUT and MINIMUM S-T-CUT problems for which standard EAs are not efficient [92]. Contiguous hypermutation operators used in the complete B-cell AIS have been also analysed for VERTEX COVER and for the longest common subsequent problem. We will illustrate such results in Section 3.7.

The easiest function for this operator was shown to be extremely hard for the SBMs used by EAs (they require $\Omega(n^n)$ steps in expectation) [7, 6]. This function, called MINBLOCK, partitions the search space into $l = \lceil \frac{n}{2} \rceil + 1$ segments of $L_0 \sqcup L_1 \sqcup L_2 \cdots \sqcup L_l = \{0,1\}$ where the fitness of each $x \in L_i$ is $l - i$. The set L_0 only includes $\{1^n\}$ which is the unique global

optimum, and L_1 includes only $\{0^n\}$ which is the second best point. For each $i \in \{2,3,\cdots l\}$, $L_i = \{x \in \{0,1\}^n \mid x \text{ contains } i-1 \text{ different 1-blocks}\}$. The simple $(1+1)$ algorithm using the contiguous hypermutation operator optimises MINBLOCK in $\Theta(n^2)$. The reason for the bad performance of EAs for MINBLOCK is the symmetry of 0-bits and 1-bits, which leads the algorithm to find 0^n with constant probability $1/2$. From then, to find the 1^n point, all bits need to be flipped simultaneously by the EA which requires exponential time.

3.5.3 Hypermutation Operators with Mutation Potential

The available results on HMPs are limited to inversely fitness proportional mutation potentials $M(x) = \lfloor (1 - f_{opt}/v) \cdot cn \rfloor$ [52] (see Section 2.3.3). The analyses were done for a $(1+1)$ algorithm which uses inversely proportional HMP to optimise ZEROMIN$(x) = n + 1 -$ ONEMAX(x). The authors in [52] considered some variants of this mutation potential with and without FCM and also distinguished the cases where the bits are drawn with and without replacement by the mutation operator. The results show that without FCM, the algorithm optimises the ZEROMIN function in $2^{\Omega(n)}$ fitness evaluations in expectation. However, using FCM, the expected runtime is reduced to $\Theta(n^2 \log n)$. The reason is that without FCM, the high mutation rates at the beginning of the search make any progress very difficult. Using FCM, as M is always linear through the run, the algorithm is still by a linear factor slower than the $(1+1)$ EA. It also has been shown in [52] that drawing bits either with or without replacement does not affect the performance significantly.

The analyses in [52] were expanded to another benchmark function where the use of inversely fitness proportional HMP is beneficial compared to local mutations [52]. This fitness function, called SP-TARGET (short path with target) [67], contains a large area of global optima and a path with a Hamming distance of at least $\omega(\log n / \log\log n)$ to this large area which directs the local mutations to a local optimum. It is designed in a way that there is a need for algorithms to make a big jump from this path to the large area of global optima in order to optimise the function. While the $(1+1)$ EA needs super-polynomial time w.o.p. to optimise SP-TARGET as a large number of simultaneous bit-flips are needed to do the jump, the $(1+1)$ algorithm using inversely proportional HMP can flip the required number of bits efficiently and optimises SP-TARGET in at most $O(n^3)$ evaluations in expectation.

3.6 Runtime Analysis of Ageing Operators

As discussed in Subsection 2.3.4, ageing operators assign an age to individuals and then decide how and when to remove them. At initialisation, an age of zero is given to all

initialised solutions. This age is increased by one at each generation. Each newly created individual takes the age of its parent, unless it improves its fitness in which case the age is set to zero. Then, depending on the type of ageing operator used (stochastic, static, or hybrid), old individuals are removed from the population and new randomly initialised individuals are added if needed. The static ageing operator removes individuals deterministically after they reach a predefined threshold τ, while the stochastic ageing operator removes each individual with probability p_{die} at the end of every generation. Hybrid ageing is a combination of the other two operators and removes the individuals which have passed a threshold τ with probability p_{die}.

3.6.1 Static Ageing

The first runtime analysis of the ageing operators investigated the effect of τ (the maximal age) on the performance of the static ageing operator embedded in the $(\mu+1)$ EA [43]. The importance of setting τ has been demonstrated by designing some families of functions for which the runtime changes from polynomial to exponential by changing the τ value or when τ falls out of a specific interval. It was argued that τ should be large enough to allow the individuals to improve, otherwise it would behave as a random search. Additionally, large values of τ reduce the effect of the ageing mechanism (i.e., few individuals are removed by ageing). Some cases to show when small τ values prevent the algorithm from finding local optima which trap the algorithm were also shown.

The difference between the static ageing used by AISs and the ageing mechanism used by EAs which typically assigns age zero to newly created offspring was investigated in [51]. The results show that while the static ageing can escape local optima of example functions by performing restarts when identifying the absence of progress, it does not let the algorithms perform random walks on plateaus (as the points have the same fitness on plateaus, it recognises it as the absence of progress). Although the ageing mechanism in EAs helps them to perform random walks on plateaus (as all offspring are aged zero), it is not helpful for escaping from local optima. In order to overcome the incapability of static ageing in performing random walks, the authors suggested to assign an age of zero to any offspring which is as fit as its parent but does not have the same genotype.

The superiority of static ageing over restart strategies applied by EAs was investigated in [49]. The analyses were done for the $(\mu+1)$ EA applying k-point crossover and static ageing on a function which is designed to show where ageing is helpful but restarts are not. While the $(\mu+1)$ EA applying k-point crossover and static ageing with $\tau=\omega(\mu n\log\mu)$ (the common lower bound for τ presented in [43]) needs $O(\mu\cdot(\tau+n^2+\mu n\log n))$ expected fitness function evaluations to optimise the designed fitness function (described in the following),

any other standard search heuristics without ageing and crossover needs exponential time w.o.p. In the designed fitness function, most of the points are evaluated by ZEROMAX. Directing the search towards 0^n, a path is reached which guides the search to the local optimum. This path is easy to find and to follow by single bit mutations. The global optima are located in an area around the middle of the Boolean hypercube with points in the form of $1^{n/4}0^{n/4}q$ where $q \in \{0,1\}^{n/2}$ has more than $n/12$ 1-bits. By keeping at least two points in the local optimum and removing the rest of the locally optimal points (not necessarily all of them) by ageing, finding a global optimum is possible. While a k-point crossover of the local optimum with a random individual can easily discover the global optima, without it, it is unlikely to optimise the function. When crossover operations do not naturally occur in the natural immune system, in this thesis we will highlight several situations where (hybrid) ageing, without the need of crossover, is very effective.

3.6.2 Stochastic Ageing

The effectiveness of stochastic ageing was studied in the $(\mu+1)$ EA [74]. The authors first showed that stochastic ageing can perform restarts similarly to static ageing. They showed this using a function called LOCALOPT which was previously proved to be efficiently optimised by the $(\mu+1)$ EA using static ageing [51]. For the majority of the LOCALOPT search space, points are evaluated by ZEROMAX. This guides the algorithm towards 0^n which is the start of two different paths. One includes the points in the form of $0^{n-i}1^i$ for $i \in \{1,\cdots,\frac{n}{2}\}$. The fitness improves in this path with increasing rightmost 1-bits, leading the algorithm towards the local optimum with $i = n/2$ 1-bits. The other path includes points in the form of $1^{i\cdot k}0^{n-(i\cdot k)}$. The fitness improves by increasing the leftmost 1-bits in this path, reaching the global optimum at the end. It is much easier to find the first path, hence to discover the local optimum first. After reaching the local optimum, EAs using SBM need exponential time to find the global optimum.

It was also shown that a $(\mu+1)$ EA applying stochastic ageing is as efficient as static ageing for a function which previously was used to show the efficiency of the latter [49]. However, μ should not be large to obtain such result ($\mu \leq n^{\varepsilon}$ for constant ε).

3.6.3 Hybrid Ageing

Among the different variants of ageing, hybrid ageing has been shown to be very efficient at escaping local optima [74]. Arguing that stochastic ageing is only suitable for rather small population sizes (as it otherwise would have a very small probability of performing restarts), the hybrid ageing operator was proposed to suit all reasonable population sizes [74]. This

ageing operator keeps an individual for τ generations and then will give it a probability to die at each following generation. It was shown that hybrid ageing allows successful escape from the local optima of a function from dynamic optimisation called BALANCE in expected polynomial time while static ageing needs at least exponential time w.o.p. [74]. Regarding the parameter setting for p_{die}, it was shown that while for the stochastic ageing p_{die} should be close to 0 to get desirable results, for hybrid ageing p_{die} should be rather large, i.e., $\Theta(1-1/\mu)$ for a $(\mu+1)$ algorithm.

As hybrid ageing has been shown to be the most efficient operator of the three for escaping local optima, we use it in the AISs we will analyse later (i.e., instead of the static and stochastic ageing operators proposed originally for Opt-IA). For the population-based AISs we analyse in the next chapters, at each generation, we have a main population of size μ and a clone population of size $(\mu \cdot \text{dup})$. When we use ageing to escape local optima, we want the entire population to die except for the one individual that escapes. Hence we want the p_{die} to be set in a way that in expectation one individual survives. A choice of $p_{\text{die}} = (1 - \frac{1}{(\mu \cdot \text{dup})+\mu})$ serves our purpose.

3.7 Runtime Analysis of Complete Artificial Immune Systems

The majority of the literature has focused on the immune system operators applied in isolation. B-cell is the only complete artificial immune system that has had its runtime investigated for optimisation problems [47].

3.7.1 The Standard B-cell Algorithm

The first runtime analysis of a complete randomised search heuristic (as used in practice) for a classical combinatorial problem was performed for the B-cell algorithm [47]. The obtained results indicated that B-cell outperforms EAs on an VERTEX COVER instance which is known to be difficult for EAs and RLS [34]. This instance is a bipartite graph with two sets of nodes V_1 and V_2 where the nodes in V_1 are completely connected to all nodes in V_2. The size of the first set is $|V_1| = \varepsilon n$ (i.e., small) and the second set size is $|V_2| = (1-\varepsilon)n$ (i.e., large). It was proved that the $(1+1)$ EA and RLS easily get stuck in the local optimum and only by using restarts they are able to find the minimum cover in expected polynomial time. However, B-cell can achieve expected polynomial runtime without any restart required. After finding a feasible solution (for which either V_1 or V_2 should be selected completely) in expected polynomial time, B-cell finds either a local optimum if all the V_1 vertices are

deselected or the global optimum if V_1 vertices are in the cover. To deselect any of the vertices, it is enough to flip any bit by the metadynamic operator (Algorithm 2.9, page 29) and then choose this vertex and length $l = 1$ during the contiguous hypermutation operation. In expected polynomial time, the local optimum is found and then a mutation flipping all bits is the only acceptable move which, again, happens in expected polynomial time.

In the same paper, it was also shown that B-cell is more efficient than an algorithm using crossover for instances where crossover was proven to be effective previously. As the contiguous hypermutation of B-cell flips bits in a specific region of the bit string, the mapping of the nodes in the given graph and the bits in the bit string has significant effects on the performance. Hence, an ordering heuristic was applied to put nodes sharing common neighbours closely. This heuristic is used by B-cell to obtain all the above result [47].

The second classical combinatorial optimisation problem where B-cell has been proven to be more effective than EAs, is the NP-hard problem of computing a longest common subsequence of a number of strings [53]. It was shown that B-cell optimises some hard instances of the problem for which EAs using mutation and crossover fail. These hard instances contain local optima which are far away from the global optimum. Such good results for B-cell are achievable if they initialise with trivial empty solutions (EAs performance does not depend on the initialisation). If B-cell starts with a population of random solutions, then it is proved that it obtains the same lower bound as EAs.

3.8 Conclusion

In this chapter we first introduced runtime analysis and its main tools from probability theory. We then introduced the standard benchmark functions which will be used to investigate the searching abilities of AISs in future chapters. The state-of-the-art in runtime analyses of AISs for different hypermutation and ageing operators in isolation and the complete AISs were also reviewed. In the next chapter, we will analyse the behaviour of Opt-IA on the presented unimodal and multimodal benchmark functions and on especially designed ones to serve our purposes.

Part II

Standard Opt-IA

Chapter 4

Analysis for Benchmark Problems

4.1 Introduction

In this chapter, the main operators of Opt-IA, i.e., ageing and hypermutations with mutation potential (HMP) are first analysed in isolation and afterwards together as a complete Opt-IA algorithm. The aim is to highlight function characteristics for which Opt-IA and its main components are particularly effective, hence when AIS may be preferable to standard EAs.

In Section 4.2, we present an analysis of static HMP to shed light on its power and limitations in isolation. First, in Subsection 4.2.1 it is proven that the FCM mechanism (stop at first constructive mutation) is essential for HMP to be efficient. Thus, in the rest of this thesis, HMP will always be used in conjunction with FCM. In Subsection 4.2.2, a general mathematical tool is presented that allows us to derive upper bounds on the expected runtime of HMP by simply transferring any result on the expected runtime of randomised local search (RLS) which is established via the standard artificial fitness levels method (see Subsection 3.3.3, page 37). In Subsection 4.2.3, it is shown that the bounds provided by the mathematical tool are tight for two standard hill-climbing benchmark functions, ONEMAX and LEADINGONES. In particular, it is shown that while efficient, HMPs are still a linear factor slower at optimising these hill-climbing problems compared to the SBM (standard bit mutation) operators typically used by EAs. Hence, this disadvantage compared to SBM for optimisation of easy problems is rigorously shown. On the other hand, in Subsection 4.2.4, the superiority of hypermutations at escaping from local optima which are difficult for SBM to escape from, is rigorously shown for the standard JUMP_k (defined in (3.8), page 42) and CLIFF_d (defined in (3.7), page 41) multimodal benchmark functions.

In Section 4.3, we present an analysis of the ageing operator in isolation. Its ability to escape from some hard local optima for traditional EAs using SBM and crossover is highlighted by using the CLIFF function. In particular, in Subsection 4.3.1 it is proven

that ageing applied alongside the local mutations of RLS can solve particularly hard CLIFF functions in an expected runtime of $\Theta(n \log n)$. This is the smallest expected runtime that is achievable by RLS and SBM to optimise any function with unique optimum. In Subsection 4.3.2, a slightly larger upper bound is provided when ageing is applied in combination with SBM. On the other hand, in Subsection 4.3.3 we present an important negative result; If ageing is used alongside HMP, as occurs in the standard Opt-IA, then the resulting algorithm requires exponential time to escape from the local optima of CLIFF w.o.p. Thus, an example is provided where the capabilities of ageing at escaping from local optimum are restricted by hypermutations. The serious implications of this insight become apparent by considering that the two operators were originally designed to work harmonically together in Opt-IA.

In Section 4.4, we move our focus to the complete Opt-IA algorithm. We first consider a simplified version of Opt-IA that does not use any diversity mechanism. After showing that it can efficiently optimise all of the functions considered previously in the chapter with appropriate parameter settings, in Subsections 4.4.2 and 4.4.3 we present example search spaces where the use of Opt-IA as a complete algorithm is either essential or detrimental. We conclude the chapter by showing that similar results are achieved by the standard Opt-IA as in the literature when a selection operator which does not allow genotype diversity is used.

4.1.1 Opt-IA Parameters and Settings

Taking into consideration the theoretical understanding of AIS operators from the literature [52, 74] and also the fact that some operators are not clearly defined in the original Opt-IA paper [19, 20], we consider a slightly modified Opt-IA for the rest of this thesis which is depicted in Algorithm 4.1. It uses one hypermutation operator instead of a combination of two, as we believe it does not benefit from running two hypermutation operators at the same time (at least not for the reasons given in the literature [19, 20]). This will be argued later in this chapter. Additionally, we assume that the hypermutation operator selects bits without replacement as we do not envisage any advantage of selecting with replacement. Jansen and Zarges previously also proved that selecting with replacement does not affect the performance for ONEMAX but makes the analysis very complicated [52]. Also, knowing that hybrid ageing has the best performance for escaping local optima [74], we use hybrid ageing for Opt-IA as the default ageing operator instead of static ageing. Although we consider the constant c in the static MP for this chapter, in the upcoming chapters we would just assume $c = 1$ as in general, the constant factor does not change the runtime asymptotically, and also we would ideally want the potential to be the same size as the bit string so the algorithm can perform the largest possible jumps. Regarding the selection mechanism, although we

consider both cases of removing and keeping redundancies in this chapter, in the rest of this thesis all the algorithms accept redundancies because they speed up the takeover time that makes ageing effective.

4.2 Analysis of Hypermutations with Static Mutation Potentials in Isolation

The aim of this section is to highlight the power and limitations of the static HMP operator in isolation. For this purpose, we embed the operator (Function 2.6 with FCM and $M = cn$) into a $(1+1)$ framework (introduced in Algorithm 2.4, page 21). The resulting algorithm is called $(1+1)$ IA. We will first show in Subsection 4.2.1 that without the use of FCM, hypermutations are inefficient variation operators for virtually any objective function of interest. From there on we will only consider the operator equipped with FCM.

Then we will present a mathematical tool that states that the expected runtime of $(1+1)$ IA is at most a linear factor larger than that obtained for any RLS algorithm using the artificial fitness levels method. Essentially, the mathematical tool allows us to transfer any known upper bound on the expected runtime of any RLS algorithm obtained by AFL method, into a valid upper bound on the expected runtime of the $(1+1)$ IA. If no improvement is found in the first hypermutation operation, then the operator will perform at most cn useless fitness function evaluations before one hypermutation process is concluded. Hence, the $(1+1)$ IA cannot be much slower compared to the standard RLS. We formalise this result in Theorem 13.

In Subsection 4.2.3, we show that the bounds provided by our AFL method are tight for some standard benchmark functions by proving that the $(1+1)$ IA has expected runtimes of $\Theta(n^2 \log n)$ for ONEMAX and $\Theta(n^3)$ for LEADINGONES, respectively, versus the expected $\Theta(n \log n)$ and $\Theta(n^2)$ fitness function evaluations required by RLS [78]. We conclude the section by using the standard benchmark functions JUMP_k and CLIFF_d to show that the $(1+1)$ IA provides exponential speed-ups for multimodal functions with local optima that are generally difficult to escape from.

4.2.1 Stopping at First Constructive Mutation is Essential

We start by highlighting the limitations of static HMPs when FCM is not used. Since $M = cn$ distinct bits have to be flipped at once, the outcome of the hypermutation operator is characterised by a uniform distribution over the set of all solutions which have Hamming distance M to the parent. Since M is linear in n, the size of this set of points is exponentially

Algorithm 4.1: Opt-IA

input : M, div, τ;
begin
  $t := 0$;
  initialise $P = \{x_1, ..., x_\mu\}$, a population of μ b-cells uniformly at random;
  for *all* $x \in P$ **do**
    $x.age := 0$;
  while *a global optimum is not found* **do**
    $P^{(\text{clo})} = \emptyset$ and $P^{(\text{hyp})} = \emptyset$;
    `// Cloning:`
    for *all* $x \in P$ **do**
      make dup copies of x and add to $P^{(\text{clo})}$;
    `// Ageing (age increment):`
    for *all* $x \in (P^{(clo)} \cup P)$ **do**
      $x.age := x.age + 1$;
    `// Hypermutation:`
    for *all* $x \in P^{(clo)}$ **do**
      call Function 2.6;
      add y to $P^{(\text{hyp})}$;
      `// Ageing (age assignment):`
      if $f(y) > f(x)$ **then**
        $y.age := 0$;
      else $y.age := x.age$;
    `// Ageing (removal for hybrid ageing)`
    for *all* $x \in (P \cup P^{(hyp)})$ **do**
      if $x.age > \tau$ **then**
        remove x with probability p_{die};
    `// Selection:`
    $P = (P \cup P^{(\text{hyp})})$;
    if $div = 1$ **then** remove any offspring with the same genotype as individuals in P;
    if $|P| > \mu$ **then** remove the $(|P| - \mu)$ least fit individuals, breaking ties uniformly at random;
    if $|P| < \mu$ **then** add $(\mu - |P|)$ individuals initialised uniformly at random with $age := 0$;
    $t := t + 1$;

large and thus the probability of a particular outcome is exponentially small. In the following theorem, we formalise this limitation.

Theorem 12. *For any function with a polynomial number of optima, the* $(1+1)$ *IA without FCM needs expected exponential time* $e^{\Omega(n)}$ *to find any of the optima.*

Proof. We will first consider the probability that the initial solution is optimal and then the probability of finding an optimal solution in a single step given that the current solution is suboptimal. Note that if $c = 1$, sampling the complementary bit string of an optimal solution at initialisation may allow the hypermutation operator to flip all the bits in the next iteration and find an optimal solution. However, since each optimal solution has a single complementary bit string, the total number of solutions which are either optimal or complementary to an optimal solution is also polynomial in n. Since the initial solution is sampled uniformly at random among 2^n possible bit strings, the probability that one of these polynomially many solutions is sampled is $poly(n)/2^n = 2^{-\Omega(n)}$.

We now analyse the expected time of the last step before an optimal solution is found given that the current solution is neither an optimal solution nor the complementary bit string of an optimal solution. When $c = 1$, the probability of finding the optima is zero since the hypermutation deterministically samples the complementary bit strings of current b-cells. For $c < 1$, we assume that all the optima are at Hamming distance $M = cn$ from the current b-cell. Otherwise, if none of the optima are at Hamming distance M, the probability of reaching an optimum would be zero. Then, given that the number of different points at Hamming distance cn from any point is $\binom{n}{cn}$ and they all are reachable with equal probability, the probability of finding this optimum in the last step for any $c \neq 1$ is $p \leq \frac{poly(n)}{\binom{n}{cn}} \leq \frac{poly(n)}{e^{\Omega(n)}} = e^{-\Omega(n)}$. By a simple waiting time argument (see Theorem 3, page 35), the expected time to reach any optimum is at least $e^{\Omega(n)}$. □

The theorem explains why poor results were achieved in previous experimental work both on benchmark functions and real world applications such as the hard protein folding problem [19, 20]. The authors indeed state that *"With this policy [without FCM], however, and for the problems which are faced in this paper, the implemented IA did not provide good results"* [20]. Theorem 12 shows that this is the case for any objective function of practical interest. Although we are considering static HMPs in this chapter, Theorem 12 can easily be extended to both the inversely proportional HMP considered in [52] and to the proportional HMPs from the literature [19]. From now on we will only consider HMPs coupled with FCM (at least until Chapter 7 when we introduce alternative mechanisms). In the next subsection, we prove that hypermutations are not too slow compared to randomised local searches. Before that, we state and prove the following helper lemma.

Lemma 1. *The probability that the static HMP applied to* $x \in \{0,1\}^n$ *either evaluates a specific* $y \in \{0,1\}^n$ *with Hamming distance* $k \leq cn$ *to* x *(i.e., event* E_y*), or that it stops*

earlier on a constructive mutation (i.e., event E_c) is bounded from below by $\binom{n}{k}^{-1}$ (i.e., $P(E_y \vee E_c) \geq \binom{n}{k}^{-1}$). Moreover, if there are no constructive mutations with Hamming distance smaller than k, then $P(E_y) = \binom{n}{k}^{-1}$.

Proof. Since the bits to be flipped are picked without replacement, each successive bit-flip increases the Hamming distance between the current solution and the original solution by one. The lower bound is based on the fact that the first k bit positions to be flipped have $\binom{n}{k}$ different and equally probable outcomes. Since the only event that can prevent the static hypermutation to evaluate the k-th solution in the sequence is the discovery of a constructive mutation in one of the first $k-1$ evaluations, $P(E_y \vee E_c)$ is at least $\binom{n}{k}^{-1}$. If no such constructive mutation exists (i.e., $P(E_c) = 0$) then, $P(E_y)$ is exactly equal to $\binom{n}{k}^{-1}$. □

4.2.2 Artificial Fitness Levels for Hypermutations

We are ready to show that static HMPs are not much slower compared to any upper bound obtained by applying the artificial fitness levels method (see Subsection 3.3.3, page 37) on the expected runtime of the $(1+1)$ RLS, which flips exactly one bit to produce a new solution and applies non-strict elitist selection. AFL requires a partition of the search space $\mathscr{X}$ into m mutually exclusive sets $\bigcup_{s\in\{1,\ldots,m\}} A_s = \mathscr{X}$ such that $\forall i < j, \quad x \in A_i \wedge y \in A_j \implies f(x) < f(y)$. The expected runtime of a $(1+1)$ algorithm $\mathscr{A}$ with variation operator $\mathrm{HM}(x) : \mathscr{X} \to \mathscr{X}$ to solve any function defined on $\mathscr{X}$ can be bounded from above via AFL by $E(T) \leq \sum_{i=1}^{m} \frac{1}{p_i}$, where $p_i = \min_{x\in A_i} \left\{ P\left(\mathrm{HM}(x) \in \bigcup_{j=i+1}^{m} A_j \right) \right\}$ [78, 46]. Below, we show that hypermutations are not much slower compared to randomised local search operators.

Theorem 13. *Let $E\left(T^{AFL}_{\mathscr{A}}\right)$ be any upper bound on the expected runtime of algorithm $\mathscr{A}$ established via the artificial fitness levels method. Then, $E\left(T_{(1+1)\ IA}\right) \leq cn \cdot E\left(T^{AFL}_{(1+1)\ RLS}\right)$.*

Proof. The upper bound on the expected runtime of RLS to solve any function obtained by applying AFL is $E\left(T^{AFL}_{(1+1)\ RLS}\right) \leq \sum_{i=1}^{m} 1/p_i$, where p_i is s/n when all individuals in level i have at least s Hamming neighbours which belong to a higher fitness level. The probability of mutating to one of the solutions in the first mutation step is the same for static HMP. Such a solution will be evaluated with probability $1/e$. If a solution is not found in the first mutation step, then at most cn fitness function evaluation would be wasted. Since the algorithm is elitist and only accepts individuals of equal or better fitness, each level has to be left only once, independent of whether improvements are achieved by one or more bit-flips. Hence the claim follows. □

4.2.3 Performance Analysis for Unimodal Functions

In the following, the hill-climbing ability of the static HMP is investigated by analysing its performance for ONEMAX and LEADINGONES.

Theorem 14. *The expected runtime of the* $(1+1)$ *IA to optimise* ONEMAX(x) *is* $\Theta(n^2 log\, n)$.

Proof. The upper bound follows from Theorem 13 since it is easy to derive an upper bound of $O(n \log n)$ for RLS using AFL [78]. For the lower bound, we follow the analysis in Theorem 3 of [52] for inversely proportional HMP with FCM to optimise ZEROMIN (minimisation form of ONEMAX, defined in (3.5) in page 40) . The proof there relies on inversely proportional HMP wasting cn function evaluations every time it fails to find an improvement. This is obviously also true for static HMPs (albeit for a different constant c), hence the proof also applies to our algorithm.

By Chernoff bounds (see Theorem 8, page 36), the initial solution has at least $n/3$ 0-bits w.o.p. Using the Ballot theorem (see Theorem 11, page 38), we can achieve an upper bound on the probability of improvement in one hypermutation operation. Here, p is the number of 1-bits and q is the number of 0-bits. The probability of 1-bits being always ahead of 0-bits (i.e., picking more 0-bits than 1-bits to flip) through flipping bits in one hypermutation operation is $1 - \frac{q-p}{p+q} = 2i/n$. Hence, the probability of improvement in one hypermutation operator is at most $2i/n$. Failing to improve, the operator waists cn fitness function evaluations. As at least $n/3$ improvements are needed, we get the upperbound of $\sum_{i=1}^{n/3}(\frac{n}{2i} - 1) \cdot cn = \Omega(n^2 \log n)$ on the expected runtime.

□

We now turn to the LEADINGONES benchmark function (defined by (3.6), page 41), which simply returns the number of consecutive 1-bits from left before the first 0-bit.

The following proof uses the fact that the bits after the leftmost 0-bit are uniformly distributed throughout the run. The proof comes from Lemma 1 in [57] which shows it is the case for all unary unbiased algorithms. As HMP with FCM is unary unbiased (it can be seen as applying at most n unbiased operators RLS_1, RLS_2,..., RLS_n in each generation, where it is not decided in advance whether all these operators are applied or only a subset of them), the lemma holds.

Theorem 15. *The expected runtime of the* $(1+1)$ *IA on* LEADINGONES *is* $\Theta(n^3)$.

Proof. The upper bound is implied by Theorem 13 because AFL gives an $O(n^2)$ runtime for RLS on LEADINGONES [78]. Let $E(f_i)$ be the expected number of fitness function evaluations until an improvement is made, considering that the initial solution has i leading

1-bits, followed by a 0-bit and $n-i-1$ bits which each can be either one or zero with equal probability. Let events E_1, E_2 and E_3 be that the first mutation step flips one of the leading ones (with probability i/n), the first 0-bit (with probability $1/n$) or any of the remaining bits (with probability $(n-i-1)/n$), respectively. If E_1 occurs, then the following mutation steps cannot reach any solution with fitness value i or higher and all cn mutation steps are executed. Since no improvements have been achieved, the remaining expected number of evaluations will be the same as the initial expectation (i.e., $E(f_i \mid E_1) = cn + E(f_i)$). If E_2 occurs, then a new solution with higher fitness value is acquired and the mutation process stops (i.e., $E(f_i \mid E_2) = 1$). However if E_3 occurs, since the number of leading 1-bits in the new solution is i, the hypermutation operator stops without any improvement (i.e., $E(f_i \mid E_3) = 1 + E(f_i)$). According to the law of total expectation: $E(f_i) = \frac{i}{n}(cn + E(f_i)) + \frac{1}{n} \cdot 1 + \frac{n-i-1}{n}(1 + E(f_i))$. Solving this equation for $E(f_i)$, we obtain $E(f_i) = icn + n - i$. Since the expected number of consecutive 1-bits that follow the leftmost 0-bit is less than two [57, 31], the probability of not skipping a level i is $\Omega(1)$. The initial solution on the other hand will have more than $n/2$ leading ones with probability at most $2^{-n/2}$. Thus, we obtain a lower bound of $(1-2^{-n/2}) \sum_{i=n/2}^{n} (f_i) = \Omega(1) \sum_{i=n/2}^{n} (icn + n - i) = \Omega(n^3)$ on the expectation by summing over fitness levels starting from level $i = n/2$. □

4.2.4 Performance Analysis for Multimodal Functions

We now focus on establishing that hypermutations may produce considerable speed-ups if local optima need to be overcome. The JUMP_k function consists of a ONEMAX slope with a gap of length k bits that needs to be overcome for the optimum to be found. Mutation-based EAs require $\Theta(n^k)$ expected function evaluations to optimise JUMP_k. Hence, EAs require increasing runtimes as the length of the gap increases, from super-polynomial to exponential as soon as $k = \omega(1)$. The following theorem shows that hypermutations allow speed-ups by an exponential factor of $(k/e)^k/n$, when the jump is hard to perform. A similar result has been shown for the recently introduced Fast-GA [29].

Theorem 16. *Let $cn > k$. Then, the expected runtime of the $(1+1)$ IA using static HMP to optimise JUMP_k is at most $O(\frac{n^{k+1} \cdot e^k}{k^k})$.*

Proof. The $(1+1)$ IA reaches the fitness level $n-k$ (i.e., local optima) in $O(n^2 \log n)$ steps according to Theorem 14. All local optima have Hamming distance k to the optimum and the probability that static HMP finds the optimum is bounded from below in Lemma 1 by $\binom{n}{k}^{-1}$. Hence, the total expected time to find the optimum is at most $O(n^2 log\, n) + cn \cdot \binom{n}{k} = O\left(\frac{n^{k+1} \cdot e^k}{k^k}\right)$. □

Obviously, static HMPs can jump over large fitness valleys also on functions with other characteristics. For instance, the CLIFF_d function (defined by (3.7), page 41) was originally introduced to show when non-elitist EAs may outperform elitist ones [44].

Corollary 1. *Let $cn > d$. Then, the expected runtime of the $(1+1)$ IA using static HMP to optimise CLIFF_d is $O\left(\frac{n^{d+1} \cdot e^d}{d^d}\right)$.*

The analysis can also be extended to show an equivalent speed-up compared to the $(1+1)$ EA for crossing the fitness valleys of arbitrary length and depth recently introduced in [73].

4.3 Analysis of Ageing in Isolation

It is well-understood that the (hybrid) ageing operator can allow algorithms to escape from local optima. This effect was shown on the BALANCE function from dynamic optimisation for an RLS algorithm embedding a hybrid ageing operator [74]. However, for that specific function, an SBM operator would fail to escape, due to the large basin of attraction of the local optima. In this section we highlight the capabilities of ageing in a more general setting (i.e., the standard CLIFF_d benchmark function) and show that ageing may also be efficient when coupled with SBM. Then, we show an opposite effect for the case when ageing is combined with hypermutations. The reason is that while ageing enables the algorithm to escape from the local optima of CLIFF, the hypermutation operator stops only at the first constructive mutation.

4.3.1 Ageing Combined with Randomised Local Search

Ageing can allow an algorithm to escape from a local optimum if one not locally optimal b-cell is created and survives while all the other b-cells die. For this to happen, it is necessary that all the b-cells are old and have similar age. This is achieved on a local optimum by creating copies of the locally optimal b-cell (the b-cells will inherit the age of their parent). Hence, the ability of a mutation operator to create copies enhances this particular capability of the ageing operator. To this end, we first consider a modified RLS algorithm that with some constant probability p does not flip any bit and implements (hybrid) ageing. This algorithm, uses a $(\mu+1)$ framework and is called $(\mu+1)$ $\text{RLS}_p^{\text{ageing}}$. It is presented in Algorithm 4.2.

Apart from making ageing more effective, this slight modification to the standard RLS considerably simplifies the proofs of the statements we wish to make now. In Subsection 4.4.5, we will generalise the result to an RLS algorithm that does not allow genotype duplicates

Algorithm 4.2: $(\mu+1)$ $\text{RLS}_p^{\text{ageing}}$

```
1  begin
2      t := 0;
3      Initialise P := {x_1, ..., x_μ}, a population of μ b-cells uniformly at random;
4      for each x ∈ P do
5          x.age ∈ P := 0;
6      while a global optimum is not found do
7          for all x ∈ P do
8              x.age := x.age + 1;
9          choose an x ∈ P uniformly at random;
10         y := x;
11         with probability 1/2 < 1 − p ≤ 1 flip one bit of y chosen uniformly at
             random;
12         if f(y) < f(x) then //  for minimisation
13             y.age := 0;
14         else y.age := x.age;
15         P = (P ∪ y);
16         for all x ∈ P do
17             if x.age > τ then
18                 remove x with probability p_die = 1 − 1/μ;
19         if |P| > μ then remove the individual with the worst fitness breaking ties
             uniformly at random;
20         if |P| < μ then add (μ − |P|) individuals initialised uniformly at random
             with age := 0;
21         t := t + 1;
```

as in the standard Opt-IA. Theorem 17 proves a surprising result for the considered $(\mu+1)$ $\text{RLS}_p^{\text{ageing}}$ for $\textsc{Cliff}_d$. Not only is the algorithm very fast, but our upper bound becomes lowest (i.e., $O(n \log n)$) when the function is the most difficult (i.e., when the cliff is located at distance $d = \Theta(n)$ from the optimum). In this setting, the algorithm is asymptotically as fast as any evolutionary algorithm using SBM can be on any function with unique optimum [87]. A similar result has recently been shown also for a hyper-heuristic which switches between elitist and non-elitist selection operators [60].

Before we move to the main results, we will prove a helper lemma for which we will closely follow the arguments of Lemma 5 in [74] (stated below as Lemma 2) to get a similar but more precise bound.

Lemma 2 (Lemma 5 in [74])**.** *Let the whole population be on a local optimum. Then, as long as a better local optimum is not found first and no individual reaches age $\tau+1$, all the individuals will have the same age in expected time $O(\mu^3)$.*

Lemma 3. *Consider the $(\mu+1)$ EA^{ageing} with a population which consists of solutions of equal fitness. Given that no improvement is found and no solutions reach age τ before, the expected time until the population consists of solutions with the same age is at most $144\mu^3$.*

Proof. Let X_t denote the largest number of individuals in the population having the same age. While $X_t < \mu$, the size of X_t increases if one of the X_t individuals is selected as parent, the SBM does not flip any bits and a solution from another age group is removed from the population. The probability of such an event is:

$$P(X_{t+1} = X_t + 1 \mid X_t) \geq \frac{X_t(\mu - X_t)}{\mu(\mu+1)} \cdot \left(1 - \frac{1}{n}\right)^n.$$

Similarly, X_t decreases only if a solution from another age group is cloned and one of the X_t solutions are removed from the population. This event happens with probability:

$$P(X_{t+1} = X_t - 1 \mid X_t) \leq \frac{(\mu - X_t)X_t}{\mu(\mu+1)} \cdot \left(1 - \frac{1}{n}\right)^n \leq P(X_{t+1} = X_t + 1 | X_t).$$

Let a *relevant step* be any iteration where $X_t \neq X_{t+1}$. Then, the number of generations where $X_{t+1} \geq X_t$ in $2\mu^2$ relevant steps has a distribution that stochastically dominates the binomial distribution with parameters $2\mu^2$ and $1/2$. Now, we use Lemma 10.15 in [26] which states that:

Lemma 4 (Lemma 10.15 in [26])**.** *Let X be a binomial random variable with parameters $n \geq 0$ and $p = 1/2$. Then, $P(X \geq \frac{n}{2} + \frac{1}{2}\sqrt{\frac{n}{2}}) \geq \frac{1}{8}$.*

Hence, for any random variable $X \sim Bin(2\mu^2, 1/2)$, $P(X \geq \mu^2 + \frac{1}{2}\sqrt{\mu^2}) \geq \frac{1}{8}$ using Lemma 4. Therefore, the probability that X_t increases at least $\mu^2 + \mu/2$ times in a phase of $2\mu^2$ relevant steps is at least $1/8$. Since $X_t \geq 1$, the expected number of relevant steps until $X_t = \mu$ is at most $(1/8)^{-1}2\mu^2 = 16\mu^2$. The probability of a relevant step is at least:

$$\frac{(\mu - X_t)X_t}{\mu(\mu+1)} \cdot \left(1 - \frac{1}{n}\right)^n \geq \frac{(\mu-1)}{\mu(\mu+1)} \cdot \left(1 - \frac{1}{n}\right)^n \geq \frac{1}{3\mu} \cdot \frac{1}{3} = \frac{1}{9\mu},$$

for any $\mu \geq 2$ and $n \geq 3$. Thus, the expected waiting time for a relevant step is at most 9μ and the total expected time until the population reaches the same age is at most $144\mu^3$. □

We now prove the main result of this section.

Theorem 17. *For $\mu = O(\log n)$, $p < 1$ a constant and $\tau = \Theta(n \log n)$, the $(\mu+1)$ RLS$^{ageing}_p$ optimises* CLIFF$_d$ *in expected time $O\left(\frac{\mu^2 n^3 \log n}{d^2}\right)$ if $d < n/4 - \varepsilon n$ for any constant $0 < \varepsilon < 1/4$.*

Proof. We follow the proof of Theorem 10 in [74] for the $(\mu+1)$ RLS optimising the BALANCE function and adapt the arguments therein to the ONEMAX landscape and to the RLS we use. Given that there are $i < \mu$ individuals with j 1-bits in the population, the probability of creating a new individual with j 1-bits is at least $(i/\mu)p$ because the RLS$_p$ operator creates a copy with a constant probability p. Hence we follow the proof in [74] to show that in $O(\mu n + n \log n)$ (which is $O(n \log n)$ due to the bound on μ) expected steps the population climbs up the ONEMAX slope (i.e., samples a solution with $n-d$ 1-bits) and subsequently the whole population will be taken over by the local optimum. Given that there are already k copies of the best individual in the population, the probability that one is selected for mutation is k/μ and the probability that a new copy is added to the population is at least kp/μ. Thus, in at most $\sum_{k=1}^{\mu} \mu/kp = O(\mu \log \mu)$ generations in expectation after the first local optimum is sampled, the population is taken over by the local optimum. Now we can apply Lemma 3 to show that in expected $O(\mu^3)$ steps the whole population will have the same age. As a result, after another at most $\tau = \Theta(n \log n)$ generations the whole population will reach age τ simultaneously because no improvements may occur unless the optimum is found. Overall, the total expected time until the population consists only of local optima with age τ is at most $O(\mu n + n \log n + \mu \log \mu + \mu^3 + \tau) = O(n \log n)$. Now we calculate the probability that in the next step one individual jumps to the bottom of the cliff and the rest die in the same generation. The first event happens with probability $(1-p)(d/n) = \Omega(d/n)$ (i.e., an offspring solution with $n-d+1$ 1-bits is created by flipping one of the d 0-bits in the parent solution). The probability that the rest of the population dies is $1/\mu \cdot (1-1/\mu)^{\mu}$. We now require that the survivor creates an offspring with higher fitness (which means it will have *age* $= 0$) by flipping one of its 0-bits (i.e., climbing one step up the second slope). This event happens with probability at least $(1-p)(d-1)/(\mu n) = \Omega\left(\frac{d}{\mu n}\right)$ and in the same generation with probability $(1-1/\mu) = \Omega(1)$ the parent of age $\tau+1$ dies due to ageing. Finally, the new solution (i.e., the *safe* individual) takes over the population in $O(\mu \log \mu)$ expected generations by standard arguments. It takes at least $\Omega(n)$ generations before any of the new random individuals has εn more 0-bits than it had after initialisation since FCM terminates after each improvement. Using Markov's inequality (see Theorem 6, page 35), we can show that the probability that the takeover happens after more than $\Omega(n)$ steps is at most $O((\mu \log \mu)/n)$.

The overall probability of this series of consecutive events is $\Omega\left(\frac{d}{n}\right) \cdot \frac{1}{\mu}\left(1-\frac{1}{\mu}\right)^{\mu} \cdot \Omega\left(\frac{d}{\mu n}\right) \cdot (1-\frac{1}{\mu}) \cdot \left(1 - O\left(\frac{\mu \log \mu}{n}\right)\right) = \Omega\left(\frac{d^2}{n^2 \mu^2}\right)$ and the expected number of trials (i.e., climb-ups and restarts) until we get a survivor which is safe at the bottom of the cliff is

$O\left(\frac{n^2\mu^2}{d^2}\right)$. Every time the set of events fails to happen, we wait for another $O(\mu n+n\log n)$ fitness evaluations until the population reaches a configuration where all individuals are locally optimal and have age τ. Once a safe individual has taken over the population, the expected time to find the global optimum will be at most $O(\mu n+n\log n)$. Overall, the total expected time to optimise $\textsc{Cliff}_d$ conditional on the best individual never dying when climbing up the slopes is $E(T_{total}) \leq O(\mu n+n\log n)\cdot O\left(\frac{n^2\mu^2}{d^2}\right)+O(\mu n+n\log n) = O\left(\frac{\mu^2 n^3\log n}{d^2}\right)$. Finally, we consider the probability that the best individual in the population never dies when climbing up the slopes due to ageing. After any higher fitness level is discovered, it takes $O(\mu\log\mu)$ generations in expectation and at most $n^{1/2}$ generations with overwhelming probability until the whole population takes over the level. For the first $n-\log n$ levels, the probability of improving a solution is at least $\Omega(\log n/n)$ and the probability that this improvement does not happen in $\tau-n^{1/2}=\Omega(n\log n)$ generations is at most $(1-\Omega(\log n/n))^{\Omega(n\log n)} = e^{-\Omega(\log^2 n)} = n^{-\Omega(\log n)}$. For the remaining fitness levels, the probability of reaching age τ before improving is similarly $(1-\Omega(1/n))^{\Omega(n\log n)} = e^{-\Omega(\log n)} = n^{-\Omega(1)}$. By the union bound (see Theorem 4, page 35) over all levels, the probability that the best solution is never lost due to ageing is at least $1-o(1)$. We pessimistically assume that the whole optimisation process has to restart if the best individual reaches age τ. However, since at most $1/(1-o(1)) = O(1)$ restarts are necessary in expectation, our bound on the expected runtime holds.

□

4.3.2 Ageing Combined with Standard Bit Mutation

Now we consider the $(\mu+1)$ $\text{EA}^{\text{ageing}}$ which differs from the $(\mu+1)$ $\text{RLS}_p^{\text{ageing}}$ by using SBM with mutation rate $1/n$ instead of flipping exactly one bit. SBM allows the copying of individuals but, since it is a global operator, it can jump back to the local optima from anywhere in the search space with non-zero probability. Nevertheless, the following theorem shows that, for not too large populations, the algorithm is still very efficient when the cliff is at a linear distance from the optimum. Its proof follows similar arguments to those of Theorem 17. The main difference in the analysis is that here we need to show that once the solutions have jumped to the bottom of the cliff, they have a good probability of reaching the optimum before jumping back to the top of the cliff.

Theorem 18. *The $(\mu+1)$ EA^{ageing} optimises $\textsc{Cliff}_d$ in expected $O(n^{1+\varepsilon})$ time if $d = (1/4)(1-c)n$ for some constant $0<c<1$, $\tau=\Theta(n\log n)$ and $\mu=\Theta(1)$, where ε is an arbitrarily small positive constant.*

Proof. The probability that the SBM operator increases the number of 1-bits in a parent solution with $\Omega(n)$ 0-bits is at least $\Omega(n)/(ne) = \Omega(1)$. Following the same arguments as in the proof of Theorem 17 while considering $n/4 > d = \Theta(n)$ and $\mu = O(1)$ we can show that with constant probability and in expected $O(n \log n)$ time the algorithm reaches a configuration where there is a single individual at the bottom of the cliff with $n-d+2$ 1-bits and age zero while the rest of the population consists of solutions which have been randomly sampled in the previous iteration.

We will now show that the newly generated individuals have worse fitness than the solution at the bottom of the cliff when they are initialised. Since $d = (1/4)(1-c)n$, the fitness value of the solutions with more than $n-d$ 1-bits is at least $n-(1/4)(1-c)n-(1/4)(1-c)n = (n/2)(1+c)$. Due to Chernoff bounds (see Theorem 8, page 36), the newly created individuals have less than $(n/2)(1+(c/2)) < (n/2)(1+c)$ 1-bits with overwhelming probability.

Next we prove that for any constant ε, there exist some positive constant c^*, such that the best solution at the bottom of the cliff (the *leading solution*) will be improved consecutively for $c^* \log n$ iterations with probability at least $n^{-\varepsilon}$. The leading solution with i 0-bits is selected for mutation and SBM flips a single 0-bit with probability $p_i = (1/\mu) \cdot (i/n)(1-n^{-1})^{n-1}$. With probability $\prod_{i=n-d+2}^{n-d+1+c^* \log n} p_i$, $c^* \log n$ consecutive improvements occur. We can bound this probability from below by the final improvement probability p_f raised to the power of $c^* \log n$ since the improvement probability is inversely proportional to the number of 0-bits. Considering that $p_f \geq (n-d+1+c^* \log n)/(n\mu e) = \Omega(1)$, we can set $c^* := -\varepsilon \log_{p_f} 2 = \Omega(1)$, which yields $p_f^{c^* \log n} = n^{-\varepsilon}$. Here, we note that immediately after $c^* \log n$ consecutive improvements, all the individuals in the population have at least $n-d+1+c^* \log n-\mu$ 1-bits. More precisely, there will be one and only one individual in the population with j bits for all j in $[n-d+1+c^* \log n-\mu, n-d+2+c^* \log n]$. Since μ is constant and SBM flips at least k bits with probability $n^{-k}\binom{n}{k}$, the probability that a single operation of SBM decreases the number of 1-bits in any individual below $n-d+1$ is in the order of $O(1/(\log n)!)$. Since this probability is not polynomially bounded, with probability at least $1-O(1/n)$ it will not happen in any polynomial number of iterations. After the consecutive improvements occur, it takes $O(n \log n)$ expected steps until the second slope is climbed and the optimum is found.

The asymptotic bound on the runtime is obtained by considering that the algorithm will reach the local optimum n^{ε} times in expectation before the consecutive improvements are observed. Since the time to restart after reaching the local optimum is in the order of $O(n \log n)$, the expected time is $O(n^{1+\varepsilon} \log n)$. As ε is an arbitrarily small constant, the order $O(n^{1+\varepsilon} \log n)$ is equivalent to the order $O(n^{1+\varepsilon'})$. □

4.3.3 Ageing Combined with Hypermutations

In this subsection, we show the undesirable effects obtained when combining static HMPs with ageing for CLIFF_d with linear d. HMPs with FCM, require an increase in the number of 1-bits in the current solution by d at least once before the hypermutation stops. This requirement implies the exponential lower bound on the runtime. Hence, the power of ageing at escaping local optima vanishes. Before stating the main theorem, we introduce a helper lemma which will be used in the proof of Theorem 19. We used Serfling's inequality (see Theorem 9, page 36) for the hypergeometric distribution to obtain the following result, however the proof can be modified to work with standard Chernoff bounds as well.

Lemma 5. *The probability that a solution with at least $(n/2)+|a|+b$ 1-bits, where $-n/2 \leq a \leq n/2$ and $b < n/2$, is sampled at mutation step k, given that the initial solution has $(n/2)+a$ 1-bits, is bounded from above by $e^{-\frac{b^2}{k}}$.*

Proof. For the input bit string x, let the multi-set of weights $C := \{c_i | i \in [n]\}$ be defined as $c_i := (-1)^{x_i}$ (i.e., $c_i = -1$ if $x_i = 1$ and $c_i = 1$ if $x_i = 0$). Thus, for permutation π of bit-flips over $[n]$, the number of 1-bits after the kth mutation step is $\text{ONEMAX}(x) + \sum_{j=1}^{k} c_{\pi_j}$ since flipping the position i implies that the number of 1-bits changes by c_i. Let $\bar{X} := (1/k)\sum_{j=1}^{k} c_{\pi_j}$ be the sample mean, $\bar{\mu} := (1/n)\sum_{j=1}^{n} c_i$ the population mean. The number of 1-bits which incur the weight of -1 when flipped is at least $n/2+a$. Thus, $\bar{\mu} \leq (1/n)\left(-(n/2)-a+(n/2)-a\right) = -2a/n$. To find a solution with at least $(n/2)+|a|+b$ 1-bits it is necessary that the sample mean $\bar{X}$ is at least $(|a|-a+b)/k$. Thus:

$$\begin{aligned}
\bar{X} \geq \frac{|a|-a+b}{k} &\implies \bar{X}-\bar{\mu} \geq \frac{|a|-a+b}{k}+\frac{2a}{n} \\
&\geq \frac{b}{k}+\frac{|a|-a}{k}+\frac{2a}{n} \geq \frac{b}{k} \\
&\implies \sqrt{k}(\bar{X}-\bar{\mu}) \geq \frac{b}{\sqrt{k}}.
\end{aligned}$$

According to Serfling's inequality:

$$\begin{aligned}
P\left(\sqrt{k}(\bar{X}-\bar{\mu}) \geq \frac{b}{\sqrt{k}}\right) &\leq \exp\left(-\frac{2\left(\frac{b}{\sqrt{k}}\right)^2}{\left(1-\frac{k-1}{n}\right)\left(1-(-1)\right)}\right) \\
&\leq \exp\left(-\frac{b^2}{k}\right).
\end{aligned}$$

□

Having established our intermediary lemma, we can now prove the exponential lower bound on the runtime of the $(1+1)$ IA with ageing on the $\textsc{Cliff}_d$ problem. The following theorem holds for the static HMP operator using FCM.

Theorem 19. *The $(1+1)$ IA using static HMP and ageing requires at least $2^{\Omega(n)}$ fitness function evaluations w.o.p. to find the optimum of $\textsc{Cliff}_d$ for $d = (1-c)n/4$, where $0 < c < 1$ is a constant.*

Proof. The number of 1-bits in any initial solution is between $(n/2) - n^{3/5}$ and $(n/2) + n^{3/5}$ w.o.p. due to Chernoff bounds (see Theorem 8, page 36). According to Lemma 5, the probability that an offspring with more than $a + n/10$ 1-bits is mutated from a parent with $a \geq (n/2) - n^{3/5}$ 1-bits is in the order of $2^{-\Omega(n)}$ using the union bound (see Theorem 4, page 35). Therefore, w.o.p. we will not observe that a solution with less than $n - d - (n/10)$ 1-bits having an offspring with more than $n - d$ 1-bits. However, solutions with b 1-bits such that $n - d - (n/10) \leq b \leq n - d$, have higher fitness than post-cliff solutions with less than $b + d$ 1-bits. Thus, according to Lemma 5, the probability that an acceptable solution is obtained is in the order of $2^{-\Omega(d)} = 2^{-\Omega(n)}$ using the union bound over n mutation steps. □

4.4 Analysis of a Complete Opt-IA

After having analysed the operators separately, in this section we consider a complete Opt-IA. The considered Opt-IA (Algorithm 4.1), uses static HMP coupled with FCM as variation operator, hybrid ageing and standard $(\mu + \lambda)$ selection which allows genotype duplicates. Also, a mutation is considered constructive if it results in creating an equally fit solution or a better one.

In this section, we first show that Opt-IA is efficient for all the functions considered previously in this chapter. Then, in Subsection 4.4.2 we present a problem where the use of the whole Opt-IA is crucial. In Subsection 4.4.3, we show limitations of Opt-IA by presenting a class of functions where standard EAs are efficient while Opt-IA is not w.o.p. We conclude the section with Subsection 4.4.4 where we present an analysis for $\textsc{Trap}$ functions which disproves previous claims in the literature about Opt-IA's behaviour on this class of problems.

The following theorem shows the runtime needed by Opt-IA to optimise the benchmark functions considered previously in this chapter. The theorem uses that the ageing parameter τ is set large enough such that no individuals die with high probability before the optima are found.

Theorem 20. *Let τ be large enough. Then the following upper bounds on the expected runtime of Opt-IA hold:*

$E(T_{\text{ONEMAX}}) = O\left(\mu \cdot dup \cdot n^2 \log n\right)$ *for* $\tau = \Omega(n^2)$,
$E(T_{\text{LEADINGONES}}) = O\left(\mu \cdot dup \cdot n^3\right)$ *for* $\tau = \Omega(n^2)$,
$E(T_{\text{JUMP}_k}) = O\left(\mu \cdot dup \cdot \frac{n^{k+1} \cdot e^k}{k^k}\right)$ *for* $\tau = \Omega(n^{k+1})$,
and $E(T_{\text{CLIFF}_d}) = O\left(\mu \cdot dup \cdot \frac{n^{d+1} \cdot e^d}{d^d}\right)$ *for* $\tau = \Omega(n^{d+1})$.

Proof. The claims use that if τ is large enough (i.e., $\Omega(n^2)$ for ONEMAX and LEADINGONES, $\Omega(n^{k+1})$ for JUMP_k and $\Omega(n^{d+1})$ for CLIFF_d), then with probability $1-o(1)$ the current best solution will never reach age τ and die due to ageing before an improvement is found. For the Opt-IA to lose the current best solution due to ageing, it is necessary that the best solution is not improved for τ generations consecutively. If the improvement probability for any non-optimal solution is at least p_{min} and if the age τ is set to be $p_{min}^{-1}n$, then the probability that a solution will reach age τ before creating an offspring with higher fitness is at most $(1-p_{min})^{\tau} \leq e^{-n}$. By the union bound, it is also exponentially unlikely that this occurs in a polynomial number of fitness levels that need to be traversed before reaching the optimum. Since the suggested τ for each function is larger than the corresponding $p_{min}^{-1}n$, the upper bounds of the $(1+1)$ IA (which does not implement ageing) for ONEMAX, LEADINGONES, JUMP_k and CLIFF_d are valid for Opt-IA when multiplied by $\mu \cdot \text{dup}$ to take into account the population and clones. □

4.4.1 A Tight Bound for ONEMAX

Theorem 21 shows that the presented upper bound in Theorem 20 for ONEMAX is actually tight for sub-logarithmic population and clone sizes (i.e., $\mu = o(\log n)$ and $\text{dup} = o(\log n)$).

Theorem 21. *Opt-IA using static HMP needs at least* $cn^2(\frac{\log(n/3)}{2} - \frac{\mu \cdot dup}{3})$ *expected fitness function evaluations for mutation potential cn to optimise* ONEMAX.

Proof. By Chernoff bounds (see Theorem 8, page 36), any initial individual has at least $i = n/3$ 0-bits with overwhelming probability $1 - e^{-\Omega(n)}$. Hence, using union bound (see Theorem 4, page 35), with probability at least $1 - \mu e^{-\Omega(n)}$ all individuals have at least $n/3$ 0-bits.

To calculate the probability of an improvement (i.e., flipping equal or more 0-bits than 1-bits during one mutation operation), we use the Ballot theorem (see Theorem 11, page 38) in a similar way to [52]. Considering the number of 0-bits as $i = q$ and the number of 1-bits as $n - i = p$, the probability of an improvement is at most $1 - (p-q)/(p+q) = 1 - (n-2i)/n = 2i/n$ according to the Ballot theorem[1]. Hence, the probability that at

[1] Like in [52] we consider that using $cn < n$ mutations, the probability can only be lower than the result stated in the Ballot theorem.

least one out of $\text{dup} \cdot \mu$ individuals succeeds is $P \leq \text{dup} \cdot \mu 2i/n$ by the union bound. We optimistically assume that the rest of the individuals also improve their fitness after such event happens. Recall that the mutation operator wastes cn fitness function evaluations every time it does not improve the fitness. Therefore, the expected time to see an improvement is $E(T_{\text{improve}}) \geq \left(\frac{n}{\text{dup}\cdot 2\mu i} - 1\right) \cdot \mu cn \cdot \text{dup}$. Since the mutation operator stops at the first constructive mutation (i.e., when the number of 1-bits is increased by one), it is necessary to improve at least $n/3$ times. So the total expected time to optimise ONEMAX is $E(T_{\text{total}}) \geq (1-\mu e^{-\Omega(n)}) \sum_{i=1}^{n/3} E(T_{\text{improve}}) = cn^2(\frac{\log n/3}{2} - \frac{\mu \cdot \text{dup}}{3})$.

If individuals were to be removed because of ageing, then the new randomly generated individuals that replace them will have to improve at least $n/3$ times all over again w.o.p. Hence, the runtime may only increase in such an event. □

4.4.2 Where Using Both Ageing and Hypermutations is Essential

In this section, we design the function HIDDENPATH to illustrate a problem class where the use of static hypermutation and ageing together is crucial. When either of these two characteristic operators of Opt-IA is not used, we will prove that the expected runtime is at least super-polynomial. HIDDENPATH: $\{0,1\}^n \to \mathbb{R}$ can be described by a series of modifications to the well-know ZEROMAX function (similar to ONEMAX except that it counts the number of 0-bits) . The distinguishing solutions are those with five 0-bits and those with $n-1$ 0-bits, along with $\log n - 3$ solutions of the form $1^{n-k}0^k$ for $5 \leq k \leq \log n + 1$ (called SP points). The solutions with exactly $n-1$ 0-bits constitute the local optima (LOCALOPT) of HIDDENPATH , and the solutions with exactly five 0-bits form a gradient with fitness increasing with more 0-bits in the rightmost five bit positions. Given that $|x|_0$ and $|x|_1$ respectively denote the number of 0-bits and 1-bits in a bit string x, HIDDENPATH is formally defined as in Definition 2. HIDDENPATH is illustrated in Figure 4.1 where OPT shows the global optimum and the fitness increases with darker shades of gray.

Definition 2. *Given the definitions of* SP *and* LOCALOPT *as above, for any positive constant* $\varepsilon < 1$*, the* HIDDENPATH *function is defined for all* $x \in \{0,1\}^n$ *by:*

$$\text{HIDDENPATH}(x) = \begin{cases} n - \varepsilon + \frac{\sum_{i=n-4}^{n}(1-x_i)}{n} & \text{if } |x|_0 = 5 \text{ and } x \neq 1^{n-5}0^5, \\ 0 & \text{if } |x|_0 < 5 \text{ or } |x|_0 = n, \\ n - \varepsilon + \varepsilon k/\log n & \text{if } x \in \text{SP} := \{x \mid x = 1^{n-k}0^k \text{ and } 5 \leq k \leq \log n + 1\}, \\ n & \text{if } x \in \text{LOCALOPT} := \{x \mid |x|_0 = n-1\}, \\ |x|_0 & \text{otherwise}. \end{cases}$$

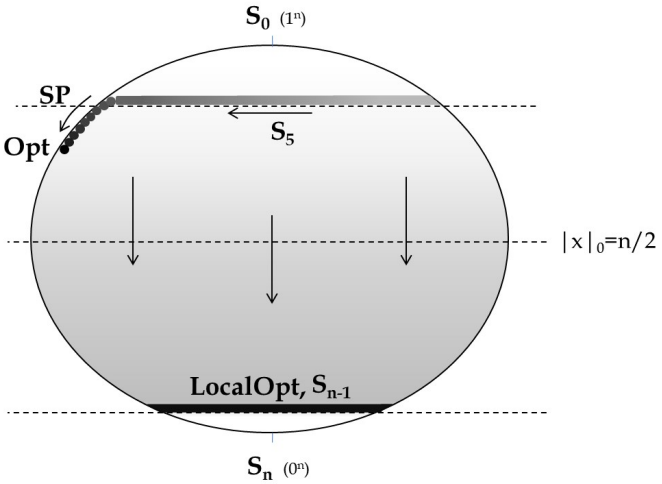

Fig. 4.1 The HIDDENPATH function

Since the all 0-bits string returns fitness value zero, there is a drift towards solutions with $n-1$ 0-bits while the global optimum is the $1^{n-\log n-1}0^{\log n+1}$ bit string. The solutions with exactly five 0-bits work as a net that stops any static hypermutation that has an input solution with less than five 0-bits. The path to the global optimum consists of $\log n-3$ Hamming neighbours and the first solution on this path has five 0-bits.

Theorem 22. *For $c=1$, $dup=1$, $\mu=O(\log n)$ and $\tau=\Omega(n^2\log n)$, Opt-IA using static HMP needs expected $O(\tau\mu n+\mu n^{7/2})$ fitness function evaluations to optimise* HIDDENPATH.

Proof. For convenience we will call any solution with i 0-bits (except the SP solutions) an S_i solution. After $O(n\log n)$ generations in expectation, an S_{n-1} solution is found by optimising ZEROMAX. Assuming the global optimum is not found first, consider the generation when an S_{n-1} solution is found for the first time. Another S_{n-1} solution is created and accepted by Opt-IA with probability at least $1/n$ since it is sufficient to flip the single 1-bit in the first mutation step and any 0-bit in the second step will be flipped next with probability one. Thus, S_{n-1} solutions take over the population in expected $O(n\mu)$ generations. Since, apart from the optimum, no other solution has higher fitness than S_{n-1} solutions, the population consists only of S_{n-1} solutions after the takeover occurs. A solution reaches age τ before the takeover only with probability $2^{\Omega(\sqrt{n})}$, due to Markov's inequality applied iteratively (see Theorem 6, page 35) for $\Omega(\sqrt{n})$ consecutive phases of length $\Theta(n\mu)$. We now bound the expected time until the entire population has the same age. Considering that the probability of creating another S_{n-1} solution is $\Theta(1/n)$, the probability of creating two copies in one single generation is $\binom{\mu}{2}O\left(\frac{1}{n^2}\right)=O\left(\frac{\log^2 n}{n^2}\right)$. With probability $\omega(1)$, this event does not happen in $o(n^2/\log^2 n)$ generations by the union bound (see Theorem 4,

page 35). Conditional on that at most one additional S_{n-1} solution is created in every generation, we can follow a similar argument as in the proof of Theorem 17. Hence, we can show that in expected $O(\mu^3 n)$ iterations after the takeover, the whole population reaches the same age. When the population of S_{n-1} solutions with the same age reaches age τ, with probability $1/\mu \cdot (1-(1/2\mu))^{2\mu-1} = \Theta(1/\mu)$ a single new clone survives while the rest of the population dies. With probability $1-O(1/n)$ the survived clone has hypermutated all n bits (i.e., the survived clone is an S_1 solution). In the following generation, the population consists of an S_1 solution and $\mu-1$ randomly sampled solutions. With probability 1, the S_1 solution produces an S_5 solution via hypermutation. On the other hand, with overwhelming probability the randomly sampled solutions still have fitness value $n/2 \pm O(\sqrt{n})$, hence the S_1 solution is removed from the population while the S_5 b-cell is kept. Overall, after $\Theta(\mu)+o(1)$ expected restarts an S_5 solution will be found in a total expected runtime of $O(\mu \cdot (n \log n + \mu n + \mu^3 n + \tau + 1)) = O(\mu\tau)$ generations. We momentarily ignore the event that the S_5 solution reaches an S_{n-1} point via hypermutation.

Now, the population consists of a single S_5 solution and $\mu-1$ solutions with at most $n/2+O(\sqrt{n})$ 0-bits. We want to bound the expected time until S_5 solutions take over the population conditional on no S_{n-1} solutions being created. Clones of any S_5 solutions are also S_5 solutions after hypermutation if one of the first two bits to be flipped is a 0 and the other is a 1, which happens with $O(1/n)$ probability. Moreover, if the outcome of hypermutation is neither an S_5, an SP nor an S_{n-1} solution, then it is an S_{n-5} solution since all n bit-flips will have been executed. Since S_{n-5} solutions have higher fitness value than the randomly sampled solutions, they stay in the population. In the subsequent generation, if hypermutation does not improve an S_{n-5} solution (which happens with probability $O(1/n)$), it executes n bit-flips to create yet another S_5 solution unless the SP path is found.

This feedback causes the number of S_{n-5} and S_5 solutions to double in each generation with probability $\omega(1)$ until they collectively take over the population in $O(\log \mu)$ generations in expectation. Then, with probability $\omega(1)$ all the S_{n-5} solutions produce an S_5 solution via hypermutation and consequently the population consists only of S_5 solutions. Since the takeover happens in expected $O(\log \mu)$ generations, the probability that it fails to complete in $O(\log \mu)$ generations for consecutive $\Omega(\sqrt{n})$ times is exponentially small. Hence, in $O(\sqrt{n} \log \mu)$ generations w.o.p. (by applying Markov's inequality iteratively) the entire population are S_5 solutions conditional on S_{n-4} solutions not being created before. Since the probability that the static hypermutation creates an S_{n-4} solution from an S_{n-5} solution is less than $(4/n) \cdot \mu$, and the probability that it happens in $O(\sqrt{n} \log \mu)$ generations is less than $(4/n) \cdot \mu \cdot O(\sqrt{n} \log \mu) = o(1)$, the takeover occurs with high probability, i.e., $1-o(1)$.

After the whole population consists of only S_5 solutions, except for the global optimum and the local optima, only other points on the gradient would have higher fitness. The probability of improving on the gradient is at least $1/n^2$ which is the probability of choosing two specific bits to be flipped (a 0-bit and a 1-bit) leading towards the b-cell with the best fitness on the gradient. Considering that the total number of improvements on the gradient is at most 5, in $O(n^2 \cdot 5) = O(n^2)$ generations in expectation the first point of SP will be found. Now we consider the probability of jumping back to the local optima from the gradient, which is $\Theta(1)/\binom{n}{n-4} = O(n^{-4})$, before finding the first point of the SP. Due to Markov's inequality applied iteratively over $\Omega(\sqrt{n})$ consecutive phases of length $\Theta(n^2)$ with an appropriate constant, the probability that the time to find the first point of SP is more than $\Omega(n^{5/2})$ is less than $2^{-\Omega(\sqrt{n})}$. The probability of jumping to a local optimum in $O(n^{5/2})$ steps is at most $O(n^{-4})O(n^{5/2}) = O(n^{-3/2})$ by the union bound. Therefore, SP is found before any local optima in at most $O(n^{5/2})$ generations with probability $1-o(1)$. Hence, the previously excluded event has now been taken into account.

After $1^{n-5}0^5$ is added to the population, the best solution on the path is improved with probability $\Omega(1/n)$ by hypermutation and in expected $O(n\log n)$ generations the global optimum is found. Since all SP and S_5 solutions have a Hamming distance smaller than $n-4$ and larger than $n/2$ to any S_{n-1} solution, the probability that a local optimum is found before the global optimum is at most $O(n\log n)\cdot \mu n \binom{n}{4}^{-1} = o(1)$ by the union bound. Thus with probability $1-o(1)$, the time to find the optimum is $O(n\log n)$ generations. We pessimistically assume that we start over when this event does not occur, which implies that the whole process until that point should be repeated $1/(1-o(1))$ times in expectation. Overall, the dominating term in the runtime is $O(\tau + n^{5/2})$ generations. By multiplying this time with the maximum possible amount of wasted fitness evaluations per generation (μcn), the upper bound is proven. □

In the following two theorems, we show that static HMP and ageing used in conjunction are essential.

Theorem 23. *Opt-IA using static HMP without ageing (i.e., $\tau = \infty$) with $\mu = O(\log n)$ and $dup = O(1)$ cannot optimise* HIDDENPATH *in less than $n^{\Omega(\log n)}$ expected fitness function evaluations.*

Proof. The only points with higher fitness than solutions with more than $n/2$ 0-bits are SP points and S_5 points. If all solutions in the population have less than $n - c_1 \log n$ 0-bits or less than $n - c_1 \log n$ 1-bits for some $c_1 > 1$, then the Hamming distance of any solution in the population to any SP solution or an S_5 solution is at least $\Omega(\log n)$. Since there are no other improving solution, the probability that an SP or an S_5 solution will be discovered in a single

hypermutation operation is exponentially small according to the last statement of Lemma 1. As a result, conditional on not seeing these points, the algorithm will find a solution with $n-c_1 \log n$ 0-bits in $O(\mu \cdot \text{dup} \cdot n^2 \log n)$ fitness function evaluations by optimising ZEROMAX (i.e., Theorem 20). In the rest of the proof we will calculate the probability of reaching an S_{n-1} solution before finding either an SP point or S_5 point, or a complementary solution of an SP point. The latter solutions, if accepted, with high probability would hypermutate into SP points. We call an event *bad* where any of the mentioned points are discovered.

Any solution in the search space can have at most two Hamming neighbours on SP or at most two Hamming neighbours that their complementary bit strings are on SP. The probability of sampling any of these neighbours is in the order of $O(1/n)$ since it is necessary to flip a specific bit position. Hamming distance to the remaining SP points and their complementary bit strings are at least two. The probability of reaching a particular solution with Hamming distance at least two is at most $\binom{n}{2}^{-1} = O(1/n^2)$, since two particular bits need to be flipped. Excluding the two Hamming neighbours (which are sampled with probability $O(1/n)$), the probability of reaching any other SP point is $O(1/n^2) \cdot O(\log n) = O(\log n/n^2)$ by the union bound. Now, taking the Hamming neighbours into consideration, the probability that any of the $\log n - 3$ SP points (or their complementary bit strings) are discovered is $O(1/n) + O(1/n^2) \cdot O(\log n) = O(1/n)$. For any initial solution with less than $n-10$ 0-bits, the probability of finding a solution with five 0-bits is at most $\binom{n}{6}^{-1}\binom{n}{5} = O(1/n)$. Since the probability of reaching an S_5 point from solutions with more than $n-10$ 0-bits may be much larger, we first calculate the probability of finding $n-11$ 0-bits before a bad event occurs.

The probability of finding a solution which improves the ZEROMAX value is at least $\Theta(1/n)$ (even when we exclude the potential improvements whose complementary bit strings are on SP). Since at every static HMP, the probabilities of finding an S_5 solution, an SP solution or its complementary bit string are all in the order of $O(1/n)$ (up to $n-10$ 0-bits), the conditional probability that a solution which improves the current best ZEROMAX value is found before any other improvement is in the order of $\Theta(1/n)/(\Theta(1/n) + O(\text{dup} \cdot \mu/n)) = \Omega(1/(\text{dup} \cdot \mu))$. The probability that it happens $c_1 \log n$ times immediately after the first solution with more than $n - c_1 \log n$ 0-bits is added to the population is at least $\Omega(\text{dup} \cdot \mu)^{-c_1(\log n)} = (\text{dup} \cdot \mu)^{-O(\log n)}$. This sequence of $c_1 \log n$ improvements implies that a solution with $n-11$ 0-bits is added to the population. We now consider the probability that one individual finds the local optimum (i.e., an S_{n-1} solution) by improving its ZEROMAX value in 10 consecutive generations and that none of the other individuals improve in these generations. The probability that the current best individual improves in the next step is at least $1/n$ and the probability that the other individuals do not improve is at least

$(1-c/n)^{\mu} \geq 1/e$. Hence, the probability that this happens consecutively in the next 10 steps is at least $(1/ne)^{10} = \Omega(n^{-10})$.

Once a locally optimal solution is found, with high probability it takes over the population before the optimum is found and the expected runtime conditional on the current population consisting only of solutions with $n-1$ 0-bits is at least $\binom{n}{\log n} \geq \frac{(n-\log n)^{\log n}}{(\log n)^{\log n}}$. By the law of total expectation, the unconditional expected runtime is bounded from below by $E(T) \geq (1-o(1))\, n^{-10}(\mathrm{dup}\cdot\mu)^{-O(\log n)}\frac{(n-\log n)^{\log n}}{(\log n)^{\log n}}$. This expression is in the order of $n^{\Omega(\log n)}$ for any $\mu = O(\log n)$ and $\mathrm{dup} = O(1)$. □

Theorem 24. *With probability at least $1-e^{-\Omega(n)}$, Opt-IA using SBM and ageing cannot optimise* HIDDENPATH *in less than $n^{\Omega(n)}$ fitness function evaluations.*

Proof. Since the number of 1-bits in an initial solution is binomially distributed with expectation $n/2$, the probability that an initial solution has less than $n/3$ 1-bits is bounded from above by $e^{-\Omega(n)}$ using Chernoff bounds. Hence, w.o.p., the population has a Hamming distance of at least $n/3-\log n-1 > n/4$ to any solution on the path and any solution with five 0-bits ($\mathrm{SP}\cup S_5$) by the union bound. Therefore, the probability of finding either any of the $\log n$ points on SP or one of the $\binom{n}{5}$ points on S_5 is $\left(\log n+\binom{n}{5}\right)\cdot 1/n^{n/4}(1-1/n)^{n-n/4} \leq n^{-\Omega(n)}$.

Since accepting any improvements, except for the $(\mathrm{SP}\cup S_5)$ solutions, increases the distance to the $(\mathrm{SP}\cup S_5)$ solutions, the probability of jumping to an solution in $(\mathrm{SP}\cup S_5)$ further decreases throughout the search process. □

4.4.3 When Using Ageing and Hypermutations Together is Detrimental

In this subsection, we present a function class for which Opt-IA is inefficient. The class of functions, which we call HYPERTRAP$_\gamma$, is defined formally in Definition 3. HYPERTRAP$_\gamma$ is inspired by the SP-TARGET function introduced in [67] and used in [52] as an example where hypermutations outperform SBM. Compared to SP-TARGET, HYPERTRAP$_\gamma$ has local and global optima inverted with the purpose of trapping hypermutations. Also there are some other modifications to prevent the algorithm from finding the global optimum via large mutations. The parameter γ defines the distance between the local optima and the path to the global optimum.

In HYPERTRAP$_\gamma$, the solutions with $|x|_1 < n/2$ are evaluated by ONEMAX. Solutions of the form 1^i0^{n-i} with $n/2 \leq i \leq n$, shape a short path which is called SP. The last point of SP, i.e., 1^n is the global optimum (OPT). Local optima of HYPERTRAP$_\gamma$ (LOCALOPT) are formed by the points with $|x|_1 \geq 3n/4$ which have a Hamming distance of at least γn to all points in SP. Also, the points with $|x|_1 = n/2$ are ranked among each other such that bit

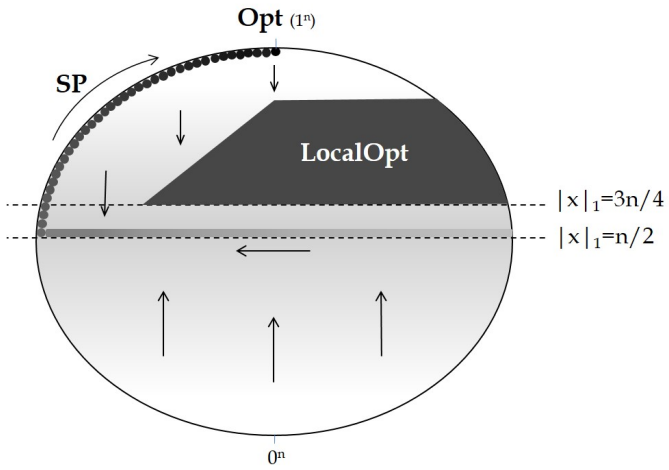

Fig. 4.2 The HYPERTRAP$_\gamma$ function

strings with more 1-bits in the beginning have higher fitness. This ranking forms a gradient from $0^{n/2}1^{n/2}$, which has the lowest fitness, to $1^{n/2-1}0^{n/2+1}$, which has the highest fitness. Finally, the fitness of points which do not belong to any of the mentioned sub-spaces is evaluated by ZEROMAX (i.e., the fitness is $|x|_0$).

Definition 3. *Given the definitions of* SP, OPT *and* LOCALOPT *as above and* $H(x,\text{SP})$ *showing the minimum Hamming distance of the individual* x *to all* SP *points, the* HYPERTRAP$_\gamma$ *function with* $0 < \gamma \leq 1/8$ *is defined for all* $x \in \{0,1\}^n$ *by:*

$$\text{HYPERTRAP}_\gamma(x) :=$$

$$\begin{cases} |x|_1 & \text{if } |x|_1 < n/2, \\ n/2 + \frac{\sum_{i=1}^{n}(n-i)\cdot(x_i)}{n} & \text{if } |x|_1 = n/2, \\ n^2 \cdot |x|_1 & \text{if } x \in \text{SP} := \{x \mid x = 1^i 0^{n-i} \text{ and } n/2 \leq i < n\}, \\ n^3 & \text{if } x \in \text{LOCALOPT} := \{x \mid |x|_1 \geq 3n/4 \text{ and } H(x,\text{SP}) \geq \gamma n\}, \\ n^4 & \text{if } x = 1^n = \text{OPT}, \\ |x|_0 & \text{if } |x|_1 > n/2 \text{ and } x \notin (\text{SP} \cup \text{LOCALOPT} \cup \text{OPT}). \end{cases}$$

This function is depicted in Figure 4.2 in which fitness increases with darker shades of gray. We will show that there exists a HYPERTRAP$_\gamma$ such that Opt-IA with mutation potential cn gets trapped in the local optima.

Theorem 25. *With probability* $1 - 2^{-\Omega(\sqrt{n})}$*, Opt-IA using static HMP with mutation potential* cn *cannot optimise* HYPERTRAP$_{c/8}$ *in less than exponential time.*

Proof. We will show that the population will first follow the ZEROMAX (or ONEMAX) gradient until it samples a solution with $n/2$ 0-bits and then the gradient of $\sum_{i=1}^{n}(n-i)\cdot(x_i)$, until it samples an SP point with approximately $n/2$ 0-bits. Afterwards, we will prove that large jumps on SP are unlikely. Finally, we will show that with overwhelming probability a locally optimal solution is sampled before a linear number of SP points are traversed. We optimistically assume that τ is large enough such that individuals do not die before finding the optimum.

With overwhelming probability, all randomly initialised individuals have $(1/2\pm\varepsilon)n$ 0-bits for any arbitrarily small $\varepsilon=\theta(1)$. Starting with such points, the probability of jumping to the global optimum is exponentially small according to Lemma 1. Being optimistic, we assume that the algorithm does not sample a locally optimal point for now. Hence, until an SP point is found, the current best fitness can only be improved if a point with either a higher ZEROMAX (or ONEMAX) or a higher $\sum_{i=1}^{n}(n-i)\cdot(x_i)$ value than the current best individual is sampled. Since there are less than n^2 different values of $\sum_{i=1}^{n}(n-i)\cdot(x_i)$, and less than n different values of ZEROMAX (or ONEMAX), the current best individual cannot be improved more than n^2+n times after initialisation without sampling an SP point.

Let SP_{lower} denote the SP points with less than $(1/2+2\varepsilon)n$ 1-bits, and SP_{upper} denote $\text{SP}\setminus\text{SP}_{lower}$. We will now show that w.o.p. an SP_{lower} point will be sampled before an SP_{upper} point. The search points with $(1/2-\varepsilon)n$ to $(1/2+\varepsilon)n$ 1-bits have at least a distance of εn to the SP_{upper} points. Using Lemma 1, we can bound the probability of sampling an SP_{upper} by $\binom{n}{\varepsilon n}^{-1}$ from any input solution with $(1/2\pm\varepsilon)n$ 1-bits. Excluding SP points, a solution with $(1/2\pm\varepsilon)n$ 1-bits has an improvement probability of at least $1/n^2$ (i.e., $\min\{1/n^2,\Theta(1)\}=1/n^2$ with the second term being the probability of improving the ZEROMAX or ONEMAX value and the first term being the probability of improving when the solution has exactly $n/2$ 0-bits). Thus, the conditional probability that the best solution is improved before any search point in the population is mutated into an SP_{upper} point is $(n^{-2})/(n^{-2}+\text{dup}\cdot\mu\cdot\binom{n}{\varepsilon n}^{-1})=1-2^{-\Omega(n)}$. This implies that an SP_{lower} point is sampled before an SP_{upper} point with overwhelming probability. Indeed, by the union bound, this event happens n^2+n times consecutively after initialisation with probability at least $1-(n^2+n)\cdot 2^{-\Omega(n)}=1-2^{-\Omega(n)}$, thus a point in SP_{lower} is sampled before an SP_{upper} point with overwhelmingly high probability. Each point of SP has at least $n/2$ 1-bits at the beginning of its bit string. In order to improve by any value in one mutation operation, 1-bits should never be touched before the mutation operator stops. Hence, the probability of improving by X on SP is $p(X)<(1/2)^X$. This yields that the probability of improving by $\sqrt{n}$ in one generation is at most $\text{dup}\cdot\mu\cdot(1/2)^{\sqrt{n}}$, as there are μ individuals in the population.

We have now shown that it is exponentially unlikely that even an arbitrarily small fraction of the path points are avoided by jumping directly to path points with more 1-bits.

Let t be the first iteration when the current SP solution has at least $99n/100$ 1-bits for the first time. We will first bound the probability of improving from below in order to find an upped bound on the conditional probability that more than $\sqrt{n}$ new 1-bits are added given that an improving path point is sampled. An improvement is achieved if the 0-bit adjacent to the last 1-bit is flipped in the first mutation step, which happens with probability $1/n$. Thus, the conditional probability of improving more than $\Omega(\sqrt{n})$ points on SP is at most $2^{-\Omega(\sqrt{n})}/(1/n) = 2^{-\Omega(\sqrt{n})}$. Therefore, in generation t, the current best individual cannot have more than $(99n/100) + \Omega(\sqrt{n})$ 1-bits with probability $1 - \text{dup} \cdot \mu \cdot 2^{-\Omega(\sqrt{n})}$ by the union bound. Similarly, the probability that no jump of size $\Omega(\sqrt{n})$ happens in $poly(n)$ generations is at least $1 - \text{dup} \cdot \mu \cdot 2^{-\Omega(\sqrt{n})}$. So, we can conclude that the number of generations to add another $n/200$ 1-bits to the prefix of the current best solution is at least $(n/200)/O(\sqrt{n}) = \Omega(\sqrt{n})$ with the same probability.

Now, we show that during this number of generations, the algorithm gets trapped in the local optima with overwhelming probability with the optimistic assumption that only the best b-cell in the population can be mutated into a locally optimal solution. Considering the best current individual, in each generation, a 1-bit is flipped with probability at least 99/100 in the first mutation step. Thus, the probability of not touching a 1-bit in $\Omega(\sqrt{n})$ generations is less than $(1/100)^{\Omega(\sqrt{n})}$. Considering that μ and dup are both $poly(n)$, the probability of flipping a 1-bit from the prefix in this number of generations and of not locating the optimum in the same amount of time is:

$$P \geq \left(1 - \left(\frac{1}{100}\right)^{\Omega(\sqrt{n})}\right) \cdot \left(1 - \frac{\sqrt{n}}{200} \cdot \text{dup} \cdot \mu \cdot 2^{-\Omega(\sqrt{n})}\right) \geq 1 - 2^{-\Omega(\sqrt{n})}.$$

After flipping a 1-bit, the mutation operator mutates at most cn bits until it finds an improvement. All sampled solutions in the first $n/4 - n/100$ mutation steps will have more than $3n/4$ 1-bits and thus satisfy the first condition of the local optima. If the Hamming distance between one of the first $n/4 - n/100$ sampled solutions and all SP points is at least γn, then the algorithm is in the trap. We only consider the SP points with a prefix containing more than $3n/4 - \gamma n$ 1-bits since SP points with less than $3n/4 - \gamma n$ 1-bits have already Hamming distance more than γn to the local optima.

Similarly to the analysis done in [52], we consider the kth mutation step for $k := 4\gamma n(1 + 1/5)/(3 - 4\gamma) = \frac{6cn}{30-5c} \leq cn$ where the last expression is due to $\gamma = c/8$. After k steps, the expected number of bits flipped in the prefix of length $n(3/4 - \gamma)$ is at least $k(3/4 - \gamma) = (1 + 1/5)\gamma n$. For any mutation potential $0 < c \leq 1$, $\frac{6cn}{30-5c} \leq n/4 - n/100$. Using Chernoff

bounds, we can show that the probability of having less than γn 0-bits in the prefix is $P(X \leq (1-0.2)E(X)) \leq e^{\frac{-\gamma n \cdot 6}{1000}} = e^{-\Omega(n)}$.

Altogether, with probability $(1-e^{-\Omega(\gamma n)}) \cdot (1-2^{-\Omega(\sqrt{n})}) = 1-e^{-\Omega(\sqrt{n})}$, a point in the local optima is sampled. In $O(\log \mu) = O(\log n)$ generations this individual takes over the population. Once in the local optima, the algorithm needs at least $\text{dup} \cdot \mu \cdot 1/\binom{n}{\gamma n}$ time to find the global optimum according to Lemma 1. □

In the following theorem we see how the $(1+1)$ EA using SBM optimises HYPERTRAP_γ in polynomial time w.o.p.

Theorem 26. *The* $(1+1)$ *EA optimises* HYPERTRAP_γ *in* $O(n^{3+\varepsilon})$ *steps w.o.p.* $1-e^{-\Omega(n^\varepsilon)}$ *for* $\varepsilon > 0$.

Proof. With overwhelming probability, the algorithm is initialised within $n/2 \pm n/10$ 1-bits by Chernoff bounds. Since the distance to any trap point is linear and the probability that SBM flips at least $\sqrt{n}$ bits in a single iteration ($\binom{n}{\sqrt{n}} n^{-\sqrt{n}} = 2^{-\Omega(\sqrt{n})}$) is exponentially small, so is the probability of mutating into a trap point. As the algorithm approaches the bit string with $n/2$ 1-bits, the distance to the trap remains linear. Conditional on not entering the trap, by standard AFL arguments (see Subsection 3.3.3, page 37) it takes the $(1+1)$ EA at most $O(\sqrt{n})$ steps in expectation to find a point with $n/2$ 1-bits, i.e., on the gradient. After finding such a point, the $(1+1)$ EA improves on the gradient with probability at least $1/n^2 \cdot (1-1/n)^{n-2} \geq 1/(en^2)$ which is the probability of flipping the rightmost 1-bit and the leftmost 0-bit while leaving the rest of the bits untouched. To reach the first point of SP, there are a linear number of 1-bits that need to be shifted to the beginning of the bit string. It therefore takes $O(n^3)$ steps in expectation to find SP. The $(1+1)$ EA improves on SP with probability at least $1/n \cdot (1-1/n)^{n-1} \geq 1/en$, which is the probability of flipping the leftmost 0-bit and not touching the other bits. Hence, the global optimum is found in $O(n^2)$ steps in expectation giving a total expected runtime of $O(n^3)$ conditional on not falling into the trap. By applying Markov's inequality iteratively (see Theorem 6, page 35) over $\Omega(n^\varepsilon)$ consecutive phases of length $\Theta(n^3)$ with an appropriate constant, the probability that the optimum is not found within $O(n^{3+\varepsilon})$ steps is less than e^{-n^ε} with ε being an arbitrarily small constant. Since the probability of finding a local optimum from the gradient points or SP points is $n^{-\Omega(n)}$ in each step, the probability of not falling into the trap in $n^{3+\varepsilon}$ steps is less than $(n^{3+\varepsilon} \cdot n^{-c'n})$ by the union bound. Overall, the total probability of finding the optimum within $O(n^{3+\varepsilon})$ steps is bigger than $(1-e^{-n^\varepsilon}) \cdot (1-n^{3+\varepsilon} \cdot n^{-c'n}) = 1-e^{-\Omega(n^\varepsilon)}$. □

4.4.4 Opt-IA Is Efficient for TRAP Functions

In [19], where Opt-IA was originally introduced, the effectiveness of the algorithm was tested for optimising the following simple trap function:

$$\text{SIMPLE TRAP}(x) := \begin{cases} \frac{a}{z}(z - |x|_1) & \text{if } |x|_1 \leq z, \\ \frac{b}{n-z}(|x|_1 - z) & \text{otherwise,} \end{cases}$$

where $z \approx n/4$, $b = n - z - 1$, $3b/2 \leq a \leq 2b$ and the optimal solution is the 0^n bit string.

The reported experimental results were averaged over 100 independent runs, each with a termination criterion of reaching 5×10^5 fitness function evaluations. For all of the results, the population size was $\mu = 10$ and dup $= 1$. In these experiments, Opt-IA using either hypermutations or hypermacromutation never finds the optimum of SIMPLE TRAP. However, the following theorem shows that Opt-IA indeed optimises SIMPLE TRAP efficiently.

Theorem 27. *Opt-IA using static HMP with mutation potential cn needs at most* $1 \cdot \mu n^2 \log n$ *expected fitness function evaluations to optimise* SIMPLE TRAP *with* $\tau = \Omega(n^2)$ *for* $c = 1$ *and dup* $= 1$.

Proof. Given that the number of 1-bits in the best solution is i, the probability of improving is at least $(n-i)/n$ if the best solution has more than z 1-bits and i/n otherwise. By following the proof of Theorem 14, at least one individual will reach 1^n or 0^n in at most $1 \cdot \mu n^2 \log n$ fitness function evaluations in expectation as long as no individual dies due to ageing.

The age of an individual reaches n^2 only if the improvement fails to happen in n^2 generations which happens with probability at most $(1 - 1/n)^{n^2} = 2^{-\Omega(n)}$ since the improvement probability is at least $1/n$. This implies that the expected number of restarts by ageing is exponentially small. By following the proof of Theorem 14, at least one individual will reach 1^n or 0^n in $O(\mu n^2 \log n)$ fitness function evaluations in expectation as long as no individual dies due to ageing. Pessimistically assuming that 1^n is found first, the algorithm finds the global optimum in one step with probability 1 by flipping all bits and performing n fitness function evaluations. □

Given that Opt-IA was tested in [19] also with the parameters suggested by Theorem 27 (i.e., $c = 1$, dup $= 1$, $\tau = \infty$), we speculate that either FCM was not used by mistake or that the stopping criterion (i.e., the total number of allowed fitness evaluations, i.e., 5×10^5) was too small. We point out that, for large enough τ also using hypermacromutation as mutation operator would lead to an expected runtime of at most $1 \cdot \mu n^2 \log n$ for TRAP functions [50], with or without FCM. In any case, it is not necessary to apply both static HMP and hypermacromutation together to efficiently optimise TRAP functions as reported

in [19]. On the other hand, the inversely proportional HMP considered in [52] would fail to optimise this function efficiently because it cannot flip n bits when on the local optimum.

4.4.5 Standard Opt-IA with Genotype Diversity

None of the algorithms considered in the previous sections of this chapter use the genotype diversity mechanism. In this section, we do not allow genotype redundancies in the population as proposed in the original Opt-IA algorithm (i.e., div := 1 in Algorithm 4.1) [19, 20]. This change potentially affects the behaviour of the algorithm. In particular, the probability of creating copies of individuals may change considerably. In the following, we will first consider the ageing operator in isolation with genotype diversity (i.e., no genotype duplicates are allowed in the population). Afterwards we will analyse the complete Opt-IA algorithm with the same diversity mechanism (as originally introduced in the literature).

We start by analysing the (μ+1) RLS$^{\text{ageing}}$ with genotype diversity for optimising $\textsc{Cliff}_d$ (defined by (3.7)) for which the $(\mu+1)$ RLS$_p^{\text{ageing}}$ without genotype diversity was previously analysed in Section 4.3. The main difference compared to the analysis there is that taking over the population on the local optima is now harder since identical individuals are not allowed. The proof of the following theorem shows that the algorithm can still take over and use ageing to escape from the local optima as long as the population size is not too large.

Theorem 28. *For $\mu = \Theta(1)$ and $\tau = \Theta(n \log n)$, the (μ+1) RLS^{ageing} with genotype diversity optimises $\textsc{Cliff}_d$ in expected $O(n \log n)$ fitness function evaluations for any linear $d \leq n/4 - \varepsilon$.*

Proof. By Chernoff bounds (see Theorem 8, page 36), with overwhelming probability the initial individuals are sampled with $n(1/2 \pm \varepsilon)$ 1-bits for any arbitrarily small $0 < \varepsilon = \Theta(1)$. Since the population size is constant and there is a constant probability of improving in the first mutation step, a local optimum is found in at most $O(n)$ steps in expectation by picking the best individual and improving it $O(n)$ times.

If there is a single locally optimal solution in the population, then with probability $1/\mu$ this individual is selected as parent and produces an offspring with fitness $n-d-1$ (i.e., one bit away) with probability $(n-d-i)/n \leq (n-d-\mu)/n = \Theta(1)$ where i is the number of individuals already with fitness $n-d-1$. In the next generation, with probability $(i+1)/\mu$, one of the individuals on the local optimum or one step away from it is selected as parent and produces an offspring on either the local optimum or one bit away with probability at least $(n-d-\mu)/n = \Theta(1)$. Hence, in expected time $\mu \cdot \Theta(1) = \Theta(\mu)$ all μ individuals are on the local optimum.

When the last inferior solution is replaced by an individual on the local optimum, the other individuals have ages in the order of $O(\mu)$. Thus, the probability that the rest of the population does not die until the youngest individual reaches age τ, is at least $(1/\mu)^{(\mu-1)\cdot O(\mu)} = \Theta(1)$, the probability that $\mu-1$ individuals above age τ survive $O(\mu)$ consecutive generations.

In the first generation when the last individual reaches age τ, with probability d/n an offspring is created at the bottom of the cliff (i.e., with a fitness value of $n-d+1$) and with probability $1/\mu \cdot (1-1/\mu)^{\mu} = \Theta(1)$ all the parents die together at that step and the offspring survives. The rest of the proof follows the same arguments as the proof of Theorem 17.

Overall, the total expected time to optimise CLIFF_d is dominated by the time to climb the second ONEMAX slope which takes $O(n \log n)$ steps in expectation. □

Now, we start analysing Opt-IA with genotype diversity (div = 1 in Algorithm 4.1) to optimise all the functions for which Opt-IA without genotype diversity was analysed in Section 4.4. The following are straightforward corollaries of Theorem 20 and Theorem 27. Since the ageing mechanism never triggers here and the proofs of those theorems do not depend on creating genotype copies, the arguments are still valid for Opt-IA with genotype diversity.

Corollary 2. *The upper bounds on the expected runtime of Opt-IA using static HMP with genotype diversity and ageing parameter τ large enough for* ONEMAX, LEADINGONES, JUMP_k *and* CLIFF_d *are as follows:*
$E(T_{\text{ONEMAX}}) = O\left(\mu \cdot dup \cdot n^2 \log n\right)$ *for* $\tau = \Omega(n^2)$,
$E(T_{\text{LEADINGONES}}) = O\left(\mu \cdot dup \cdot n^3\right)$ *for* $\tau = \Omega(n^2)$,
$E(T_{\text{JUMP}_k}) = O\left(\mu \cdot dup \cdot \frac{n^{k+1} \cdot e^k}{k^k}\right)$ *for* $\tau = \Omega(n^{k+1})$,
and $E(T_{\text{CLIFF}_d}) = O\left(\mu \cdot dup \cdot \frac{n^{d+1} \cdot e^d}{d^d}\right)$ *for* $\tau = \Omega(n^{d+1})$.

Corollary 3. *Opt-IA using static HMP with genotype diversity needs $O\left(\mu n^2 \log n\right)$ expected fitness function evaluations to optimise* SIMPLE TRAP *with $\tau = \Omega(\mu n^{1+\varepsilon})$, $c = 1$ and $dup = 1$.*

The following theorem shows the same expected runtime for Opt-IA with genotype diversity for HIDDENPATH as that of the Opt-IA without genotype diversity proven in Theorem 22. However, we reduce the population to be of constant size. The proof follows the main arguments of the proof of Theorem 22. Here we only discuss the probability of events which should be handled differently considering that genotype duplicates are not allowed.

Theorem 29. *For $c = 1$, $dup = 1$, $\mu = \Theta(1)$ and $\tau = \Omega(n^2 \log n)$, Opt-IA using static HMP with genotype diversity needs $O(\tau\mu n + \mu n^{7/2})$ expected fitness function evaluations to optimise* HIDDENPATH.

Proof. We follow the analysis of the proof of Theorem 22 for Opt-IA without genotype diversity. Although the analysis did not benefit from genotype duplicates, not allowing them potentially affects the runtime of the events where the population takes over. The potentially affected events are:

- For S_{n-1} solutions to take over the population, the probabilities are different here since a new S_{n-1} solution will not be accepted if it is identical to any current S_{n-1} solutions. Here, after finding the first S_{n-1} solution, the rest are created and accepted by Opt-IA with probability at least $\Omega(1/n \cdot (n-\mu)/(n-1)) = \Omega(1/n)$. Therefore, applying the argument made in the proof of Theorem 22 does not change the runtime asymptotically.

- The arguments about the expected time needed for S_{n-1} solutions to reach the same age after the takeover are the same as in the proof of Theorem 22; without genotype duplicates, the probability of creating another S_{n-1} is still $\Omega(1/n)$. Hence, the probability of creating two copies in the same generation is still unlikely and we can adapt the arguments made in the proof of Theorem 22. Therefore, in expected $O(\mu^3 n)$ generations after the takeover of S_{n-1}, the population reaches the same age.

- In the proof of Theorem 22, the expected time for S_5 solutions to take over the population of recently initialised individuals is bounded relying on having multiple copies of one S_5 and one S_{n-5} solution. This proof strategy cannot be applied with the genotype diversity mechanism which only allows unique solutions in the population.

  In order to create a unique S_5 solution from another S_5 solution, it is sufficient that the first bit position to be flipped has value 1 in the parent bit string and the second position to be flipped has value 0 in all S_5 solutions currently in the population, including the parent. Such a mutation occurs with probability at least $(\frac{5}{n} \cdot \frac{n-5-\mu}{n-1}) \geq 4/n = \Omega(1/n)$. This lower bound in the probability implies that the number of S_5 individuals in the population is increased by a factor of $1+\Omega(1/n)$ at every generation in expectation. For any constant c, the exponent t that satisfies $1 \cdot (1+c/n)^t = \mu$ is in the order of $O(n \log \mu)$. Thus, in at most $O(n \log \mu)$ generations in expectation the S_5 solutions take over the population if no solutions with higher fitness have been added to the population before the takeover. Due to $\mu = \Theta(1)$, the expected number of generations is simplified to $O(n)$. By Markov's inequality (see Theorem 6, page 35), the probability that the expected time is in the order of $O(n)$ is $\Omega(1)$.

  Only the solutions on SP and S_{n-1} solutions have better fitness than S_5 solutions. The rest of the proof of Theorem 22 can still be applied if the only remaining non S_5 solutions in the population are SP solutions. So, we will only show that it is unlikely

that an S_{n-1} solution will be sampled before the population consists only of S_5 and SP solutions. The number of 0-bits in randomly initialised solutions is in the interval of $[n/2-n^{2/3}, n/2+n^{2/3}]$ with probability $1-2^{\Omega(n^{1/3})}$ due to Chernoff bounds. We can then follow the strategy from Theorem 21 to show that the expectation of T^*, the time until an S_{n-1} solution descending from a randomly created solution is sampled, is in the order of $\Omega(n \log n)$. In order to bound the probability that this event will not happen in $O(n)$, we will bound the variance of this runtime. Since FCM stops after a solution with more 0-bits is sampled, we can pessimistically divide the runtime into phases of length T_i, $i \in \{2, \ldots, n/2+n^{2/3}\}$, where the best among the newly created solutions has i 1-bits. The length of phase T_i is distributed according to a geometric distribution with success probability at most $\mu \cdot 2i/n$ due to the Ballot theorem (see Theorem 11, page 38) and the union bound summed up over μ candidate solutions which can be improved at every generation. Being geometrically distributed, the variance of T_i is at most $(1-(d \cdot i/n))/(d \cdot i/n)^2$ for some constant $d > 2\mu$. Summing up over all phases of independently distributed lengths, we obtain the variance of T^* as:

$$Var(T^*) \leq \sum_{i=2}^{n/2+n^{2/3}} \frac{1-(d \cdot i/n)}{(d \cdot i/n)^2} \leq \frac{n^2}{d^2} \sum_{i=2}^{n/2+n^{2/3}} \frac{1}{i^2} = \frac{n^2}{d^2} \frac{\pi^2}{6} = \Theta(n^2).$$

Due to Chebyshev's inequality (see Theorem 7, page 36), the probability that such an event happens in $O(n)$ generations instead of its expectation, which is in the order of $\Omega(n \log n)$, is at most $O(1/\log n)$. Conversely, the probability that such a failure does not occur is $1-O(1/\log n)$.

The path solutions have between 5 to $\log n + 1$ 1-bits, thus the probability that the hypermutation operator yields an S_{n-1} solution as output given an SP or S_5 solution as input is at most $\binom{n}{n-4}^{-1} = O(n^{-4})$. The probability that such a mutation occurs in $O(n)$ generations is at most $O(n^{-3})$ by the union bound (see Theorem 4, page 35). Considering all the possible failure events, the takeover happens in $O(n)$ generations with $\Omega(1)$ probability before any S_{n-1} individual is added to the population.

The rest of the proof of Theorem 22 is not affected by the genotype diversity. □

4.5 Conclusion

This chapter presented an analysis of the standard Opt-IA artificial immune system. We first highlighted how both ageing and static HMP may allow an algorithm to efficiently escape local optima that are particularly hard for standard evolutionary algorithms. Concerning

hypermutations, we proved that FCM is essential to the operator and suggest considering a mutation *constructive* if the produced fitness is at least as good as the previous one. The reason is that far away points of equal fitness should be attractive for the sake of increasing exploration capabilities. Our analysis on the JUMP_k function suggests that static HMP with FCM is generally preferable to the SBM operator when escaping the local optima requires a jump of size k such that $(k/e)^k \geq n$. This advantage is least pronounced when there is a single solution with better fitness than the local optima as in the case of JUMP_k and the performance difference with the SBM in terms of expected escape time scales multiplicatively with the number of acceptable solutions at distance k. Hence, the JUMP_k function may be considered as a worst-case scenario concerning the advantages of hypermutations over SBM for escaping local optima.

Concerning ageing, we showed for the first time that the operator can be very efficient when coupled with SBM and hypermutations (i.e., for HIDDENPATH). To the best of our knowledge, the operator allows the best known expected runtime (i.e., $O(n \log n)$) for hard CLIFF_d functions if used with RLS. However, while ageing and SBM can still optimise CLIFF efficiently, these advantages of ageing for the function disappear if the operator is coupled with static HMPs. Afterwards, we presented a class of functions where both the characteristics of ageing and static HMP are crucial, hence Opt-IA is efficient while standard evolutionary algorithms are inefficient even if coupled with one extra AIS operator. Finally, we proved that all the positive results presented for the Opt-IA algorithm without genotype diversity also hold for Opt-IA with genotype diversity as used in the original algorithm. However, small population sizes may be required if ageing has to be triggered to escape from local optima with genotype diversity. To complete the picture we presented a class of problems where the use of hypermutations and ageing is detrimental while standard evolutionary algorithms are efficient.

Our analysis shows that for easy problems for which local search strategies are efficient, using static HMP and ageing may be detrimental. We have shown this effect for the simple ONEMAX and LEADINGONES functions for which we have proven a linear slow-down in the expected runtime compared to local search strategies and simple evolutionary algorithms. While such a slow-down may be considered an acceptable expense in exchange for being efficient on more complicated optimisation problems, we have shown for $\text{HYPERTRAP}_{c/8}$ that the consequences may be more drastic, by making the difference between polynomial and exponential runtimes. Indeed, the function is easy for local search algorithms with neighbourhoods of size greater than 1 and for simple EAs. However, static HMP make Opt-IA fail with overwhelming probability and ageing does not help the algorithm to escape from the trap permanently.

On the other hand, for more complicated multimodal functions, we have shown several advantages of static HMP and ageing to escape from local optima (i.e., JUMP_k and CLIFF_d). Furthermore, the combination of static HMP and ageing may be advantageous to locate new promising basins of attraction that are hard to find via more traditional optimisation techniques such as local search or EAs using SBM. We have illustrated such effect in the analysis of HIDDENPATH, where the combination of static HMP and ageing allow Opt-IA to locate a new basin of attraction that initially has lower fitness than the easy-to-find local optima. However, in the long run, having identified this new basin of attraction allows the algorithm to find the global optimum.

The next chapter focuses on showing that such advantages also hold for a classical combinatorial optimisation problem. The results will be achieved by building upon the work presented in this chapter.

Chapter 5

Analysis for the NP-Hard Partition Problem

5.1 Introduction

In the previous chapter, it was shown how both the hypermutation with static mutation potentials (static HMP) and the ageing operator of Opt-IA can lead to considerable speed-ups compared to the performance of well-studied EAs using standard bit mutation (SBM). The analyses were done for standard multimodal benchmark functions used in the evolutionary computation community such as JUMP, CLIFF and TRAP. While some of these speed-ups over standard EA's performance are particularly impressive, no similar evidence of superior performance of these two operators is available for more realistic problems.

In this chapter, we perform an analysis for PARTITION, also known as number partitioning, which is considered as one of the six basic NP-complete problems [37]. PARTITION as a decision making and an optimisation process arises in many resource allocation tasks in manufacturing, production and information processing systems [81, 39]. It also has applications in cryptography [62] and game theory [63]. In Section 5.2, PARTITION is first introduced and then in Subsections 5.2.1 and 5.2.2, problem specific algorithms and general-purpose heuristics (i.e., $(1+1)$ EA and RLS) that have been applied in the literature to target it are covered respectively. In Section 5.3, some preliminaries are introduced regarding the definition of a local optimum for PARTITION instances and the expected time needed for immune operators to escape from one.

It has been shown that RLS and EAs using SBM may get stuck on local optima of the problem which lead to a worst case approximation ratio of $4/3$ and, in order to achieve a $(1+\varepsilon)$ approximation for arbitrary ε, a clever restart strategy has to be put in place. The

generalised class of worst-case instances leading to such approximation ratio is introduced in Section 5.4. In Subsection 5.4.1, we first prove that the $(1+1)$ EA has exponential expected runtime on the whole class. Then, in Subsections 5.4.2 and 5.4.3, we show the power of static HMPs and ageing by proving that each of them solve to optimality the instances that are hard for RLS and SBM, by efficiently escaping local optima. Afterwards, in Section 5.5, we focus on worst-case approximation guarantees. In Subsection 5.5.1 we prove that AISs using static HMP guarantee arbitrarily good solutions of approximation ratio $(1+\varepsilon)$ in expected $O(n^3)$ time for any constant ε without requiring any restarts. Furthermore, in Subsection 5.5.2 we prove that an AIS with SBM and ageing can efficiently achieve the same approximation ratio in $O(n^2)$ expected fitness function evaluations, automatically restarting the optimisation process by implicitly detecting when it is stuck on a local optimum. Such expected polynomial runtimes are achievable for RLS and $(1+1)$ EA if the desired approximation ratio is decided in advance while none of this information is required for either of the two AISs to efficiently identify the same approximation ratios. This is the first time that either hypermutations or ageing have been theoretically analysed for a standard problem from combinatorial optimisation and the first time performance guarantees of any AIS are proven for a classical problem from combinatorial optimisation.

5.2 The Partition Problem

Given n jobs with positive processing times $p_1 \geq p_2 \geq \cdots \geq p_{n-1} \geq p_n$, the makespan scheduling problem with m parallel and identical machines is that of scheduling the n jobs on the m machines in a way that the overall completion time (i.e., the *makespan*) is minimised [81]. Assigning the jobs to the machines is equivalent to dividing the set of processing times into m disjoint subsets such that the largest subset sum yields the makespan. When $m = 2$ and the processing times p_i are scaled up to be integers p_i^*, the problem is equivalent to the PARTITION problem with the set of integers $a_i := p_i^*$. This simple to define scheduling problem is well-studied in theoretical computer science and is known to be NP-hard [54, 2]. Hence, it cannot be expected that any algorithm finds optimal solutions to all instances in polynomial time (unless $P = NP$). The PARTITION (optimisation) problem is formally defined as below.

Definition 4. *Given positive integers $a_1, a_2, \ldots, a_n$, what is* $\arg\min_{B \subseteq [n]} \max \left(\sum_{i \in B} a_i, \sum_{i \in [n] \setminus B} a_i \right)$*?*

5.2.1 State-of-the-Art of Problem-Specific Algorithms

The PARTITION problem is considered an easy NP-hard problem since there are some dynamic programming algorithms that provide optimal solutions in pseudo-polynomial time of $O(n\sum p_i)$ [81]. The general idea behind such algorithms is to create an $n \times \sum p_i$ table where each pair (i, j) indicates whether there exists a subset of i jobs that sums up to j. Then, the optimal subset can be computed by simple backtracking. Clearly, such algorithms are not practical when $n \times \sum p_i$ is large.

As the NP-hardness of PARTITION suggests, not all instances can be optimised in polynomial time unless $P = NP$. Hence, approximation algorithms are used to obtain approximate solutions. For a minimisation problem, an algorithm A which runs in polynomial time with respect to its input size is called a $(1+\varepsilon)$ *approximation algorithm* if it can guarantee solutions with quality $(1+\varepsilon) \cdot f(opt)$ for some $\varepsilon > 0$ where $f(opt)$ is the quality of an optimal solution. Such an algorithm is called a *polynomial time approximation scheme* (PTAS) if it can provide a $(1+\varepsilon)$ solution for any arbitrarily small $\varepsilon > 0$. If the runtime is also polynomial in $1/\varepsilon$, then algorithm A is called a *fully polynomial time approximation scheme* (FPTAS).

Since the PARTITION problem is a special case of makespan scheduling with $m = 2$, any approximation algorithm for the latter problem may be applied to PARTITION. The first approximation algorithm for makespan scheduling was presented by Graham in 1969 [38]. He showed a worst-case approximation ratio of $2 - 1/m$ where m is the number of machines (this ratio is $3/2$ for PARTITION). Graham also proposed a simple heuristic which guarantees a better approximation ratio of $4/3 - 1/(3m)$ (which is $7/6$ for PARTITION) with running time of $O(n \log n)$. This greedy scheme, known as *longest processing time* (LPT), assigns the job with longest processing time to the emptier machine at each step. He then improved the approximation ratio to $1 + \frac{1-1/m}{1+\lceil k/m \rceil}$ by using a more costly heuristic. This PTAS first assigns the k largest jobs in an optimal way, then distributes the remaining (small) jobs one by one in any arbitrary order to the emptiest machine. The ideas behind these heuristics are widely used in later works, including a heuristic proposed by Hochbaum and Shmoys which guarantees a $(1+\varepsilon)$ approximation in $O((n/\varepsilon)^{1/\varepsilon^2})$ for makespan scheduling on an arbitrary number of machines [41]. This heuristic first finds an optimal way of assigning large jobs and then distributes the small jobs using LPT. This is roughly the best runtime achievable for makespan scheduling on an arbitrary number of machines [41]. However, when the number of machines is limited, better results are available in the literature. For $m = 2$ (i.e., PARTITION), Ibarra and Kim have shown the first FPTAS for PARTITION which runs in $O(n/\varepsilon^2)$ and shortly after Sahni showed an FPTAS based on dynamic programming which runs in $O(n^2/\varepsilon)$. This runtime is $O(n \cdot (n^2/\varepsilon)^{m-1})$ for larger number of machines

$m > 3$ [85]. Recently, an FPTAS was shown for PARTITION that admits a sub-quadratic approximation scheme with time complexity $O(n + 1/\varepsilon^{5/3})$ [65]. This is the first time any sub-quadratic FPTAS is shown for an NP-hard problem.

5.2.2 State-of-the-Art of General-Purpose Heuristics

For the application of randomised search heuristics, a solution to PARTITION can be represented with a bit string $x \in \{0,1\}^n$ where each bit i represents job number i. The bit value specifies which machine the job i is assigned to: if $x_i = 0$, then job i is placed in the first machine (M_1) and if $x_i = 1$, then it is placed in the second machine (M_2). Hence, the goal is to minimise the makespan $f(x) := \max\left\{\sum_{i=1}^{n} p_i x_i, \sum_{i=1}^{n} p_i(1-x_i)\right\}$, i.e., the fitness function returns the processing time of the last machine to terminate.

Both RLS and the $(1+1)$ EA have roughly $4/3$ worst case expected approximation ratios for the problem (i.e., there exist instances where they can get stuck on solutions by a factor of $4/3-\varepsilon$ worse than the optimal one) [90]. However, if an appropriate restart strategy is set-up in advance, both algorithms may be turned into polynomial randomised approximation schemes, i.e., a PRAS (algorithms that compute a $(1+\varepsilon)$ approximation in polynomial time in the problem size with probability at least $3/4$ [68]) with a runtime bounded by $O(n\ln(1/\varepsilon)) \cdot 2^{(e\log e+e)\lceil 2/\varepsilon\rceil \ln(4/\varepsilon)+O(1/\varepsilon)}$ [90]. Hence, as long as ε does not depend on the problem size, the algorithms can achieve arbitrarily good approximations with an appropriate restart strategy. The analysis of the $(1+1)$ EA and RLS essentially shows that the algorithms with a proper restart strategy simulate Graham's PTAS. In this chapter we will show that AISs can achieve stronger results.

5.3 General Preliminary Results

For the approximation results (especially for the hypermutation), we had to diverge significantly from the analysis done in [90], which relies on a constant upper bound on the probability that the assignment of a constant number of specific jobs to the machines will not be changed for at least a linear number of iterations [90]. The static HMP makes very large changes to the solution in every iteration and, unlike for the standard bit mutation operator, we cannot rely on successive solutions resembling each other in terms of their bit string representations. Thus, we have worked with ranges of the objective function values. In particular, we divided the range of objective function values into L intervals, where L is the number of local optima with distinct makespan values. Measuring the progress in the optimisation task according to the interval of the current solution allows the achievement

of the approximation result for the hypermutation operator and the bound on the time until the ageing operator reinitialises the search process. The bound on the reinitialisation time in turn allows us to transform the bound obtained in [90] on the probability that the standard $(1+1)$ EA finds the approximation in a single run into an upper bound on the approximation time for the $(\mu+1)$ EA$^{\text{ageing}}$.

We will refer to any solution x of the PARTITION problem as a *local optimum* if there exists no solution with smaller makespan which is at Hamming distance one from x. Moreover, $\mathscr{L} := \{\ell_1, \ell_2, \ldots, \ell_L\}$ will denote the set of all local optima for a PARTITION instance where $L := |\mathscr{L}|$ and the local optima ℓ_i are indexed according to non-increasing makespan.

We can now state the following helper lemma which bounds the expected number of iterations that the $(\mu+1)$ EA$^{\text{ageing}}$ (Algorithm 2.7 for minimisation, page 26) and the $(1+1)$ IA (Algorithm 2.4 for minimisation using static HMP) with current fitness better than ℓ_i require to reach a fitness at least as good as $f(\ell_{i+1})$.

Lemma 6. *Let $x \in \{0,1\}^n$ be a non-locally optimal solution to the* PARTITION *instance such that $f(\ell_i) > f(x) \geq f(\ell_{i+1})$ for some $i \in [L-1]$. Then,*

1. *The $(1+1)$ IA with current solution x samples a solution y such that $f(y) \leq f(\ell_{i+1})$ in at most $2n^2$ expected generations.*

2. *The $(\mu+1)$ EA^{ageing} with $\mu = O(\log n)$, $\tau = \Omega(n^{1+c})$ for an arbitrarily small positive constant c and current best solution x, samples a solution y such that $f(y) \leq f(\ell_{i+1})$ in $2en^2(1+o(1))$ iterations with probability at least $1 - n^{-\Omega(n^{c/2})}$.*

Proof. Let X_t be the best solution of the $(\mu+1)$ EA$^{\text{ageing}}$ or the current solution of the $(1+1)$ IA at time t, and p_t^* denote the weight of the largest job in X_t that can be moved from the fuller machine to the emptier machine such that the resulting solution yields a better makespan than X_t (given that no other jobs are moved). Moreover, we will define $D_t := f(X_t) - f(\ell_{i+1})$ and let r_t be the smallest $j \in [n]$ such that $\sum_{i=n-j+1}^{n} p_i \geq D_t$.

We will first prove that $p_t^* \geq p_{n-(r_t-1)}$ by contradiction. The assumption $p_t^* < p_{n-r_t+1}$ implies that $\sum_{p_i \leq p_t^*} p_i < D_t$. Now, we consider a solution y identical to X_t with the exception that all the jobs with weight at most p_t^* in the fuller machine of X_t are moved to the emptier machine. Thus, $f(X_t) > f(y) > f(X_t) - D_t = f(\ell_{i+1})$ while y is a local optimum. Since there exist no local optima with makespan value in $[f(x), f(\ell_{i+1})]$ by definition, the initial assumption creates a contradiction and we conclude that $p_t^* \geq p_{n-r_t+1}$.

We consider an iteration where only the job with processing time p_t^* is moved from the fuller machine to the emptier machine of X_t, and will denote any such event as a *large improving move*. After a large improving move, either the makespan will decrease by p_t^* or the

emptier machine in X_t becomes the fuller machine in X_{t+1} and $f(X_t) - p_t^* < f(X_{t+1}) < f(X_t)$. We will denote the latter case as a *switch*.

In the former case, if the makespan decreases by p_t^*, then $r_{t+1} \leq r_t - 1$ since $D_{t+1} = D_t - p_t^* \leq D_t - p_{n-r_t+1}$. Since $r_{t+1} \leq r_t \leq n$ by definition, after at most n large improving moves where $f(X_{t+1}) = f(X_t) - p_t^*$, we have $f(X_t) \leq f(\ell_{i+1})$.

If a switch occurs at time t, then the load of the emptier machine of $X_{t'}$ will be at least $f(X_t) - p_t^*$ for any $t' > t$ and moving a single job with processing time at least p_t^* from the fuller machine to the emptier machine of $X_{t'}$ will yield a solution with makespan worse than $f(X_{t+1})$. Therefore, if a switch occurs at time t, by the definition of p_t^*, for any $t' > t$, $p_{t'}^* < p_t^*$. Since there are at most n distinct processing times among all jobs, there can be at most n large improvements throughout the optimisation process where a switch occurs. Thus, we can conclude that in at most $2n$ large improvements, $f(X_t) \leq f(\ell_{i+1})$.

We will now bound the waiting time T_{imp} until a large improving move occurs separately for the $(1+1)$ IA and the $(\mu+1)$ EA$^{\text{ageing}}$.

For the $(1+1)$ IA, the probability that the job with processing time p_t^* is moved from the fuller machine to the emptier machine in the first mutation step is at least $1/n$. Since the resulting solution has smaller makespan, static HMP stops after this first mutation step. Due to the waiting time argument (see Theorem 3, page 35), $E[T_{imp}] = n$. Our first claim follows by multiplying this expectation with $2n$.

For the $(\mu+1)$ EA$^{\text{ageing}}$, we will first show that it takes $o(n)$ expected time until the best solution takes over the population if no improvements are found. Given that there are k individuals in the population with fitness value $f(X_t)$, with probability $(k/\mu)(1-1/n)^n \geq (k/\mu)(1-1/n)e^{-1}$, the $(\mu+1)$ EA$^{\text{ageing}}$ picks one of the k individuals as parent, does not flip any bit and adds a new solution with the same fitness value to the population. Thus, in expected $\sum_{k=1}^{\mu}(\mu/k)e(1-1/n)^{-1} \leq e\mu\ln\mu(1+o(1)) = o(n)$ generations, the whole population consists of solutions with fitness value at least $f(X_t)$. In the following generations with probability at least $1/n(1-1/n)^{n-1} \geq 1/(en)$, the SBM only moves the job with weight p_t^* and improves the best solution. Thus, in at most $e\mu\ln\mu(1+o(1)) + en = en(1+o(1))$ expected iterations of the $(\mu+1)$ EA$^{\text{ageing}}$, a large improving move occurs.

Since T_{imp} is a waiting time, we can use Markov's inequality iteratively (see Theorem 6, page 35) in the following way to show that the ageing operator in the $(\mu+1)$ EA$^{\text{ageing}}$ with overwhelmingly probability does not trigger : $P(T_{imp} \geq n^c \cdot E[T_{imp}]) \leq P(T_{imp} \geq n^{c/2} \cdot E[T_{imp}])^{n^{c/2}} = n^{-\Omega(n^{c/2})}$. The second claim is obtained by multiplying the bound on the waiting time with $2n$. □

5.4 Class of Worst Case Instances for Evolutionary Algorithms

The instance from the literature P_ε^* leading to a $4/3$ worst-case expected approximation ratio for RLS and the $(1+1)$ EA consists of two large jobs with long processing times $p_1 := p_2 := 1/3 - \varepsilon/4$ and the remaining $n-2$ jobs with short processing times $p_i := (1/3+\varepsilon/2)/(n-2)$ for $3 \leq i \leq n$ for an arbitrarily small constant $\varepsilon > 0$ (the sum of the processing times are normalised to 1 for cosmetic reasons) [90]. Any partition where one large job and half of the small jobs are placed on each machine is naturally a global optimum of the instance (i.e., the makespan is $1/2$). Note that the sum of all the processing times of the small jobs is slightly larger than the processing time of one large job. The local optimum leading to the $4/3$ expected approximation consists of the two large jobs on one machine and all the small jobs on the other. It is depicted in Figure 5.1. The makespan of the local optimum is $p_1 + p_2 = 2/3 - \varepsilon/2$ and, in order to decrease it, a large job has to be moved to the fuller machine and at the same time at least $\Omega(n)$ small jobs have to be moved to the emptier machine. Since this requires flipping $\Omega(n)$ bits, which cannot happen with RLS and only happens with exponentially small probability $n^{-\Omega(n)}$ with the SBM of EAs, their expected runtime to escape from such a configuration is at least exponential.

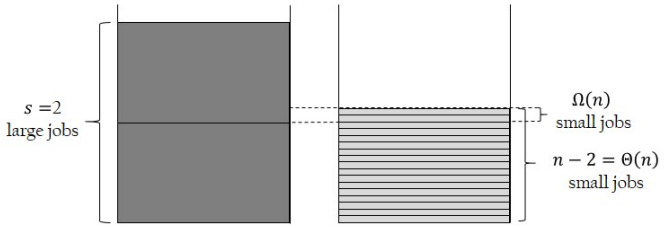

Fig. 5.1 The $4/3$ approximation local optimum of P_ε^* [90]

To highlight the power of the AIS to overcome hard local optima for EAs, in this section we generalise the instance P_ε^* to contain an arbitrary but constant number $s = \Theta(1)$ of large jobs and show how the considered AISs efficiently solve the instance class to optimality by escaping from local optima efficiently while RLS and the $(1+1)$ EA cannot. Hence, static HMP and ageing are efficient at locating the optimal solution on a vast number of instances where SBM and local mutations alone fail. The generalised instance class G_ε^* is defined as follows.

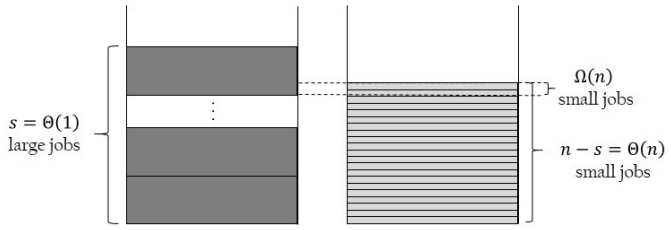

Fig. 5.2 A local optimum of G^*_ε

Definition 5. *The class of* PARTITION *instances G^*_ε is characterised by an even number of jobs n, an even number of large jobs $s = \Theta(1)$ and the following processing times:*

$$p_i = \begin{cases} \frac{1}{2s-1} - \frac{\varepsilon}{2s} & \text{if } i \leq s, \\ \frac{s-1}{n-s} \cdot \left(\frac{1}{2s-1} + \frac{\varepsilon}{2(s-1)}\right) & \text{otherwise,} \end{cases}$$

where $0 < \varepsilon < 1/(2s-1)$ is an arbitrarily small constant. In an optimal solution, each machine contains half of the large jobs and half of the small jobs.

The instance class has the same property as the instance P^*_ε. If all the large jobs are placed on one machine and all the small jobs on the other (see Figure 5.2), then at least $\Omega(n)$ small jobs need to be moved in exchange for a large job to decrease the discrepancy (the difference between the loads of the machines). Such a local optimum allows us to derive a lower bound on the worst-case expected approximation ratio of algorithms that get stuck there. Clearly, the greater the number of large jobs, the smaller the bound on the approximation ratio will be. In the following lemma, we will point out some important characteristics of the instance class.

Lemma 7. *Let $x \in \{0,1\}^n$ be a candidate solution to the instance $I \in G^*_\varepsilon$, h and k be the numbers of large and small jobs respectively on the fuller machine of x and $\mathscr{L} \subseteq \{0,1\}^n$ be the set of local optima of I. Then, the following three properties hold:*

Property 1: $|\{y \in \mathbb{R} : \exists x \in \{0,1\}^n \text{ s.t. } f(x) = y\}| = O(n)$.

Property 2: $|\{y \in \mathbb{R} : \exists x \in \mathscr{L} \text{ s.t. } f(x) = y\}| = O(1)$.

Property 3: $\forall x \in \{0,1\}^n$ *s.t.* $f(x) > 1/2$, *either* $k = \Omega(n)$ *and* $x \notin \mathscr{L}$, *or* $\max(k, n-s-k) \geq \left(\frac{1}{2} + a\right)(n-s)$ *for some* $a = \Omega(1)$.

The lemma provides upper bounds on the number of distinct makespan values that can be observed, first for all possible solutions in Property 1 and then only among local optima in Property 2. Finally, Property 3 provides restrictions on the distribution of the small jobs among the machines which hold for all suboptimal solutions.

Proof. Since $p_i = p_j = p_1$ for all $(i,j) \in [s]^2$ and $p_i = p_j = p_{s+1}$ for all $(i,j) \in ([n] \setminus [s])^2$, $f(x) = h \cdot p_1 + k \cdot p_{s+1}$. The first property follows from $(h,k) \in [s] \times [n-s]$ and $s = O(1)$.

For the second property which considers only the local optima $\mathscr{L}$, note that if a local optimum ℓ with h large jobs and k small jobs exists, then a solution which has h large jobs and $k' < k$ small jobs on its fuller machine cannot exist since, otherwise, moving a small job to the emptier machine would improve ℓ. Moreover, any solution with h large jobs and $k' > k$ small jobs on its fuller machine cannot be a local optima since it can be improved by moving a small job from its fuller machine to its emptier machine. Thus, there exists a unique locally optimal makespan value for each $h \in [s]$. The property follows from $s = O(1)$.

In order to show the third property, we will partition the set of suboptimal solutions $\mathscr{X} \subset \{0,1\}^n$ into three disjoint sets:

- $\mathscr{X}_1 := \{x \in \mathscr{X} : f(x) \geq \sum_{i=1}^{s} p_i + \left(\frac{1}{s+1} \sum_{i=s+1}^{n} p_i\right)\}$,
- $\mathscr{X}_2 := \{x \in \mathscr{X} \setminus \mathscr{X}_1 : h < s \wedge x \notin \mathscr{L}\}$,
- $\mathscr{X}_3 := \mathscr{X} \setminus (\mathscr{X}_1 \cup \mathscr{X}_2)$.

For all $x \in \mathscr{X}_1$, since $f(x) \geq \sum_{i=1}^{s} p_i + \left(\frac{1}{s+1} \sum_{i=s+1}^{n} p_i\right) > \frac{1}{2} + \frac{1}{s+1} \sum_{i=s+1}^{n} p_i$, moving a single small job from the fuller machine to the emptier machine always improves the makespan, thus $x \notin \mathscr{L}$. Moreover, since the sum of the processing times of the large jobs on the fuller machine can be at most $\sum_{i=1}^{s} p_i$, we get $k \geq (n-s)/(s+1) = \Omega(n)$.

For any $x \in \mathscr{X}_2$, $x \notin \mathscr{L}$ by definition. We will prove by contradiction that $k = \Omega(n)$. Let us assume $k = o(n)$. Since the condition $h < s$ implies that there is at least one large job on the emptier machine, the load of the emptier machine is at least:

$$\begin{aligned}
&\left(\frac{1}{2s-1} - \frac{\varepsilon}{2s}\right) + (n - s - o(n)) \cdot \frac{s-1}{n-s} \left(\frac{1}{2s-1} + \frac{\varepsilon}{2(s-1)}\right) \\
&\geq \frac{1}{2s-1} - \frac{\varepsilon}{2s} + \frac{s-1}{2s-1} + \frac{s\varepsilon(s-1)}{s2(s-1)} = \frac{1+s-1}{2s-1} + \frac{s\varepsilon - \varepsilon}{2s} \\
&\geq \frac{s}{2s-1} + \frac{\varepsilon(s-1)}{2s-1} \geq \frac{s+\varepsilon(s-1)}{2s-1} > \frac{s}{2s-1} > \frac{1}{2},
\end{aligned}$$

which contradicts that it is the emptier machine. Thus, $k = \Omega(n)$.

Finally, we will show that for all $x \in \mathscr{X}_3$, $\max(k, n-s-k) \geq \left(\frac{1}{2} + a\right)(n-s)$ for some $a = \Omega(1)$. If $x \in \mathscr{X}_3$ and $h = s$, then $k < \frac{n-s}{s+1}$ since $f(x) < \sum_{i=1}^{s} p_i + \left(\frac{1}{s+1} \sum_{i=s+1}^{n} p_i\right)$. The inequality $k < \frac{n-s}{s+1}$ in turn implies that $n-s-k > (n-s) - \frac{n-s}{s+1} = (n-s)(1-(1/(s+1))) \geq (1/2+a)(n-s)$ for some a. If $x \in \mathscr{X}_3$ and $x \in \mathscr{L}$, then either there are no small jobs on the fuller machine ($k = 0$) or the difference between the loads is so small that moving a small job

to the emptier machine does not improve the makespan. For the latter case, since $f(x) > \frac{1}{2}$, the machines cannot have the same number of small and large jobs. Moreover, since s and n are even integers and $x \in \mathscr{L}$, if one of the machines has more than half of the large (small) jobs assigned to it, then more than half of the small (large) jobs has to be assigned to the other machine, because otherwise moving a single small job from the fuller machine to the emptier one would improve the makespan. Without loss of generality (w.l.o.g.), let M_1 be the machine with more small jobs and fewer large jobs. Since $\sum_{i=1}^{n} p_i = 1$, the load of M_1 is at least $1/2 - p_{s+1}/2$. By definition, the number of large jobs in M_1, is bounded from above by $s/2-1$. Then, the contribution of small jobs to the load of M_1 is at least:

$$\frac{1}{2}\left(s \cdot p_1 + (n-s)p_{s+1} - p_{s+1}\right) - \left(\frac{s}{2} - 1\right) p_1 = \frac{n-s-1}{2} p_{s+1} + p_1,$$

where the first expression makes use of $s \cdot p_1 + (n-s)p_{s+1} = 1$. Dividing by p_{s+1}, the number of small jobs is at least:

$$\frac{n-s-1}{2} + \frac{p_1}{p_{s+1}} > \frac{n-s-1}{2} + \frac{n-s}{s} = (n-s)\left(\frac{1}{2} + \frac{1}{s}\right)\left(1 - O\left(\frac{1}{n}\right)\right).$$

Thus, for all $x \in \mathscr{X}_3$, $\min(k, n-s-k) \geq (n-s)(\frac{1}{2}+a)$ where $a := \min(\frac{1}{s} - O(\frac{1}{n}), \frac{1}{2} - \frac{1}{s+1}) = \Omega(1)$. □

5.4.1 Standard Bit Mutations Are Inefficient

In this subsection we prove that the $(1+1)$ EA has exponential expected runtime on G^*_ε. The proof is similar to that of the $4/3$ approximation [90]. It essentially shows that with constant probability the algorithm gets trapped in a state where it is exponentially unlikely to move the extra large jobs out of its fuller machine. That RLS also fails is essentially a corollary.

Theorem 30. *The $(1+1)$ EA needs at least $n^{\Omega(n)}$ fitness function evaluations in expectation to optimise any instance of G^*_ε.*

Proof. The proof is similar to that of the $4/3$ approximation [90]. We will first prove that with constant probability the $(1+1)$ EA initialises with all large jobs in M_1 and then fills M_2 with $n - s - \frac{(n-s)\varepsilon}{2(s-1)}$ small jobs before moving any large job to M_2 (Event $\mathscr{E}_1$). From then on, we will show that the algorithm requires exponential expected time to move a large job. Since the sum of the processing times of large jobs is larger than $1/2$ (the makespan of the optimum), the expected optimisation time will be exponential as well.

With probability $(1/2)^s = \Theta(1)$, all large jobs are assigned to the same machine (M_1 w.l.o.g.) in the initial solution. The expected number of small jobs initially assigned to

M_1 is $(n-s)/2$ and since they are assigned independently, the probability that more than $((n-s)/2)+(n-s)^{3/5}$ of them are assigned to M_1 initially is exponentially small by Chernoff bounds (see Theorem 8, page 36).

Let k_t be the number of small jobs on the fuller machine M_1 at time t. As long as the large jobs are not moved, new solutions are only accepted if the number of small jobs in M_2 increases or stays the same. A single small job is moved to M_2 with probability $(k_t/n)(1-1/n)^{n-s-1} \geq k_t/(en)$.

The expected time conditional on large jobs never having been moved until a $(1-\frac{\varepsilon}{2(s-1)})$ fraction of the total small jobs are moved to M_2 is:

$$\sum_{k=n-s-(n-s-\frac{\varepsilon(n-s)}{2(s-1)})}^{(n-s)/2+(n-s)^{3/5}} \frac{en}{k} \leq en \sum_{k=\frac{\varepsilon(n-s)}{2(s-1)}}^{(n-s)/2+(n-s)^{3/5}} \frac{2(s-1)}{\varepsilon(n-s)}$$
$$\leq 2(s-1)\left(\frac{e}{\varepsilon}\right)\left(\frac{n}{n-s}\right)\left(\frac{n-s}{2}+(n-s)^{3/5}\right) = O(n).$$

Let c be a constant such that $cn-1$ is at least twice as large as the above expectation. Due to Markov's inequality at least a $(1-\frac{\varepsilon}{2(s-1)})$ fraction of the small jobs are moved to M_2 in $cn-1$ iterations with probability at least $1/2$. Now, we calculate the probability that no large jobs have been moved in $cn-1$ iterations. The probability that SBM does not move any large jobs to the emptier machine (M_2) is $1-s/n$ in each iteration. The probability that the large jobs are not moved during the first $cn-1$ iterations is at least $(1-s/n)^{cn-1} \geq e^{-cs} = \Omega(1)$. Hence, $P(\mathscr{E}_1) = \Omega(1)$. Now that we found the probability of event $\mathscr{E}_1$, we can find the expected runtime conditional on $\mathscr{E}_1$.

Let at least a $(1-\frac{\varepsilon}{2(s-1)})$ fraction of small jobs and all the large jobs be on M_2. The sum of the processing times of small jobs on M_2 is greater than the processing time of $(s-1)$ large jobs (i.e., $(s-1)(\frac{1}{2s-1}-\frac{\varepsilon}{2s})$) and greater than:

$$(s-1)\left(\frac{1}{2s-1}+\frac{\varepsilon}{2(s-1)}\right)-\frac{\varepsilon}{2}\left(\frac{1}{2s-1}+\frac{\varepsilon}{2(s-1)}\right)$$
$$=\frac{s-1}{2s-1}+\frac{\varepsilon}{2}\left(1-\frac{1}{2s-1}-\frac{\varepsilon}{2(s-1)}\right)>\frac{s-1}{2s-1}+\frac{\varepsilon}{3}-\frac{\varepsilon^2}{4}>\frac{s-1}{2s-1}+\frac{\varepsilon}{4}.$$

Hence, the makespan can be at most $S<1-\left(\frac{s-1}{2s-1}+\frac{\varepsilon}{4}\right)=\frac{s}{2s-1}-\frac{\varepsilon}{4}$. If a large job was to be transferred to M_2, then its total processing time would become $S'>\frac{s}{2s-1}+\frac{\varepsilon}{4}-\frac{\varepsilon}{2s}>\frac{s}{2s-1}$. Hence, in order for a large job to be transferred to M_2, a processing time of at least $\varepsilon/4$ has to be transferred to M_1 in the same iteration. Then the conditional expected time

to transfer a large job would be exponential since at least a linear number of small jobs need to be moved from M_2 to M_1 such that their processing times of order $O(1/n)$ sum up to a constant $\varepsilon/4$. Such a mutation requires at least a linear number of bits to be flipped which happens with probability at most $n^{-\Omega(n)}$. The probability that such a mutation happens in $n^{\delta n}$ steps is still in the same asymptotic order for a sufficiently small constant δ. Since any mutation that increases the load of M_2 will not be accepted and the large jobs must be equally distributed among the machines in the optimal solution, with probability $P(\mathscr{E}_1) \cdot (1 - n^{-\Omega(n)}) = \Omega(1)$ the runtime will be in the order of $n^{\Omega(n)}$. The theorem statement follows since $E(T) \geq E(T \mid \mathscr{E}_1) \cdot P(\mathscr{E}_1) = n^{\Omega(n)}$ by the definition of expectation of a random variable. □

In the following subsection, we will show the efficiency of static HMP for optimising G^*_ε. Afterwards, we will do the same for ageing.

5.4.2 Hypermutations Are Efficient

We will start this subsection by proving some general statements about static HMP. This operator flips all the bits in a bit string successively and evaluates the fitness after each step. Unless an improvement is found first, each static HMP operation results in a sequence of n solutions sampled in the search space. This sequence of solutions contains exactly one bit string for each Hamming distance $d \in [n]$ to the initial bit string, i.e., it is a permutation of the bit positions. Moreover, the string which will be sampled at distance d is uniformly distributed among all the bit strings of distance d to the input solution.

The number of possible outcomes grows exponentially in the number of flipped bits until the $n/2$ th bit is flipped and afterwards it reduces at the same rate. Thus, the first and the last constant number of sampled strings are picked among a polynomially large subset of the search space. The following lemma bounds the probability that a particular target substring will be satisfied by all solutions sampled by the hypermutation operator between mth and $(n-m)$th bit-flips.

Lemma 8. *Let $x^i \in \{0,1\}^n$ be the i th bit string sampled by static HMP. For any arbitrary subset of indices $S \subset [n]$, let x^i_S be the subsequence of x^i which consists of bit positions in S. For any given target string $s^* \in \{0,1\}^{|S|}$ and integer $m > |S|$, the probability that $\forall i \in \{m, \ldots, n-m\}$, $x^i_S = s^*$ is at least $\left(\frac{m-|S|+1}{n-|S|+1}\right)^{|S|}$.*

Proof. Let S_A and S_D denote the set of positions where the initial subsequence x^0_S and s^* agree and disagree respectively and $ord(j) \in [n]$ denote the mutation step when the bit position $j \in S$ is flipped. Then, $\forall i \in \{m, \ldots, n-m\}, x^i_S = s^*$ holds if and only if $\forall j \in S_D, ord(j) \leq m$

(meaning that bit j should be flipped in the first m bit string samples) and $\forall j \in S_A, ord(j) \geq n-m$ (meaning that j gets flipped after the $(n-m)$th bit). Let $y_1, \cdots, y_{S_D}$ denote the positions in the set S_D and let E_k be the event that in the final permutation $y_1, y_2, \cdots, y_k$ are all within the first m positions (i.e., for all $j \leq k$, $ord(j) \leq m$). Hence, E_{S_D} is the event that $\forall j \in S_D$, $ord(j) \leq m$ and:

$$
\begin{aligned}
&P(E_{S_D}) = P(E_1 \cap E_2 \cap \cdots \cap E_{S_D}) = \\
&P(E_1) \cdot P(E_2 \mid E_1) \cdot P(E_3 \mid E_1 \cap E_2) \cdots P(E_{S_D} \mid E_1 \cap \cdots \cap E_{S_D}) = \\
&P(E_1) \cdot P(E_2 \mid E_1) \cdot P(E_3 \mid E_2) \cdots P(E_{S_D} \mid E_{S_D-1})
\end{aligned}
$$

where the first and the last equations hold because $E_{k-i} \subset E_k \ \ \forall i$ and the second by the definition of conditional probability. Given that $y_1, y_2, \cdots, y_k$ are in the final permutation such that $ord(y_i) \leq m$ for all $i \leq k$, the probability that also $ord(y_{k+1}) \leq m$ (i.e., $P(E_{k+1} \mid E_k)$) is $(m-k)/(n-k)$ because k positions amongst the first m are already defined. Thus, the probability that $ord(j) \leq m$ holds for all $j \in S_D$ is $P(E_{S_D}) = \prod_{k=0}^{|S_D|-1} \frac{m-k}{n-k}$. Similarly, if $ord(j) \leq m$ holds for all $j \in S_D$ and there are at least k different $j \in S_A$ such that $ord(j) \geq n-m$, the probability that there are at least $k+1$ different $j \in S_A$ such that $ord(j) \geq n-m$ is $(m-k)/(n-k-|S_D|)$. Thus, the probability that, $\forall j \in S_D$, $ord(j) \leq m$ and $\forall j \in S_A, ord(j) \geq n-m$ is $\left(\prod_{k=0}^{|S_D|-1} \frac{m-k}{n-k}\right) \left(\prod_{k=0}^{|S_A|-1} \frac{m-k}{n-k-|S_D|}\right) \geq \left(\frac{m-|S|+1}{n-|S|+1}\right)^{|S|}$. □

We can observe that the probability bounds we get from this lemma are polynomial if $|S|$ is a constant. Moreover, if both $|S| = \Theta(1)$ and $m = \Omega(n)$, we obtain probabilities in the order of $\Omega(1)$.

For G^*_ε, we would like to distribute two kinds of jobs evenly among the machines. While there is only a constant number of large jobs and $\Theta(n)$ small jobs, the previous lemma would provide reasonable probability bounds only for the large jobs. For the small jobs, we will make use of the fact that the configuration we want is not any configuration but an exact half and half split. Intuitively, it is obvious that if all the jobs start on one machine, at exactly the $(n/2)$th bit-flip the split will be half and half. However, as the starting distribution gets further away from the extremes, it becomes less clear when the split will exactly happen. Fortunately, the fact that the number of small jobs is large will work in our favour to predict the time of the split more precisely.

The following lemma establishes that larger the initial difference in the number of 0-bits and 1-bits is, the smaller the interval around the $(n/2)$th bit-flip where we observe a solution with equal number of 0-bits and 1-bits with overwhelmingly high probability.

Theorem 31. *If the input bit string of static HMP has* $\left(\frac{1}{2}+a\right)n$ *1-bits for some constant* $a>0$*, then with probability* $1-e^{-\Omega(nc^2)}$*, for any* $c=\omega(\sqrt{\log n/n})$ *such that* $c\in(0,a)$*, there exists a* $k\in\{n\frac{a-c}{2a-c},\ldots,n\frac{a}{2a-c}\}$ *such that the number of 1-bits in the k th solution sampled by HMP has exactly* $n/2$ *1-bits.*

Proof. Let Y_k be the total number of 1-bits flipped by HMP up to and including the kth mutation step. Given any parent string $x\in\{0,1\}^n$ with $((1/2)+a)n$ 1-bits, Y_k is a hypergeometrically distributed random variable with parameters n, $((1/2)+a)n$, k and expectation $E[Y_k]=((1/2)+a)\cdot k$.

Let X_k be the bit string sampled after the kth mutation step. It has $((1/2)+a)n-Y_k$ of x's 1-bits which have never been flipped and $k-Y_k$ of its 0-bits which are flipped to 1-bits. Thus, the number of 1-bits in X_k is $|X_k|_1=((1/2)+a)n+k-2Y_k$. Now, we can bound the probability that $|X_k|_1\geq n/2$ after a particular mutation step $k\geq\frac{n\cdot a}{2a-c}$ by using Chernoff bounds (see Theorem 8, page 36)[1]:

$$\begin{aligned}P(|X_k|_1>n/2)&=P\left(Y_k<\frac{k+a\cdot n}{2}\right)=P\left(Y_k<\frac{k+a\cdot n}{k+2a\cdot k}\cdot E[Y_k]\right)\\&\leq P\left(Y_k<\left(1-\frac{c}{2a+1}\right)\cdot E[Y_k]\right)\leq\exp\left(-\left(\frac{c}{1+2a}\right)^2\frac{E[Y_k]}{2}\right)\\&\leq\exp\left(-\left(\frac{c}{1+2a}\right)^2\frac{\left(\frac{1}{2}+a\right)\frac{n\cdot a}{2a-c}}{2}\right)=e^{-\Omega(nc^2)},\end{aligned}$$

where we used $k\geq\frac{n\cdot a}{2a-c}$ to obtain the first and the last inequality and Chernoff bounds to obtain the second. Then, using the same line of arguments, we bound $\mathrm{P}\{|X_k|_1<n/2\}$ at any given mutation step $k\leq n\frac{n\cdot(a-c)}{2a-c}$ by $e^{-\Omega(nc^2)}$:

$$\begin{aligned}P(|X_k|_1<n/2)&=P\left(Y_k>\left(1+\frac{a\cdot c}{(2a+1)(a-c)}\right)\cdot E[Y_k]\right)\\&\leq\exp\left(-\left(\frac{a\cdot c}{(2a+1)(a-c)}\right)^2\frac{E[Y_k]}{3}\right)\leq e^{-\Omega(nc^2)},\end{aligned}$$

where in the last inequality we used $a/(2a+1)=\Omega(1)$ and $(a-c)=o(1)$ only when $c=\Omega(1)$. Finally, we use the union bound and the $c=\omega\left(\sqrt{\frac{\log n}{n}}\right)$ assumption to prove our

[1] Note that the original statement of Theorem 8 refers to the sum of independent binary random variables, i.e., binomial distribution. However, it is shown in various works, including [26], that the theorem holds for the hypergeometric distribution as well.

claim:

$$P\left(\forall k \in \left[\frac{n\cdot(a-c)}{2a-c}, \frac{n\cdot a}{2a-c}\right] : |X_k|_1 \neq n/2\right)$$
$$\leq \sum_{j=\frac{n\cdot a}{2a-c}}^{n} P(|X_j|_1 > n/2) + \sum_{j=1}^{\frac{n\cdot(a-c)}{2a-c}} P(|X_j|_1 < n/2) \leq n\cdot e^{-\Omega(nc^2)} = e^{-\Omega(nc^2)}.$$

□

Now we have all the necessary tools to prove that the $(1+1)$ IA can solve G^*_ε efficiently. To prove our result, we will use the combination of Property 3 of Lemma 7, Lemma 8, and Theorem 31 to show that the probability of improvement is at least $\Omega(1)$ for any suboptimal solution and then obtain our bound on the expected runtime by Property 1 of Lemma 7, which indicates that there can be at most $O(n)$ improvements before the optimum is found. The heart of the proof of the following theorem is to show that from any local optimum, static HMP identifies the global optimum with constant probability unless an improvement is found first.

Theorem 32. *The $(1+1)$ IA optimises the G^*_ε class of instances in $O(n^2)$ expected fitness function evaluations.*

Proof. According to Property 3 of Lemma 7, for any suboptimal solution x, either there are at least a linear number of small jobs that can be moved from the fuller machine to the emptier machine to improve the makespan or the number of small jobs in each machine differs from $\frac{n-s}{2}$ by a linear factor. For the former case, in the first mutation step the probability that static HMP flips one of the bits which correspond to the small jobs in the fuller machine is at least $\Omega(n)/n = \Omega(1)$. Since such a mutation creates an improved solution, static HMP stops and the overall improvement probability is at least $\Omega(1)$.

For the latter case, the number of small jobs in one of the machines is at least $\left(\frac{1}{2}+a\right)(n-s)$ for some positive constant a. Using the notation of Theorem 31 with a and $c = 1/(a\log n)$, with overwhelming probability the number of small jobs assigned to each machine will be exactly $(n-s)/2$ at some bit-flip between $(n-s)\frac{(a)-1/(a\log n)}{(2/a)-1/(a\log n)} = (n-s)\frac{1-(1/\log n)}{2-1/\log n}$ and $(n-s)\frac{a}{(2a)-1/(a\log n)} = (n-s)\frac{1}{2-1/(\log n)}$. According to Lemma 8, the probability that the large jobs are distributed evenly throughout this interval is at least:

$$\left(\frac{(n-s)\frac{1-(1/\log n)}{2-(1/\log n)} - s + 1}{n-s+1}\right)^s = 2^{-s}(1-o(1)) = \Omega(1).$$

Therefore, with constant probability either the optimal solution will be found or the hypermutation will stop due to an improvement. Since according to Property 1 of Lemma 7 there are at most $O(n)$ different makespan values and the probability of improvement is at least $\Omega(1)$ for all sub-optimal solutions, the optimum is found in expected $O(n)$ iterations. Given that static HMP evaluates at most n solutions in each iteration, our claim follows. □

Since the worst case instance for the $(1+1)$ EA presented in [90] is an instance of G^*_ε with $s = 2$, the following corollary holds.

Corollary 4. *The $(1+1)$ IA optimises P^*_ε in $O(n^2)$ expected fitness function evaluations.*

5.4.3 Ageing Is Efficient

In this section we will show that the $(\mu+1)$ EA$^{\text{ageing}}$ can optimise the G^*_ε instances efficiently. Our approach to prove the following theorem is to first show that if a solution where the large jobs are equally distributed is sampled at initialisation, then it takes over the population and the $(\mu+1)$ EA$^{\text{ageing}}$ quickly finds the optimum. The contribution of ageing is considered afterwards to handle the case when no such initial solution is sampled. In such cases we will show that whenever the algorithm gets stuck on a local optima, it will reinitialise the whole population after τ iterations and sample a new population. Before we move to the main results, we will prove a helper lemma.

Lemma 9. *For any* PARTITION *instance with L local optima with different makespans, the expected time until the $(\mu+1)$ EA^{ageing} with $\mu = O(\log n)$ and $\tau = O(n^{1+c})$ for any constant c either reinitialises the population or finds the optimum is at most $L(2en^2+\tau)(1+o(1))$.*

Proof. Let X_t denote the solution with the smallest makespan ($f(X_t)$) in the population at time t. We will first bound the expected time until either the population reinitialises or given that $f(\ell_i) > f(X_t) \geq f(\ell_{i+1})$, a solution with makespan strictly smaller than $f(\ell_{i+1})$ is sampled.

According to Lemma 6, in $2en^2(1+o(1))$ iterations we have $f(X_t) \leq f(\ell_{i+1})$ with probability at least $1-n^{-\Omega(n^c)}$.

Let k_t denote the number of individuals in the population that ties for the minimal makespan value. If $f(X_t) = f(\ell_{i+1})$, then at any iteration t, with probability at least $(k/\mu)(1-1/n)^n \geq (k/\mu)(1-1/n)e^{-1}$, SBM creates a copy of one of the best solutions, thus, $k_{t+1} = k_t + 1$. Therefore, the expected time until either we sample a solution with makespan smaller than $f(\ell_{i+1})$ or increase the k_t by one is at most $(\mu/k)(1-1/n)^{-1}e$. Since $k_t \in [\mu]$, the expected time until the population consists only of solutions with makespan $f(\ell_{i+1})$ or a better makespan being sampled is at most $\sum_{k=1}^{\mu}(\mu/k)(1-1/n)^{-1}e = (1+o(1))e\mu\ln\mu$. This

occurs in less than $\tau/2$ iterations with probability at least $1-2^{-\Omega(\frac{\tau}{\mu\log\mu})} = 1-2^{-\Omega(n^{1+c'})}$ for some constant $c' > 0$ due to the iterative use of Markov's inequality. According to Lemma 3 (Chapter 4, page 63), unless the best fitness is improved, in at most $144\mu^3$ steps the population reaches the same age. Using Markov's inequality iteratively, we can bound the probability that this event happens in at most $\tau/2$ iterations by $1-2^{-\Omega(\tau/\mu^3)} = 1-2^{-\Omega(n^{1+c''})}$ for some constant $c'' > 0$. Thus, in $2en^2(1+o(1))+\tau$ iterations, if we sample a solution with makespan smaller than $f(\ell_{i+1})$, the population consists of solutions with the same fitness and the same age with overwhelming probability which in turn implies that the population is reinitialised unless an improvement is found before the age of the individuals reach τ. The total failure probability starting from a an arbitrary X_t s.t. $f(\ell_i) > f(X_t) \geq f(\ell_{i+1})$ until either the population reinitialises or a solution with $f(X_t) < f(\ell_{i+1})$ is in the order of $n^{-\Omega(n^{c/2})} + 2^{-\Omega(n^{1+c'})} + 2^{-\Omega(n^{1+c''})} = 2^{-\Omega(n^{1+c^*})}$ for some constant $c^* > 0$. Since there are $L < 2^n$ local optima with distinct makespan, by the union bound the probability of losing the best individual before the optimum is sampled for the first time is at least $1-2^{-\Omega(n^{1+c^*})} \cdot L = 1-2^{-\Omega(n^{c^*})}$. Since there are exactly L distinct local optima than can be overcome before finding the optimum, starting from any population in $L \cdot (2en^2(1+o(1))+\tau)$ iterations either the optimum is found or the reinitialization occurs with overwhelming probability. Since in the case of a failure we can repeat the argument again, in $1+o(1)$ expected repetitions one of the desired events occur. □

We will now show that with probability at least $p_{success} = \Omega(1)$, the optimal solution will be found in $O(n^{3/5})$ iterations starting from a randomly initialised population. Since Lemma 9 and Property 2 of Lemma 7 imply that the expected time until either the optimum is found or the population is reinitialised is in the order of $O(n^2+\tau)$, in expected $O(n^{3/5}) + O(n^2+\tau)/p_{success} = O(n^2+\tau)$ iterations the optimum is found.

Theorem 33. *The $(\mu+1)$ EA^{ageing} optimises the G^*_ε class of instances in $O(n^2+\tau)$ steps in expectation for $\tau = \Omega(n^{1+c})$ for any arbitrarily small constant c and $\mu = O(\log n)$.*

Proof. With probability $\binom{s}{s/2}2^{-s} = \Omega(1)$, a randomly initialised solution assigns an equal number of large jobs to each machine. We will refer to such solutions as *good* solutions. For any initialised individual, the expected number of small jobs assigned to each machine is $(n-s)/2$. By Chernoff bounds, the probability that fewer or more than $(n/2) \pm n^{3/5}$ small jobs are assigned to one machine is $e^{-\Omega(n^{1/5})}$. We will now show that if there is at least one good individual at initialisation, then in $O(\mu\log\mu)$ generations the whole population will consist of good solutions with at least constant probability.

For a *bad* initial solution, the difference between the number of large jobs in M_1 and M_2 is at least 2 by definition. With $1-e^{-\Omega(n^{1/5})}$ probability, the initial number of small

jobs assigned to each machine does not differ by more than $O(n^{3/5})$ (event $\mathscr{E}_2$) . Thus, the difference between the loads of the machines is at least $\Omega(1)$ for a bad solution, while less than $O(n^{-2/5})$ for a good one, both with overwhelmingly high probabilities. In order for a bad solution to have a better fitness than a good one, it is necessary that at least $\Omega(n)$ small jobs are moved from the fuller machine to the emptier machine which takes at least $\Omega(n/\log n)$ time with overwhelmingly high probability since the probability of SBM flipping more than $\log n$ bits is exponentially small.

If there are k good solutions in the population, the probability that one of them is selected as parent is k/μ and the probability that SBM does not flip any bits and yields a copy is $(1-(1/n))^n > 1/3$ (for $n \geq 6$). Thus, the expected time until the good solutions take over the population is at most $3\mu\sum_{k=1}^{\mu}(1/k) \leq 3\mu\ln\mu$. The probability that the good solutions take over the population in at most $O(n^{3/5})$ iterations (before any bad solution is improved to a fitness value higher than the initial fitness of a good solution) is by Markov's inequality in the order of $1-O((\log n)(\log\log n)/n^{3/5})$. Thus, if the initial population contains at least one good solution, it takes over the population with probability $1-o(1)$. Moreover, since the discrepancy (the difference between the loads of the machines) of the accepted good solutions will not increase above $\Omega(n^{-2/5})$, SBM needs to move a linear number of small jobs in a single iteration to create a bad solution from a good one, which does not happen with probability at least $1-n^{-\Omega(n)}$.

Starting from a suboptimal good solution, the fitness improves by moving any small job from the fuller machine to the emptier machine without moving any of the large jobs. The probability of such an event is at least $(n-s)/2n\cdot(1-1/n)^{n-1} = \Omega(1)$ considering that there are at least $(n-s)/2n$ small jobs on the fuller machine. Since the difference between the number of small jobs is in the order $O(n^{3/5})$, the small jobs will be distributed exactly evenly and the global optimum will be reached in expected $O(n^{3/5})$ iterations. Using Markov's inequality (see Theorem 6, page 35), we can bound the probability that the global optimum will be reached in $O(n^{3/5})$ iterations by $\Omega(1)$.

Since all necessary events described for finding the optimum have probability $\Omega(1)$, the probability that they occur in the same run is at least in the order of $\Omega(1)$ as well. Thus, all that remains is to find the expected time until the population consists of μ randomly initialised individuals given that one of the necessary events fails to occur and the global optimum is not found. This expectation is at most $L\cdot(2en^2+\tau)(1+o(1))$ due to Lemma 9 with $L=\Omega(1)$ which is in turn due to Property 2 of Lemma 7. Thus, the expectation is $(1/p_{success})\cdot O(n^2+\tau)+O(n^{3/5}) = O(n^2+\tau)$. □

Clearly the following corollary holds as P^*_ε is an instance of G^*_ε.

Corollary 5. *The* $(\mu+1)$ *EA*^{ageing} *optimises* P^*_ε *in* $O(\mu n^2+\tau)$ *steps in expectation for* $\tau=\Omega(n^{1+c})$ *and* $\mu=O(\log n)$.

5.5 $(1+\varepsilon)$ Approximation Ratio Analysis

In this section we will show that both the $(1+1)$ IA and the $(\mu+1)$ EA$^{\text{ageing}}$ are polynomial time approximation schemes, i.e., they guarantee solutions with makespan at most a $(1+\varepsilon)$ factor worse than the optimal solution in expected time polynomial in the number of jobs n and exponential only in $1/\varepsilon$. Throughout this section we will refer to the largest $s:=\lceil 2/\varepsilon\rceil-1\leq n/2$ jobs of the instance as *large jobs* and the rest as *small jobs*. Before we give the main results, we present the properties of the PARTITION problem which will be used in the proofs.

Lemma 10. *Let I be any instance of the* PARTITION *problem with optimal makespan value* y^* *and the set of local optima* $\mathscr{L}$. *Then,*

Property 1: *For all* $j\in\{s+1,\ldots,n\}$, $p_j\leq\frac{\varepsilon}{2}\sum_{i=1}^n p_i$.

Property 2: *If there is at least one small job assigned to the fuller machine of a solution* $x\in\mathscr{L}$, *then* x *is a* $(1+\varepsilon)$ *approximation.*

Property 3: *If* $\sum_{i=s+1}^n p_i\geq\frac{1}{2}\sum_{i=1}^n p_i$, *then all* $x\in\mathscr{L}$ *are* $(1+\varepsilon)$ *approximations.*

Property 4: $|\{y\in((1+\varepsilon)\cdot y^*,\infty):\exists x\in\mathscr{L}\ \ s.t.\ \ f(x)=y\}|\leq 2^{2/\varepsilon}$.

Proof. Let $W:=\sum_{i=1}^n p_i$ denote the sum of the processing times of all jobs. We will prove the first statement by contradiction. Let us assume that $p_{s+1}>\frac{\varepsilon}{2}W$. Thus, all p_i for $i\in\{1,\ldots,s+1\}$ would satisfy $p_i>\frac{\varepsilon}{2}W$, which implies $\sum_{i=1}^{s+1}p_i>\lceil\frac{2}{\varepsilon}\rceil\frac{\varepsilon}{2}W>W$, a contradiction.

The second statement follows from the first one, since if moving a small job from the fuller machine to the emptier one does not improve the makespan, then the makespan is at most $W/2+(1/2)(\varepsilon/2)W$. A trivial lower bound on the optimum, y^*, is $W/2$, therefore the approximation ratio is at most $\left(\frac{W}{2}+\frac{1}{2}\frac{\varepsilon}{2}W\right)/(W/2)\leq(1+\varepsilon)$.

For the third statement, if $\sum_{i=s+1}^n p_i\geq\frac{1}{2}\sum_{i=1}^n p_i$, then any local optimum must have some small jobs on the fuller machine since the sum of the processing times of large jobs is smaller than the sum of the processing times of the small jobs. Thus, the claim follows from the second statement.

The last statement indicates that there are at most $2^{\varepsilon/2}$ distinct makespan values among local optima which are not $(1+\varepsilon)$ approximations. The second statement states that if a local optimum is not a $(1+\varepsilon)$ approximation, then it cannot have a small job on its fuller machine.

Thus, in such a solution the makespan is exclusively determined by the configuration of the large jobs on the fuller machine. Since there are at most $\frac{\varepsilon}{2}+1$ large jobs, the number of different configurations is $2^{\frac{\varepsilon}{2}+1}$. Since the complement of each configuration has the same makespan, the statement follows. □

5.5.1 Hypermutations Find $(1+\varepsilon)$ Approximation Ratios

Theorem 34 will show that the $(1+1)$ IA can efficiently find arbitrarily good constant approximations to any PARTITION instance. Before we state our main result, we introduce the following helper lemma.

Lemma 11. *Let $x^i \in \{0,1\}^n$ be the i th bit string sampled by static HMP with the input bit string 0^n. For any arbitrary subset of indices $S \subset [n]$ and a function $f(x) := \sum_{j\in S} x_j w_j$ with a set of non-negative weights w_j, $E[f(x^i)] = \frac{i}{n}\sum_{j\in S} w_j$.*

Proof. By linearity of expectation, $E[f(x^i)] = \sum_{j\in S} E[x^i_j] w_j$, where x^i_j is the jth bit position of the ith bit string sampled by static HMP. Considering that the initial bit value is zero, $x^i_j = 1$ with probability i/n. Thus the expectation $E[x^i_j]$ is i/n for all j and the claim follows from moving the multiplicative factor of i/n out of the sum in the expected function value $\sum_{j\in S}(i/n)w_j$. □

We now state the main result of this section.

Theorem 34. *The $(1+1)$ IA finds a $(1+\varepsilon)$ approximation to any instance of* PARTITION *in at most $n(\varepsilon^{-(2/\varepsilon)-1})(1-\varepsilon)^{-2}e^3 2^{2/\varepsilon} + 2n^3 2^{2/\varepsilon} + 2n^3$ fitness function evaluations in expectation for any $\varepsilon = \omega(n^{-1/2})$.*

Proof. Let X_t denote the current solution of the $(1+1)$ IA. We will assume that the algorithm stops as soon as it finds a $(1+\varepsilon)$ approximation. The expected number of iterations where $X_t \notin \mathscr{L}$ is at most $2n^2(2^{2/\varepsilon}+1)$ due to Property 4 of Lemma 10 and Lemma 6.1 (i.e., from any non-locally optimal solution, in at most $2en^2$ generations, a better locally optimal solution will be found according to Lemma 6.1 and the number of local optima which are not $(1+\varepsilon)$ is at most $2^{2/\varepsilon}$. Hence, the total number of non-local optima would be $2n^2(2^{2/\varepsilon}+1)$). Moreover, due to Property 3 of Lemma 10 and Lemma 6, any of these local optima would be a $(1+\varepsilon)$ approximation if $\sum_{i=s+1}^{n} p_i \geq \frac{1}{2}\sum_{i=1}^{n} p_i$ (which makes the algorithm stop). Therefore, for the rest of the proof we will assume that $\sum_{i=s+1}^{n} p_i < \frac{1}{2}\sum_{i=1}^{n} p_i$.

Now, we want to bound the expected number of iterations where $X_t \in \mathscr{L}$. Let h denote the configuration of only the large jobs with the minimum makespan H. Note that this configuration might be different than the configuration of the large jobs in the optimal

solution since it does not take the contribution of the small jobs to the makespan into account. However, $H \leq y^*$ since introducing the small jobs cannot improve the optimal makespan. Since both h and its complementary bit string have the same makespan, w.l.o.g., we will assume that the fuller machines of X_t and h are both M_1.

According to Lemma 8, the $s \leq 2/\varepsilon$ large jobs are assigned in the same way as in h between the $n(\varepsilon-\varepsilon^2)$th and $n-n(\varepsilon-\varepsilon^2)$th bit-flips with probability at least $\left(\varepsilon-\varepsilon^2\right)^{2/\varepsilon} e^{-1}$. Due to Property 2 of Lemma 10, X_t does not have any small job on M_1. Since $\sum_{i=s+1}^{n} p_i < W/2$, by the $n(\varepsilon-\varepsilon^2)$th bit-flip the expected total processing time of small jobs moved from M_2 to M_1 is at most $(\varepsilon-\varepsilon^2)W/2$ according to Lemma 11. Due to Markov's inequality (see Theorem 6, page 35), with probability at least $1-((\varepsilon-\varepsilon^2)W/(\varepsilon W)) = \varepsilon$, the sum of the processing times of the moved jobs is at most $\varepsilon W/2$ and the makespan of the solution is at most $H+\varepsilon W/2$. Since $y^* \geq H$ and $y^* \geq W/2$, $(H+\varepsilon W/2)/y^* \leq (1+\varepsilon)$. This implies that with probability:

$$p_{\approx} \geq \frac{(\varepsilon-\varepsilon^2)^{2/\varepsilon}}{e}\varepsilon = \frac{\varepsilon^{(2/\varepsilon)+1}}{e(1-\varepsilon)^{-2/\varepsilon}} = \frac{\varepsilon^{(2/\varepsilon)+1}(1-\varepsilon)^2}{e(1-\varepsilon)^{-((1/\varepsilon)-1)2}} \geq \frac{\varepsilon^{(2/\varepsilon)+1}(1-\varepsilon)^2}{e^3},$$

a $(1+\varepsilon)$ approximation is found unless an improvement is obtained before. Therefore, due to Property 4 of Lemma 10, the expected number of iterations such that $X_t \in \mathscr{L}$ is at most $(1/\varepsilon^{(2/\varepsilon)+1})(1-\varepsilon)^{-2}e^3 2^{2/\varepsilon}$. Considering that static HMP evaluates at most n solutions per iteration, the expected number of fitness evaluations before a $(1+\varepsilon)$ approximation is found is at most $n(\varepsilon^{-(2/\varepsilon)-1})(1-\varepsilon)^{-2}e^3 2^{2/\varepsilon} + 2n^3 2^{2/\varepsilon} + 2n^3$. □

In [90], it is shown that the $(1+1)$ EA achieves the same approximation ratio as the $(1+1)$ IA while the leading term in the upper bound on its expected runtime is $n\log(1/\varepsilon)2^{2/\varepsilon}$. Although this result shows the power of EAs in general, for the $(1+1)$ EA to achieve this, a problem-specific restart schedule (where each run, or restart, takes $O(n\log(1/\varepsilon))$ expected iterations) has to be decided in advance where either the length of each run or the number of parallel runs depends on the desired approximation ratio. Consequently, such an algorithm loses application generality (e.g., if the ideal restart schedule for any constant ε is used, it would fail to optimise efficiently even ONEMAX or LEADINGONES since optimising these functions requires $\Omega(n\log n)$ and $\Omega(n^2)$ expected fitness function evaluations respectively). In any case, if knowledge that the tackled problem is PARTITION was available, then using the $(1+1)$ EA would not be ideal in the first place. On the other hand, the AIS does not require knowledge that the tackled problem is PARTITION, neither that a decision on the desired approximation ratio is made in advance for it to be reached efficiently.

5.5.2 Ageing Finds $(1+\varepsilon)$ Approximation Ratios

Theorem 35 shows that the $(1+1)$ EA$^{\text{ageing}}$ can find $(1+\varepsilon)$ approximations. Its proof follows the strategy used to prove that RLS and the $(1+1)$ EA achieve a $(1+\varepsilon)$ approximation if an appropriate restart strategy is put in place [90, 68]. The correct restart strategy which allows RLS and the $(1+1)$ EA to approximate PARTITION requires a fair understanding of the problem. For instance, as discussed at the end of the previous subsection, restarts have to happen roughly every $n\ln(1/\varepsilon)$ iterations to achieve the smallest upper bound. However, the same restart strategy can cause exponential runtimes for easy problems if the expected runtime is in the order of $\Omega(n^{c'})$. Our parameter choices for the ageing operator are easier in the sense that no (or very limited) problem knowledge is required to decide upon its appropriate value and the choice mostly depends on the variation operator of the algorithm rather than on the problem at hand. For example, if the ageing operator is applied with RLS, a value of $\tau = n^{1+c''}$ guarantees that with overwhelming probability no restart will ever happen unless the algorithm is trapped on a local optimum independent of the optimisation problem. If $\text{RLS}_{1,2,\ldots,k}$ was applied, then a value of $\tau = n^{k+c'''}$ allows us to draw the same conclusions. Concerning the $(1+1)$ EA, once the user has decided on their definition of local optimum (i.e., the maximum neighbourhood size for which the algorithm is expected to find a better solution before giving up), then the appropriate values for τ become obvious. In this sense, the algorithm using ageing works out *by itself* when it is stuck on a local optimum. We will show our result for the more restricted case of $\mu = 1$ which suffices for the purpose of showing that ageing is efficient for the problem. In particular, the proof is considerably simplified with $\mu = 1$ since we can use the success probability proven in [90] without any modification. Furthermore, without proof, the upper bound on the runtime for arbitrary μ increases exponentially with respect to μ/ε. Our proof idea is to use the success probability of the simple $(1+1)$ EA without ageing and show that ageing automatically causes restarts whenever the $(\mu+1)$ EA$^{\text{ageing}}$ fails to find the approximation by using only single bit-flips. Hence, a problem specific restart strategy is not necessary to achieve the desired approximation.

Theorem 35. *The* $(1+1)$ *EA*^{ageing} *with* $\tau = \Omega(n^{1+c^*})$ *for any arbitrarily small constant* $c^* > 0$ *finds a* $(1+\varepsilon)$ *approximation to any instance of* PARTITION *in at most:*

$$(2en^2+\tau)2^{(e\log e+e)\lceil 2/\varepsilon\rceil \ln(4/\varepsilon)+\lceil 4/\varepsilon\rceil -1}(1+o(1)),$$

fitness function evaluations in expectation for any $\varepsilon \geq 4/n$.

Proof. We make use of Theorem 3 in [90] which lower bounds the probability that the $(1+1)$ EA$^{\text{ageing}}$ finds the $(1+\varepsilon)$ approximation in $\lceil en\log 4/\varepsilon\rceil$ iterations by $p_{success} :=$

$2^{-(e\log e+e)\lceil 2/\varepsilon\rceil \ln(4/\varepsilon)-\lceil 2/\varepsilon\rceil}$ for any $\varepsilon \geq 4/n$. Note that, since $\tau > \lceil en\log 4/\varepsilon\rceil$, the ageing operator does not affect this probability.

Since we do not distinguish between an optimal solution and a $(1+\varepsilon)$ approximation, Lemma 9 and Property 4 of Lemma 10 imply that in at most $2^s(2en^2+\tau)(1+o(1))$ expected time either an approximation is found or a reinitialisation occurs. Multiplying with $p_{success}^{-1}$, we obtain $(2en^2+\tau)2^{(e\log e+e)\lceil 2/\varepsilon\rceil \ln(4/\varepsilon)+\lceil 4/\varepsilon\rceil-1}(1+o(1))$. □

5.6 Conclusion

In this chapter, we analysed AIS operators for an NP-hard problem. To the best of our knowledge, this is the first time that polynomial expected runtime guarantees of solution quality have been provided concerning AISs for a classical combinatorial optimisation problem. We presented a class of instances of PARTITION to illustrate how static HMP and ageing can efficiently escape from local optima and provide optimal solutions where the SBMs used by traditional EAs get stuck for exponential time. Then, we showed how this capability allows static HMP to achieve arbitrarily good $(1+\varepsilon)$ approximations for any instance of PARTITION in an expected time that is polynomial in the number of jobs n and exponential only in $1/\varepsilon$, i.e., in expected polynomial time for any constant ε. In contrast to standard EAs and RLS that require a problem specific number of parallel runs or a restart scheme to achieve such approximations, the AISs find them in a single run. The two results are achieved in different ways. The ageing operator locates more promising basins of attraction by restarting the optimisation process after implicitly detecting that it has found a local optimum. Static HMP finds improved approximate solutions efficiently by performing large jumps in the search space. Naturally, the proof strategy would also apply to the complete standard Opt-IA AIS [20, 15] if the ageing parameter is set large enough, i.e., $\tau = \Omega(n^{1+c})$ for any arbitrarily small constant c.

In the previous and the present chapters we have proven for standard benchmark problems and a classic combinatorial optimisation problem that AISs can escape from local optima faster than EAs, albeit being slower in the exploitation (i.e., hill-climbing) phases. In the following chapters we will propose algorithmic modifications that allow the AISs to be competitive with EAs also in the hill-climbing phases.

Part III

Fast Opt-IA

Chapter 6

Inversely Proportional Hypermutations

6.1 Introduction

In the previous chapters, it was argued that static HMPs have good exploration capabilities by showing that they can more efficiently escape from local optima of both standard multimodal benchmark functions and classical combinatorial problems than traditional EAs. On the other hand, they were also proved to be relatively slow in the exploitation phase compared to the standard bit mutations used by EAs. The reason is that, as the algorithm gets closer to the optimum and the improvement probability decreases, it becomes more and more likely that static HMP wastes *cn* fitness function evaluations every time it fails to find an improvement in the first mutation step. In this chapter, we consider whether the inversely proportional hypermutations from the literature, used typically in AISs such as Clonalg [23] and Opt-IA [19] (see Subsection 2.3.3, page 23), may overcome the issue. Since the idea behind inversely proportional hypermutations is to dynamically reduce the mutation rate the closer a candidate solution gets to the global optimum, the hope is that the amount of wasted fitness function evaluations is significantly reduced when the improvement probabilities are small. This may lead to faster runtimes in the exploitations phases.

We start in Section 6.2 by discussing the problems identified in the literature concerning the inversely proportional hypermutations used in the previous works which lead to inefficient algorithms even for the simple ONEMAX function. In particular, we argue why FCM is essential for inversely proportional hypermutations to be efficient. To this end, in Section 6.3, we define three inversely proportional MPs (mutation potentials) that work in harmony with FCM. Two of the MPs are adapted from the literature, while the third one is newly introduced.

In Section 6.4, we evaluate the best achievable speed-ups by analysing the operators for standard unimodal benchmark functions in the ideal situation where the location of the optimum is known. A result of this first analysis is that a mutation potential that increases

exponentially with the Hamming distance to the optimum is the most promising out of the three considered ones (it provides the largest speed-ups). Hence, we use this mutation potential, called M_{expoHD}, in the rest of the chapter.

In Section 6.5, we propose an AIS which uses inversely proportional HMP and ageing to be applied in practical applications where the optimum is not known. The idea behind the algorithm is that instead of adjusting the mutation potential with respect to the distance to the global optimum, we adjust it according to the best seen local optimum. Hence, the more the search space is explored, the better the ideal behaviour of the inversely proportional hypermutations is approximated through the discovery of better and better local optima. Although we prove that such algorithm can escape the local optima of CLIFF, we show that in general, such a strategy does not produce the desired effect, for instance for the TWOMAX bimodal benchmark function. Since the mutation rate decreases with the distance to the best found local optimum, the algorithm may encounter difficulties at identifying new promising optima located in other directions (i.e., the mutation rate increases as the algorithm makes progress). To this end, in Section 6.6, we define a *symmetric* inversely proportional HMP operator that decreases the mutation potential with respect to the distance to the best seen local optimum and uses the same rate of decrease in all other directions. We prove the effectiveness of our strategy and subsequent speed-ups over static HMPs for the TWOMAX benchmark function while showing that the local optima of CLIFF can still be escaped.

6.2 Limitations of the Inversely Proportional Hypermutations from the Literature

Inversely proportional hypermutations have been used both in Opt-IA [19] and in Clonalg [23] AISs. However, previous theoretical analyses have highlighted various problems in their application even when the fitness value of the global optimum is known [93, 52].

The first runtime analyses of inversely proportional hypermutation operators included the performance analyses of two hypermutation operators for optimising ONEMAX function [93]. The first operator flips every bit with probability $\frac{H(x,\text{opt})}{n}$ where n is the problem size, opt is the optimum and $H(x,\text{opt})$ is the Hamming distance of x and opt. Using this probability which has a linear decay (i.e., the probability of flipping each bit in the search point x linearly depends on the Hamming distance between x and opt), the operator flips in expectation exactly the number of bits that maximises the probability of reaching the optimum in the next step. For the second operator, each bit is flipped with a probability which has an exponential decay and is defined as $e^{-\rho v}$, where $v = \frac{\text{ONEMAX}(x)}{f(\text{opt})}$ (the normalised fitness). This is the

mutation probability used in Clonalg. It was shown that even when the optimum is known, an algorithm employing any of these two mutation operators will require exponential time to optimise ONEMAX w.o.p. The reason is that the initial random solutions have roughly $n/2$ 1-bits, hence have very high mutation rates. Such rates hardly allow an algorithm to make progress with any reasonable probability. Therefore, the behaviour of the mutation operator leads to very inefficient algorithms even for the simple ONEMAX function when the optimum is assumed to be known. In the following sections we will show how by stopping at the first constructive mutation, FCM can solve the problem affected by the inversely proportional hypermutation of Clonalg.

On the other hand, the inversely proportional HMP proposed for Opt-IA suffers from a different problem [52]. It uses a mutation potential of $M = \lceil (1 - f(\text{opt})/f(x)) \rceil cn$ for minimisation problems, where $f(\text{opt})$ is the best known fitness [52]. In an analysis for ZEROMIN, such a mutation potential was shown not to decrease below $cn/2$ with cn being the highest possible mutation potential [52]. Hence, the mutation potential does not decrease as desired. As a result, it is unable to asymptotically improve the runtime of static hypermutation for neither ONEMAX nor LEADINGONES [52].

The ideal behaviour of the inversely proportional hypermutation is that the mutation rate is minimal in proximity of the global optimum and should increase as the difference between the fitness of the global optimum and that of the candidate solution increases. However, achieving such behaviour in practice may be problematic because the fitness of the global optimum is usually unknown. As a result, in practical applications some information about the problem is used to identify bounds on (or estimates of) the value of the global optimum[1]. Thus, the closer is the estimate to the actual value of the global optimum, the closer should the behaviour of the inversely proportional hypermutation be to the desired one. On the other hand, if the bound is much higher (i.e., for a maximisation problem) than the true value, then there is a risk that the mutation rate is too high in proximity of the global optimum, i.e., the algorithm will struggle to identify the optimum. In the following section, we introduce three inversely proportional hypermutation MPs which do not suffer from the problems described above, i.e., the mutation rate decreases as the optimum is approached and FCM allows for progress when the rate is high.

[1] Alternatively, the fitness of the best candidate solution is sometimes used as a reference point and the mutation rate of the rest of the population is inversely proportional to the best.

6.3 Inversely Proportional Mutation Potentials with FCM

In this section, we introduce three new inversely proportional mutation potentials to be used in conjunction with FCM. Two of these MPs are inspired by the inversely proportional hypermutation rates proposed in the literature, while the third one is newly proposed by us based on the performance of the first two.

6.3.1 Hamming Distance Based Linear Decrease

Zarges analysed an inversely proportional hypermutation operator where the probability of flipping each bit increases linearly with the Hamming distance to the optimum (or its best available estimate) [93]. Precisely, this operator flips each bit with probability $\min\{H(x,\text{best})\}/n$ where n is the size of the problem and $\min\{H(x,\text{best})\}$ is the minimum Hamming distance of the current point to a best individual. As the expected number of bit-flips is $\min\{H(x,\text{best})\}$ during each execution of a mutation operator, a mutation potential inspired by this inversely proportional hypermutation is:

$$M_{\text{linHD}}(x) := \min\{H(x,\text{best})\}.$$

6.3.2 Fitness Difference Based Exponential Decrease

In Clonalg's inversely proportional hypermutation operator, the mutation probability decreases as an exponential function of the fitness of the current solution [23]. Precisely, each bit flips with probability $e^{-\rho\cdot v}$ where v is the normalised fitness value and ρ is a decay parameter that regulates the speed at which the mutation rate decreases. Since we consider maximisation problems, we use $v = \frac{f(x)}{f(\text{best})}$ as suggested by [93] where $f(\text{best})$ is the best known fitness value. Using this mutation operator as a mutation potential (i.e., its expected value) gives $M = n\cdot e^{-\rho v}$. According to both practical and theoretical results [93], a reasonable value for ρ is $\ln n$. We call this mutation potential $\text{M}_{\text{expoF(x)}}$ and define it as:

$$M_{\text{expoF(x)}}(x) := \left\lfloor n^{1-\frac{f(x)}{f(\text{best})}} \right\rfloor.$$

6.3.3 Hamming Distance Based Exponential Decrease

It is well understood that fitness-proportional selection is sensitive to the difference between the fitness values of candidate solutions, i.e., it struggles to distinguish between solutions that have similar, yet different, fitness values [76, 77, 89, 66]. For this reason, nowadays selection operators that rank individuals by fitness (e.g., tournament, linear ranking, comma

Table 6.1 The expected runtimes achieved by the $(1+1)$ IA with different MPs and $(1+1)$ EA

Algorithms	ONEMAX	LEADINGONES
$(1+1)$ EA	$\Theta(n\log n)$ [31]	$\Theta(n^2)$[31]
$(1+1)$ IA with M = n	$\Theta(n^2\log n)$	$\Theta(n^3)$
$(1+1)$ IA using M_{linHD}	$\Theta(n^2)$	$\Theta(n^3)$
$(1+1)$ IA using $M_{expoF(x)}$	$O(n^{(3/2)+\varepsilon}\log n)$, $\Omega(n^{(3/2)-\varepsilon})$	$O(n^3/\log n)$, $\Omega(n^{(5/2)+\varepsilon})$
$(1+1)$ IA using M_{expoHD}	$O(n^{(3/2)+\varepsilon}\log n)$, $\Omega(n^{(3/2)-\varepsilon})$	$O(n^{(5/2)+\varepsilon}/\ln n)$, $\Omega(n^{(9/4)-\varepsilon})$

selection) are generally preferred. We also consider a measure that avoids using the fitness values directly. To this end, we suggest a mutation potential which is similar to $M_{expoF(x)}$ with the exception that it uses the normalised Hamming distance to the best estimate rather than the normalised fitness. We call this mutation potential M_{expoHD} and define it as $M_{expoHD}=$ $n\cdot e^{-\rho\frac{n-H(x,\text{best})}{n}}$ where n is the problem size (the maximum Hamming distance between any two points), $H(x,\text{best})$ is the Hamming distance to the best known solution and ρ is the decay of the mutation potential. For the choice of $\rho=\ln n$, we get:

$$M_{\text{expoHD}}(x) := \left\lfloor n^{\frac{H(x,\text{best})}{n}} \right\rfloor.$$

6.4 Analysis in Ideal Conditions

In this section, we evaluate the performance of the proposed inversely proportional MPs, assuming the optimum is known (i.e., $f(\text{best})=f(\text{opt})$). Under this assumption, the operators exhibit their behaviour in ideal conditions. Our aim is to evaluate what speed-ups can be achieved in this ideal situation compared to the well-studied static HMP. To achieve such comparisons we perform runtime analyses of the $(1+1)$ IA using the inversely proportional HMP on the ONEMAX and LEADINGONES unimodal benchmark functions for which the performance of the same algorithm using static HMP was analysed in Chapter 4.

The $(1+1)$ IA using inversely proportional HMP simply uses one candidate solution in each iteration to which hypermutation is applied (Algorithm 2.4 using Function 2.6 with the related M values). The results proven in this section and comparisons with static HMP are summarised in Table 6.1. The original inversely proportional HMP of Opt-IA [20] has the same expected runtimes as the $(1+1)$ IA using static HMP [52].

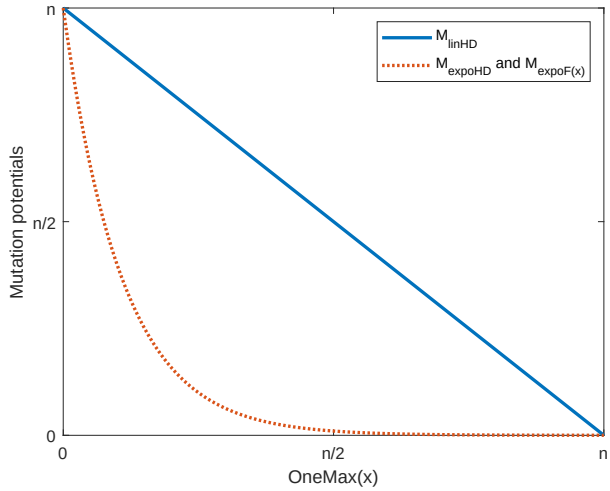

Fig. 6.1 Inversely proportional mutation potentials for ONEMAX

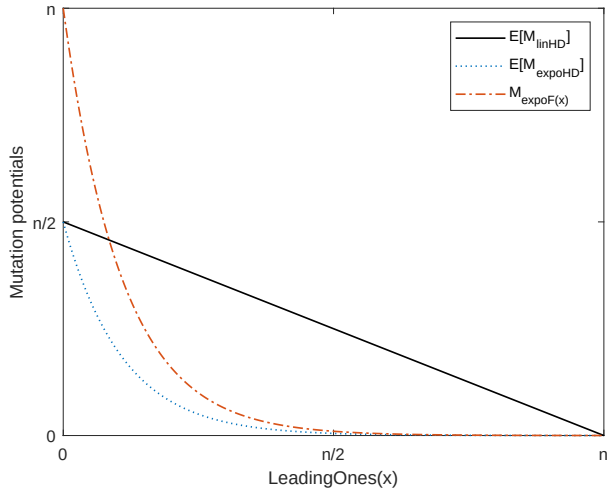

Fig. 6.2 Inversely proportional mutation potentials for LEADINGONES

Figures 6.1 and 6.2 show how the studied MPs decrease during the run of the algorithm when optimising ONEMAX and LEADINGONES, respectively. In Figure 6.2, expected values are shown for those using Hamming distance.

6.4.1 Behaviour for ONEMAX in Ideal Conditions

The following theorems derive the expected runtimes of the three variants of inversely proportional MPs for the function ONEMAX. By decreasing the mutation potential linearly with the decrease of the Hamming distance, a logarithmic factor may be shaved off from the

expected runtime of the $(1+1)$ IA compared to the expected runtime achieved by the same algorithm using static MP.

Theorem 36. *The $(1+1)$ IA using M_{linHD} optimises* ONEMAX *in $\Theta(n^2)$ expected fitness function evaluations.*

Proof. Considering i as the number of 0-bits in the candidate solution, the probability of improvement in the first step is i/n. Knowing that at most n improvements are needed to find the optimum and in case of failure, $H(x,\text{opt}) = i$ fitness function evaluations would be wasted, the total expected time to optimise ONEMAX is at most $\sum_{i=1}^{n} \frac{n}{i} \cdot i = O(n^2)$.

In order to prove a lower bound, we use the Ballot theorem (see Theorem 11, page 38). By Chernoff bounds (see Theorem 8, page 36), the number of 0-bits in the initialised solution is at least $n/3$ w.o.p. Considering the number of 0-bits as $i = q$ and the number of 1-bits as $n - i = p$, the probability of an improvement is at most $1 - (p-q)/(p+q) = 1 - (n-2i)/n = 2i/n$ by the Ballot theorem, where $i = H(x,opt)$. This means that we need to wait at least $n/(2i)$ iterations to see an improvement and each time the mutation operator fails to improve the fitness, i fitness function evaluations will be wasted. Considering that at least $n/3$ improvements are needed, the expected time to optimise ONEMAX is larger than $\sum_{i=1}^{n/3} \frac{n}{2i} \cdot i = \Omega(n^2)$. □

The following theorem shows that a greater speed-up may be achieved if the potential decreases exponentially rather than linearly. Its proof deviates from the proof of Theorem 36 only in the amount of wasted evaluations. Note that for ONEMAX, the Hamming distance of a solution to the optimum and its difference in fitness, are the same.

Theorem 37. *The $(1+1)$ IA using either $M_{expoF(x)}$ or M_{expoHD} optimises* ONEMAX *in $O(n^{(3/2)+\varepsilon}\log n)$ and $\Omega(n^{(3/2)-\varepsilon})$ expected fitness function evaluations for any arbitrarily small constant $\varepsilon > 0$.*

Proof. We prove the results for $\text{M}_{\text{expoF(x)}}$ which applies to M_{expoHD} as well. To prove the upper bound, we know that by Chernoff bounds the initialised solution has at most $n/2+\varepsilon n$ 0-bits w.o.p. for any arbitrary small $\varepsilon = \Theta(1)$. The probability of improvement in the first mutation step is at least i/n with i showing the number of 0-bits. As we need at most $n/2+\varepsilon n$ improvements and each time the mutation fails to make an improvement at least $n^{1-\frac{n-i}{n}}$ fitness function evaluation would be wasted, the total expected time to find the optimum will be $E(T) \leq \sum_{i=1}^{n/2+\varepsilon n} \frac{n}{i} \cdot n^{\frac{i}{n}} =\leq n^{(3/2)+\varepsilon}\log n$ by putting $i = n/2+\varepsilon n$ in $n^{i/n}$.

To prove the lower bound, we again consider i as the number of 0-bits. By Chernoff bounds, i is at least $n/2-\varepsilon n$ in the initialised bit string. The probability of improvement is at most $2i/n$ by the Ballot theorem in Chapter 3), and the number of wasted fitness

function evaluations at each failure is $n^{\frac{i}{n}}$. If we consider the time spent between levels $n(1/2+\varepsilon/2)$ and $n(1/2+\varepsilon)$, we get the expected time of $n^{\frac{n/2-\varepsilon n}{n}} \cdot \sum_{i=n/2+n\varepsilon/2}^{n/2+\varepsilon n} \Omega(1) = n^{1/2-\varepsilon} \cdot \varepsilon n/2 \cdot \Omega(1) = \Omega(n^{3/2-\varepsilon})$. □

6.4.2 Behaviour for LEADINGONES in Ideal Conditions

In the previous subsection, it was shown that decreasing the mutation potential linearly with the Hamming distance to the optimum gives a logarithmic factor speed-up for ONEMAX compared to using static MP. The following theorem shows that no improvement over static mutation potentials are achieved for LEADINGONES. The main reason for the lack of asymptotic improvements is that the expected time to leave each fitness level is linear for LEADINGONES.

Theorem 38. *The* $(1+1)$ *IA using* M_{linHD} *optimises* LEADINGONES *in* $\Theta(n^3)$ *expected fitness function evaluations.*

Proof. The proof for the lower bound is the same as the proof of Theorem 15 (page 59) for the expected runtime of the $(1+1)$ $\text{IA}_{\geq}^{hyp}$ with the exception that the amount of wasted fitness function evaluations in case of failure is $H(x,\text{opt})$ instead of n now. Considering i as the number of leading 1-bits, we denote the expected number of fitness function evaluations until an improvement happens by $E(f_i)$. Any such candidate solution has i leading 1-bits with a 0-bit following, and then $n-i-1$ other bits which are distributed uniformly at random [31]. We take into account three possible events E_1, E_2 and E_3 that can happen in the first bit flip: E_1 is the event of flipping a leading 1-bit which happens with probability i/n, E_2 is the event of flipping the first 0-bit which happens with probability $1/n$, and E_3 is the event of flipping any other bit which happens with probability $(n-i-1)/n$. So we get $E(f_i \mid E_1) = H(x,\text{opt}) + E(f_i)$, $E(f_i \mid E_2) = 1$, and $E(f_i \mid E_3) = 1 + E(f_i)$. Knowing that the bits after the left most 0-bit are distributed uniformly at random [31], the value of $H(x,\text{opt})$ is not less than $1 + \frac{(n-i-1)}{2} - \varepsilon n$ w.o.p. by Chernoff bounds (see Theorem 8, page 36). Hence, by the law of total expectation, the expected number of fitness function evaluations for every improvement is $E(f_i) = \frac{i}{n}\left(1 + \frac{n-i-1}{2} - \varepsilon n + E(f_i)\right) + \frac{1}{n} \cdot 1 + \frac{n-i-1}{n}(1+E(f_i))$. Solving the equation for $E(f_i)$ gives us $E(f_i) = \frac{in-i^2-i-i\varepsilon n}{2}$.

As we are computing the lower bound, we have to take into account the probability of skipping any level i. To do this, we need to calculate the expected number of consecutive 1-bits that follow the leftmost 0-bit, called free riders [31]. The expected number of free riders is $\sum_{i=1}^{n-i-1} i \cdot 1/2^{i+1} < \frac{1}{2} \cdot \sum_{i=0}^{\infty} i \cdot (1/2)^i = \frac{1}{2} \cdot \frac{1/2}{(1-1/2)^2} = 1$. This means that the probability of not skipping a level i is $\Omega(1)$. On the other hand, with probability at least $1-2^{-n/2}$, the initial solution does not have more than $n/2$ leading 1-bits. Thus, we obtain a lower bound

of $(1-2^{-n/2})\sum_{i=n/2}^{n} E(f_i) = \Omega(1)\sum_{i=n/2}^{n} \frac{in-i^2-i-i\varepsilon n}{2} = \Omega(n^3)$ on the expected number of fitness function evaluations.

Now we prove the upper bound. The probability of improvement in each step is at least $1/n$ which is the probability of flipping the leftmost 0-bit. Since at most n improvements are needed and each failure in improvement yields $\frac{n-i-1}{2}+1+\varepsilon n$ wasted fitness function evaluations in expectation (i.e., $E[H(x,\text{opt})]$) where i is the number of the leading 1-bits, the expected time to find the optimum is at most $\sum_{i=1}^{n} n \cdot \left(\frac{n-i-1}{2}+1+\varepsilon n\right) = O(n^3)$.

□

Theorem 39 shows that exponential fitness-based MP gives us at least a logarithmic and at most a $\sqrt{n}$ factor speed-up compared to static MP. Before proving the main result, we introduce the following lemma that will be used in the proof of the upcoming theorems.

Lemma 12. *For large enough n and any arbitrarily small constant* ε, $n^{1/n^\varepsilon} = (1+\frac{\ln n}{n^\varepsilon})(1 \pm o(1))$.

Proof. By raising $\left(1+\frac{\ln n}{n^\varepsilon}\right)$ to the power of $\frac{n^\varepsilon}{\ln n}\cdot\frac{\ln n}{n^\varepsilon}$, we get $\left(1+\frac{\ln n}{n^\varepsilon}\right)^{\frac{n^\varepsilon}{\ln n}\cdot\frac{\ln n}{n^\varepsilon}} = (1\pm o(1))\, e^{\frac{\ln n}{n^\varepsilon}} = (1\pm o(1)) n^{1/n^\varepsilon}$. □

Theorem 39. *The* $(1+1)$ *IA using* $M_{expoF(x)}$ *optimises* LEADINGONES *in* $O(n^3/\log n)$ *and* $\Omega(n^{(5/2)+\varepsilon})$ *expected fitness function evaluations for any arbitrarily small constant* $\varepsilon > 0$.

Proof. We first prove the upper bound. As already shown in the proof of Theorem 38, the expected number of leading 1-bits is less than 1 in the initialised bit string. The probability of improvement in the first step is $1/n$. In case of failure in improving at the first step, at most $n^{\frac{n-i}{n}}$ fitness function evaluations would get wasted where i is the number of leading 1-bits. Therefore, the total expected time to find the optimum is $E(T) \leq \sum_{i=1}^{n-1} n \cdot n^{(n-i)/n} = O(n^3/\log n)$ considering that $\sum_{i=1}^{n-1} n^{i/n} = \sum_{i=2}^{n} (n^{1/n})^{i-1} < \sum_{i=1}^{n} (n^{1/n})^{i-1} = \frac{1-n}{1-n^{1/n}}$ (using the general partial power series sum: $\sum_{i=0}^{m} a^i = \frac{1-a^{m+1}}{1-a}$ for $a \neq 1$) that gets in turn bounded from above by $n^2/(\log n)$ using Lemma 12.

The proof for the lower bound is similar to the proof of Theorem 38, except in the calculation of $E(f_i)$ when we want to consider the amount of wasted fitness function evaluations in case of E_1 happening. Here we have $E(f_i) = \frac{i}{n}\left(n^{(n-i)/n} + E(f_i)\right) + \frac{1}{n}\cdot 1 + \frac{n-i-1}{n}(1+E(f_i))$. Solving it for $E(f_i)$ gives us $E(f_i) = i\cdot n^{(n-i)/n} + n - i$. Hence, the expected time to optimise LEADINGONES is $(1-2^{-n/2})\sum_{i=n/2}^{n} E(f_i) = \Omega(1)\sum_{i=n/2}^{n}\left(i\cdot n^{(n-i)/n} + n - i\right) = \Omega(1)\left(\sum_{i=n/2}^{n}(n-i) + \sum_{i=n/2}^{n}\left(i\cdot n^{(n-i)/n}\right)\right)$. Evaluating the second sum in the interval $i \in [n/2+\frac{\varepsilon n}{2}, n/2+\varepsilon n]$, we get $\varepsilon n/2 \cdot (n/2+\varepsilon n)\cdot n^{1/2-\varepsilon/2} = \Omega(n^{(5/2)-\varepsilon})$. □

An advantage of Hamming distance-based exponential decays for MPs compared to fitness-based ones is shown by the following theorem for LEADINGONES. The reason can be seen in Figure 6.2. While the initial fitness is very low, the potential is very high, but the actual number of bits that have to be flipped to reach the optimum (i.e., the Hamming distance), is much smaller. More precisely, fitness-based potentials suggest to flip n bits when only $n/2$ bits have to be flipped in expectation to reach the optimum in one step. M_{expoHD} exploits this property, thus wastes fewer fitness evaluations than $\text{M}_{\text{expoF(x)}}$.

Theorem 40. *The* $(1+1)$ *IA using* M_{expoHD} *optimises* LEADINGONES *in* $O(\frac{n^{(5/2)+\varepsilon}}{\ln n})$ *and* $\Omega(n^{(9/4)-\varepsilon})$ *expected fitness function evaluations for any arbitrarily small constant* $\varepsilon > 0$.

Proof. The proof for the upper bound is similar to the proof of Theorem 38, however, each failure in improvement yields $n^{H(x,\text{opt})/n}$ wasted fitness function evaluations. Since the bits after the leading ones are uniformly distributed, by Chernoff bounds the number of 0-bits (Hamming distance to the optimum) is less than $1+((n-i-1)/2)+\varepsilon n$ w.o.p. Hence, the expected time to optimise LEADINGONES is $\sum_{i=1}^{n} n \cdot n^{\frac{1+((n-i-1)/2)+\varepsilon n}{n}} = n\sum_{i=1}^{n} n^{\frac{1}{2}-\frac{i}{2n}+\varepsilon+\frac{1}{2n}} = n \cdot n^{1/2+\varepsilon+\frac{1}{2n}} \sum_{i=1}^{n} n^{-\frac{i}{2n}}$. Knowing that $\sum_{i=0}^{\infty} n^{-\frac{i}{2n}} = \frac{1}{1-n^{(-1/(2n))}}$ and $n^{-1/(2n)} \leq \left(1-\frac{\ln n}{2n}\right)(1-o(1))$ for all $n>1$ by Lemma 12, we get $1-n^{(-1/(2n))} < \frac{\ln n}{2n}(1+o(1))$. Hence, the expected time is $E(T) < n^{(3/2)+\varepsilon} \cdot n^{1/(2n)} \cdot O\left(\frac{n}{\ln n}\right) = O\left(\frac{n^{(5/2)+\varepsilon}}{\ln n}\right)$.

The proof for lower bound is also similar to the proof of Theorem 38. By taking the same steps and solving the equation for $E(f_i)$,we get $E(f_i) = in^{1/n+\frac{n-i-1}{2n}-\varepsilon} + n - i$. Then, by taking into account the probabilities of starting with less than $n/2$ leading 1-bits and skipping a level i, we get the expected runtime of

$$(1-2^{-n/2}) \sum_{i=n/2}^{n} E(f_i) \geq \Omega(1) \sum_{i=n/2+\varepsilon n/2}^{n/2+\varepsilon n} \left(in^{1/n+((n-i-1)/2n)-\varepsilon} + n - i)\right)$$
$$\geq \Omega(1) \sum_{i=n/2+\varepsilon n/2}^{n/2+\varepsilon n} in^{1/2-\varepsilon+(1/(2n))-(i/(2n))} = \Omega(n^{(9/4)-\varepsilon}).$$

□

Given that M_{expoHD} provides larger speed-ups on the unimodal benchmark functions compared to the other analysed inversely proportional MPs and is stable to the scaling of fitness functions, we will use it in the remainder of the chapter.

6.5 Inefficient Behaviour In Practice

In this section, we will focus on problems that M_{expoHD} may encounter in realistic applications where the optimum is unknown. To this end, the best found solution will be used by the operator rather than the unknown optimum. We combine M_{expoHD} with (hybrid) ageing, as in the Opt-IA. The pseudo-code is provided in Algorithm 6.1. It is essentially the same as Algorithm 4.1 (page 56) with the inversely proportional MP, $\mu = 1$, dup $= 1$, and $p_{\text{die}} = 1/2$.

The algorithm can take advantage of the power of ageing (shown rigorously in Chapter 4 and 5) at automatically restarting the search process once it is stuck on a local optimum. Our aim in the design of the algorithm is that, the greater the number of local optima that are identified by the algorithm, the better M_{expoHD} should approximate its ideal behaviour. However, we will show that this is not the case all the time, e.g., it is not true for the well-studied bimodal benchmark function TwoMax (defined by (3.9), page 43). TwoMax is usually used to evaluate the global exploration capabilities of EAs, i.e., whether the population identifies both optima of the function. Our analysis shows that once the $(1+1)$ Opt-IA with M_{expoHD} escapes from one local optimum, the mutation rate will increase as the algorithm climbs up the other branch. As a result, the algorithm struggles to efficiently identify the other optimum and wastes more and more fitness function evaluations as it approaches it. Thus, the whole purpose behind inversely proportional HMP is defeated.

On the bright side, we will show that M_{expoHD} combined with ageing can escape from local optima. We will use the well known Cliff_d function (defined by (3.7), page 41) for this purpose where we have proven that static HMP combined with ageing fails (see Theorem 19, page 68).

We first start with the analysis of TwoMax. The following theorem shows that after the algorithm escapes from the local optimum, the mutation rate increases as the algorithm climbs up the opposite branch. This behaviour causes a large waste of fitness evaluations defying the objectives of inversely proportional HMP.

Theorem 41. *The expected runtime of the* $(1+1)$ *Opt-IA with* M_{expoHD} *for* TwoMax *is* $O(n^2 \log n)$ *and* $\Omega(n^{2-\varepsilon})$ *with* $\tau = \Omega(n^{1+\varepsilon})$ *for some small constant* $\varepsilon > 0$.

Proof. We first prove the upper bound. Let x_t be the current solution at the beginning of the iteration t. Immediately after the initialisation, the best seen solution is the current individual x_1 itself and the mutation potential is $n^{H(x_1,x_1)/n} = n^0 = 1$. Note that $x_t \neq x_{t+1}$ if and only if either $f(x_{t+1}) \geq f(x_t)$ or x_{t+1} is reinitialised after x_t is removed from the population due to ageing. Thus, the mutation operator flips a single bit at every iteration until ageing is triggered for the first time. The improvement probability will be at least $1/n$ until either 1^n or 0^n is sampled. Given that the ageing threshold τ is at least $n^{1+\varepsilon}$ for some constant $\varepsilon > 0$,

Algorithm 6.1: $(1+1)$ Opt-IA with $\mathrm{M}_{\mathrm{expoHD}}$

begin
  $t := 0$;
  initialise $x \in \{0,1\}^n$ uniformly at random;
  set $x.age := 0$ and best $:= x$;
  while *a global optimum is not found* **do**
    $x.age := x.age + 1$;
    $M := \mathrm{M}_{\mathrm{expoHD}}$;
    call Function 2.6;
    if $f(y) > f(x)$ **then**
      $y.age := 0$;
    else $y.age := x.age$;
    if $f(y) \geq f(best)$ **then**
      set best $:= y$;
    for $w \in \{x, y\}$ **do**
      if $w.age \geq \tau$ **then**
        with probability $p_{\mathrm{die}} = 1/2$, reinitialise w uniformly at random with $w.age = 0$;
    Set $x = \arg\max_{z \in \{x,y\}} f(z)$;
  $t := t + 1$;

the probability that the current solution will not improve τ times consecutively is at most $(1-\frac{1}{n})^{n^{1+\varepsilon}} = e^{-\Omega(n^{\varepsilon})}$. Hence, the first optimum will be found before ageing is triggered w.o.p. Given that the ageing operator is not triggered, the expected time to find the first optimum is at most $O(n \log n)$ with a proof similar to that of the standard RLS [31]. After finding the first optimum, no single bit-flip can yield an equally fit solution. The individual then reaches age τ and the ageing reinitialises the current solution. Thus, with $(1-2^{-n}) \cdot \left(1 - e^{-\Omega(n^{\varepsilon})}\right)$ probability a new solution will be initialised in $\tau + n$ steps. Now, the current *best* is the first discovered optimum.

Let the first and second branch denote the subsets of the solution space which consist of solutions with less than and more than Hamming distance $n/2$ to the first discovered optimum respectively. We first consider the case when the new solution is initialised on the first branch. Given that the current solution has fitness i, the distance to the best seen is $n-i$ and the mutation potential is $n^{\frac{n-i}{n}}$. Since the first constructive mutation ensures that the probability of improvement is always at least $1/n$, w.o.p. the ageing will not be triggered until either optimum is discovered. Thus, given that the current solution is always on the first branch, the proof of Theorem 37 carries over and the expected time to find the first discovered optimum

once again is at most $O(n^{3/2}\log n)$ in expectation. When the first discovered optimum is sampled again, w.o.p. the current solution is reinitialised with the first discovered optimum as the best seen solution in at most $\tau + n$ iterations.

Now, we consider the case where the current solution is on the second branch. A lower bound on the probability that the current solution will reach the optimum of the second branch before sampling an improving solution in the first branch will conclude the proof since the expected time to do so is $O(n^{3/2}\log n)$ given that the current solution does not switch branches before.

We will start by bounding the mutation potential for a solution in the second branch with fitness value $n-k$. Since the $\mathrm{M}_{\mathrm{expoHD}}$ is always smaller than $\mathrm{M}_{\mathrm{linHD}}$, we can assume that no more than $n-k$ bits will be flipped.

W.l.o.g, let the second branch be the branch with more 0-bits. For a hypermutation operation with mutation potential N, the Nth solution that will be sampled has the highest probability of finding a solution with at least $n-k$ 1-bits (i.e., switching to the first branch). The current solution has $n-k$ 0-bits and k 1-bits. If more than $k/2$ 1-bits are flipped, then the number of 1-bits in the final solution after $n-k$ mutation steps is less than $n-k$ since the number of 0-bits flipped to 1-bits is less than $n-k-k/2$ and the number of remaining 1-bits is less than $k/2$. The event that at most $k/2$ 1-bits are flipped is equivalent to the event that in a uniformly random permutation of n bit positions at least $k/2$ 1-bits are ranked in the last k positions of the random permutation. We will now bound the probability that exactly $k/2$ 1-bits are in the last k positions since having more 1-bits has a smaller probability. Each particular outcome of the last k positions has the equal probability of $\prod_{i=0}^{k-1}(n-i)^{-1}$. There are $\binom{k}{k/2}$ different equally likely ways to choose $k/2$ 1-bits and $k!$ different permutations of the last k positions, thus the probability of having exactly $k/2$ 1-bits in the last k positions is $\prod_{i=0}^{k-1}\frac{1}{n-i}\cdot\binom{k}{k/2}\cdot k! = \prod_{i=0}^{k-1}\frac{1}{n-i}\cdot\left(\frac{k!}{(k/2)!}\right)^2 < \left(\frac{k}{n-k}\right)^k$. For any $k \in [4, \frac{n}{2}-\Omega(\sqrt{n})]$, this probability is in the order of $\Omega(1/n^4)$. Using a union bound (see Theorem 4, page 35) over the probabilities of having more than $k/2$ 1-bits and the probabilities of improving before the final step, we get $\frac{k}{2}\cdot(n-k)\cdot\Omega(1/n^4) = \Omega(1/n^2)$. Since the new solutions are uniformly sampled, the number of bits in the solutions are initially distributed binomially with parameters n and $1/2$ which has a variance in the order of $\Theta(\sqrt{n})$ and implies that with constant probability k is less than $\frac{n}{2}-\Omega(\sqrt{n})$ in the initial solution. Given that the expected time in terms of generations is in the order of $O(n\log n)$, the total probability of switching branches while $k \in [k, \frac{n}{2}-\Omega(\sqrt{n})]$ is at most $\left(1-1/n^2\right)^{O(n\log n)} = 1-\Omega(1)$. Finally, we will consider the cases of $k \in \{1,2,3\}$ separately. When $k=1$, the probability of flipping less than $k/2$ 1-bits is equivalent to the probability of not flipping the single 1-bit which happens with probability $1/n$. For $k=2$, similarly we have to flip at most one 1-bit

with probability $O(1/n)$. For $k = 3$, it is necessary that at least two 1-bits are not flipped which happens with probability at most $O(1/n^2)$. Thus, the probability of switching branches when $k < 4$ is at most $O(1/n)$, which gives a lower bound on the total probability of switching branches in the order of $1 - \Omega(1)$ given that the initial solution has at least $k > \frac{n}{2} - \Omega(\sqrt{n})$ 1-bits (which also occurs with at least $\Omega(1)$ probability). Given that no switch occurs, the expected time to find the second optimum is $\sum_{i=1}^{n/2+\varepsilon n} \frac{n}{i} \cdot n^{\frac{n-i}{n}} = O(n^2 \log n)$.

For the lower bound, we simply show that after finding one of the optima (the goal is to find both optima), the algorithm will need in expectation $\Omega(n^{2-\varepsilon})$ generations to traverse a constant interval of length εn before reaching the second optimum. Without losing generality, we assume the branch leading to 0^n is discovered first. After dying due to ageing, we assume the algorithm re-initialises on the second branch. The probability of improvement is at most $2i/n$ by the Ballot theorem, and every time the algorithm fails to improve, at least $n^{\frac{n-\varepsilon n}{n}}$ fitness evaluations would be wasted. Hence, the expected time would be at least $\sum_{i=1}^{\varepsilon n}(\frac{1}{2\varepsilon}) \cdot 1 + (\frac{1}{2\varepsilon} - 1) \cdot n^{1-\varepsilon} = \Omega(n^{2-\varepsilon})$. □

Now we show that differently from static MP, M_{expoHD} combined with ageing can escape from the local optima of $\textsc{Cliff}_d$ (defined by (3.7), page 41), hence optimises the function efficiently.

Theorem 42. *The* $(1+1)$ *Opt-IA with* M_{expoHD} *and* $\tau = \Omega(n^{1+\varepsilon})$ *for an arbitrarily small constant* ε*, optimises* $\textsc{Cliff}_d$ *with* $d < n(\frac{1}{4} - \varepsilon)$ *and* $d = \Theta(n)$ *in* $O(n^{3/2} \log n + \tau n^{1/2} + \frac{n^{7/2}}{d^2})$ *expected fitness function evaluations.*

Proof. The analysis will follow a similar idea to the proof of Theorem 41. After initialisation, the initial mutation potential is $M = 1$ since the current solution is the best seen solution. With single bit-flips, it takes in expectation $O(n)$ to find a local optimum (i.e., a search point with $n - d$ 1-bits) as the improvement probability is always at least $(n-d)/n = \Omega(1)$. Since the local optima cannot be improved with single bit-flips, in τ generations after it was first discovered, ageing will be triggered and in the following $\Theta(n)$ steps the current solution will be removed from the population with probability $1 - 2^{-\Omega(n)}$. The Hamming distance of the reinitialised solution is distributed binomially with parameters n and $1/2$ and w.o.p. is smaller than $n/2 + n^{2/3}$, yielding an initial mutation potential of $M = O(n^{1/2})$. We pessimistically assume that the mutation potential will not decrease until a local optima is found again, which implies that the expected time will be at most $O(n^{3/2} \log n + \tau n^{1/2})$ since each iteration will waste an extra $O(n^{1/2})$ fitness function evaluations. After finding a local optimum again, the mutation potential will be $M = 1$ since it replaces the previously observed local optimum as the best seen solution. The process of reinitialisation and reaching the local optima will repeat itself until the following event happens.

If the local optima produces an offspring with $n-d+1$ bits with probability d/n and if this solution survives the ageing operator (with probability $(1-p_{\text{die}})$), then the reinitialised solution will be rejected since its fitness value is less than $n-d$ w.o.p. The Hamming distance of this new solution to the best seen will be exactly one since it is created via a single bit-flip, thus its mutation potential will be $M=1$. Moreover, if the surviving offspring improves again (with probability $(d-1)/n$) in the next iteration, it will reset its age to zero and will have Hamming distance of at least two to any local optimum. In the expected $O(n\log n)$ generations, this solution will reach the global optimum unless a solution with less or equal $n-d$ 1-bits is sampled before. Initially, this is impossible since for at least $\omega(1)$ steps we have $M=1$, and later $M<3$ holds as long as the distance to the last seen local optimum is at most $n/\ln n$ since $n^{\frac{n/\ln n}{n}}=e$. Note that the number of 1-bits does not always reflect the actual Hamming distance since more than one bit can be flipped in an accepted offspring. We will pessimistically assume that all improvements have increased the Hamming distance by three until the total Hamming distance reaches $n/\ln n$, which implies that there have been $n/(3\ln n)$ accepted solutions. The Ballot theorem (see Theorem 11, page 38) implies that sampling a solution that is at least as good as the parent (which are the only solutions that are accepted) has probability at most $2i/n$ where i is the number of 0-bits in the solution. Since the probability of improving in the first step is at least i/n, we can conclude that the conditional probability that an accepted offspring is a strict improvement is at least $1/2$. Thus, when the Hamming distance to the local optima reaches $n/\ln n$, in expectation, the current solution will have at least $n/(6\ln n)$ extra 1-bits compared to the local optima and at least $n^{3/5}$ extra 1-bits w.o.p. by Chernoff bounds. The Hamming distance to the local optima can be at most $2d$ since both the local optima and the current solution have less than d 0-bits. Since $d<n/4$, the mutation potential is at most $\sqrt{n}$. Thus, no hypermutation can yield a solution with less than $n-d+2$ 1-bits. Therefore, once a solution with $n-d+2$ bits is added to the population, the algorithm finds the optimum w.o.p. in $O(n\log n)$ iterations and in at most $O(n^{3/2}\log n)$ fitness function evaluations since the mutation potential is at most $\sqrt{n}$. The probability of obtaining a solution with $n-d+2$ 1-bits at the end of each cycle of reinitialisation and removal of the local optima due to ageing is $(1-p_{\text{die}})^2\cdot(d/n)\cdot((d-1)/n)$. Since each such cycle takes $O(n^{3/2})$ fitness evaluations, our claim follows. □

Hence, we conclude that if the algorithm identifies some slope that leads away from the previous local optimum, then the mutation rate will increase as the new optimum is approached. Firstly, this makes the new optimum hard to identify. Secondly, the high mutation rates can lead to high wastage of fitness function evaluations defeating our main motivation of reducing such wastage compared to static MP.

6.6 An Efficient Inversely Proportional Hypermutation Operator with Mutation Potentials for Opt-IA in Practice

In the previous section, we observed in the analyses of both CLIFF and TWOMAX that towards the end of the optimisation process, the mutation potential may increase as the current solution approaches an undiscovered, potentially promising optimum. This behaviour is against the design intentions of the inversely proportional MPs since in the final part of the optimisation process it gets harder to find improvements and high mutation potentials lead to many wasted fitness function evaluations. The underlying reason for this behaviour in both the CLIFF and TWOMAX landscapes is the necessity to follow a gradient that leads away from the local optimum to find the global one. Considering that this necessity would be ubiquitous in optimisation problems, we propose a new method to control mutation potentials in this section. The newly proposed mutation potential is called *Symmetric* M_{expoHD} and is defined as:

$$\text{Symmetric } \mathrm{M}_{\text{expoHD}} := \max\left\{\left\lfloor n^{\frac{H(\text{best},x.\text{org})-H(x,x.\text{org})}{n}}\right\rfloor, 1\right\}.$$

Symmetric $\mathrm{M}_{\text{expoHD}}$ is inversely proportional to the current solution's Hamming distance to its *origin*, where the origin is defined as the ancestor of the current bit string after the last removal of a solution due to ageing. At the initialisation, the origin of each solution is set to itself and newly created offspring inherit the origin of their parents. An Opt-IA applying this mutation potential is shown in Algorithm 6.2.

This mutation potential reliably decreases at the same rate it would decrease if it was approaching the currently best seen local optimum as the current solution improves and moves away from its origin up until the Hamming distance to the best found solution is covered. From then on, the algorithm will perform a local search since the mutation potential will be minimal (i.e., one bit will be flipped). Every time a local optimum is found, ageing is triggered after approximately τ steps and then both surviving and reinitialised individuals reset their origin to their own bit string. Figure 6.3 illustrates how the mutation potential is determined according to the ratio of the Hamming distance between the current solution and its origin ($H(x,x.org)$) and the Hamming distance between its origin and the best seen solution ($H(best,x.org)$).

The following theorem shows that once one optimum of TWOMAX has been identified, the value of the Symmetric $\mathrm{M}_{\text{expoHD}}$ potential decreases as both optima are approached as desired and the wished for speed-up (calculated in Section 6.4 for hill-climbing ONEMAX in ideal conditions) in the expected runtime is achieved.

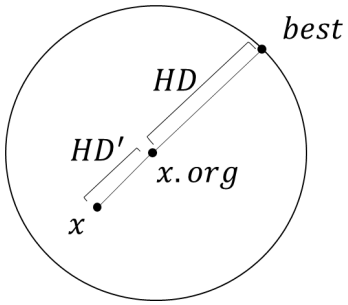

Fig. 6.3 The geometric representation of Symmetric M_{expoHD}

Algorithm 6.2: $(1+1)$ Opt-IA with Symmetric M_{expoHD}

```
1  begin
2      t := 0;
3      initialise x ∈ {0,1}^n uniformly at random;
4      set x.origin := x, x.age := 0, and best := x;
5      while a global optimum is not found do
6          x.age := x.age + 1;
7          M := Symmetric M_expoHD;
8          call Function 2.6;
9          y.origin := x.origin;
10         if f(y) > f(x) then
11             y.age := 0;
12         else y.age := x.age;
13         if f(y) ≥ best then
14             best := y;
15         for w ∈ {x,y} do
16             if w.age ≥ τ then
17                 with probability p_die = 1/2, reinitialise w uniformly at random with
                    w.age = 0;
18                 set w.origin = w;
19         Set x = arg max_{z∈{x,y}} f(z);
20     t := t + 1;
```

Theorem 43. *The* $(1+1)$ *Opt-IA using Symmetric* M_{expoHD} *with* $\tau = \Omega(n^{1+\varepsilon})$ *for any arbitrarily small constant* $\varepsilon > 0$*, needs* $O(n^{(3/2)+\varepsilon}\log n)$ *fitness function evaluations in expectation to optimise* TwoMax.

Proof. The expected time until the first branch is optimised is $O(n\log n)$ since the best seen search point is the current best individual and consequently the mutation potential is $M = 1$.

Since the improvement probability is at least $1/n$ and $\tau = \Omega(n^{1+\varepsilon})$, the ageing operator does not get triggered before finding one of the optima w.o.p. Once one of the optima is found, ageing reinitialises the individual while the first discovered optimum stays as the current best seen search point. For the randomly reinitialised solution, the Hamming distance to the best seen is binomially distributed with parameters n and $1/2$. Using Chernoff bounds, we can bound the distance to the previously seen optima by at most $n/2+n^{2/3}$ w.o.p. This Hamming distance implies an initial mutation potential of $M < n^{\frac{\frac{n}{2}+n^{2/3}}{n}} = n^{\frac{1}{2}+\frac{1}{n^{1/3}}}$ which decreases as the individual increases its distance to the origin and can never go above its initial value where the distance is zero. Pessimistically assuming that the mutation potential will be $n^{\frac{1}{2}+\frac{1}{n^{1/3}}} = O(n^{1/2})$ throughout the run, we can obtain the above upper bound by summing over all levels and using a coupon collector's argument (see Theorem 5, page 35). □

The following theorem shows that Symmetric $\mathrm{M}_{\mathrm{expoHD}}$ is also efficient for $\textsc{Cliff}_d$.

Theorem 44. *The* $(1+1)$ *Opt-IA using Symmetric* M_{expoHD} *with* $\tau = \Omega(n^{1+\varepsilon})$ *optimises* $\textsc{Cliff}_d$ *with* $d < n(\frac{1}{4}-\varepsilon)$ *and* $d = \Theta(n)$ *in* $O(n^{(3/2)+\varepsilon}\log n + \tau n^{1/2} + \frac{n^{7/2}}{d^2})$ *expected fitness function evaluations.*

Proof. The proof is almost identical to the proof of Theorem 42. The most important distinction is that once a solution with $n-d+2$ 1-bits is created, its mutation potential remains as $M = 1$ until the global optimum is found. The reason is that when ageing is triggered, the surviving solutions all reset their origins to themselves, i.e., they start doing randomised local search. □

6.7 Conclusion

An analysis of inversely proportional HMP was presented in this chapter. Previous theoretical studies have shown disappointing results concerning the inversely proportional hypermutations from the literature. Here, we proposed a new inversely proportional HMP operator based on Hamming distance and exponential decay. The crucial insight is that FCM is essential with high mutation rates and that the mutation potential should be minimal in proximity of the optimum. We have shown its effectiveness in isolation for unimodal functions compared to static HMP in the ideal situation when the optimum is known. Furthermore, we have provided a symmetric version of the operator for the complete Opt-IA AIS to be used in practical applications where the optimum is usually unknown. We have proved its efficiency for two well-studied multimodal functions where the achieved expected runtimes match the ideal expected runtimes.

While we have shown that speed-ups in the exploitation phase may be achieved via inversely proportional HMP, other inconveniences may arise from their use. For instance, the capabilities of static hypermutations of using high mutation rates for escaping from local optima may be lost because inversely proportional HMP are likely to have very low mutation rates in proximity of local optima. For instance, it is unlikely that inversely proportional HMP can guarantee the arbitrarily good approximation ratios of static HMP for PARTITION as seen in Chapter 5. However, such problems may be overcome by Opt-IA with inversely proportional HMP due to the capabilities of the ageing operator at escaping local optima.

In the next chapter, we present an alternative strategy to speed up static HMP and we show that it leads to greater speed-ups compared to inversely proportional HMPs. We will prove that this strategy does not suffer from the limitations pointed above.

Chapter 7

Hypermutations with Probabilistic Intermediate Evaluations

7.1 Introduction

Up until now, we have shown that static HMPs are by a linear factor slower than standard EAs in the exploitation phase although they are considerably more efficient at escaping local optima. The same problem was identified for the inversely proportional HMPs from the literature as well. In Chapter 6, we proposed some modifications to the inversely proportional HMP to make it reduce the mutation potential properly as it gets closer to the optimum and we proved asymptotic speed-ups compared to static HMP for exploitation. However, a reduction of the mutation potential as local optima are approached leads to new problems, e.g., the hypermutations can only escape local optima if combined with ageing. This conflicts with the main advantage of hypermutations which is to have a good exploration capability.

In this chapter, we propose some alternatives for the FCM mechanism to be used by static HMP which allow it to be as efficient as randomised local searches in the exploitation phases while maintaining its characteristic capability of escaping from local optima. Rather than evaluating the fitness after each bit-flip of a hypermutation as the traditional FCM requires, we propose to evaluate it based on the probability that the mutation will be successful. The probability of hitting a specific point at Hamming distance i from the current point, $\binom{n}{i}^{-1}$, decreases exponentially with the Hamming distance for $i < n/2$ and then it increases again in the same fashion. Based on this observation, we evaluate each bit with some probability that follows a *parabolic* distribution such that the probability of evaluating the ith bit-flip decreases as i approaches $n/2$ and then increases again. The idea behind this strategy is that HMP will be efficient at identifying improvements at a Hamming distance of $n/2$ only if a

large fraction of such points are improving ones. On the other hand, for smaller or larger distances the probability of identifying improvements is much higher. We introduce this mechanism in Section 7.2. In Section 7.3, we rigorously prove that the static HMP using such a mechanism, which we call *probabilistic first constructive mutation* (FCM_p), locates local optima asymptotically as fast as RLS for any function where the expected runtime of RLS can be proven with the AFL method (introduced in Subsection 3.3.3, page 37). At the same time, we show that the operator is still exponentially faster than EAs for the standard multimodal JUMP, CLIFF, and TRAP benchmark functions in Section 7.4.

In Chapters 4 and 5, we proved that the FCM mechanism does not allow HMP to produce solutions of lower quality, thus cancelling the advantages of ageing. Furthermore, the high mutation rates of hypermutations combined with FCM make the algorithm return to the previous local optimum with very high probability. While the latter problem is solved by our newly proposed FCM_p mechanism that does not evaluate all bit-flips in a hypermutation, a solution to the former problem requires a further modification. Hence, we propose the *probabilistic best found mutation* mechanism (BFM_p, introduced in Section 7.2) which returns the best solution it has found if no constructive mutation is encountered. We rigorously prove in Section 7.5 that Opt-IA then benefits from both operators for all problems where it was previously analysed in this thesis as desired, i.e., hypermutations and ageing work effectively and in harmony together.

Finally, after evaluating the performance of the $(1+1)$ IA with the new strategies on standard benchmark problems, in Section 7.6 we show that they are also efficient for optimising instances of more realistic problems from combinatorial optimisation, i.e., PARTITION and VERTEX COVER. In Subsection 7.6.1, we first show that the $(1+1)$ IA using static HMP with FCM_p optimises the worst case PARTITION instance for the $(1+1)$ EA more efficiently than the $(1+1)$ IA using the traditional static HMP using FCM. Then, we prove that a $(1+\varepsilon)$ approximation is achievable for any PARTITION instance by the $(1+1)$ IA using static HMP with FCM_p in $O(n^2)$ for any constant ε, i.e., a linear speed-up compared to static HMP using FCM. This result is particularly surprising because the AIS relies on flipping approximately $n/2$ bits, i.e., where our parabolic probability is the lowest. Afterwards, in Subsection 7.6.2, we analyse the performance of the $(1+1)$ IA using static HMP with FCM and also FCM_p for the VERTEX COVER problem. We follow a similar approach to [34] which has proved for the $(1+1)$ EA a tight bound of $\Theta(n \log n)$ for finding a feasible vertex cover for a graph (i.e., not necessarily the minimum vertex cover). We show that linear speed-ups are achieved by the use of static HMP with FCM_p compared to the standard static HMP using FCM, thus matching the bound of the $(1+1)$ EA. We use the term *fast* to call the operators and algorithms that use static HMP with either FCM_p or BFM_p.

7.2 Parabolic Sampling Distributions

Recall that static HMPs using FCM mutate at most $M = cn$ distinct bits for a constant $0 < c \leq 1$ and evaluate the fitness after each bit-flip. Although FCM prevents the hypermutation operator to waste further fitness function evaluations if an improvement has already been found, for any realistic objective function the number of iterations where there is an improvement constitutes an asymptotically small fraction of the total runtime. Hence, the fitness function evaluations saved due to FCM stopping the hypermutations, have a very small impact on the global performance of the algorithm. Therefore, we propose an alternative for FCM, called FCM$_p$, that instead only evaluates the fitness after each bit-flip with a probability that depends on how many bits have already been flipped in the current hypermutation operation. Let p_i be the probability that the solution is evaluated after flipping the ith bit. The used parabolic probability distribution is defined as follows, where $0 < \gamma \leq 1$[1]:

$$p_i = \begin{cases} 1/e & \text{for } i = 1 \text{ and } i = n, \\ \gamma/i & \text{for } 1 < i \leq n/2, \\ \gamma/(n-i) & \text{for } n/2 < i < n. \end{cases} \tag{7.1}$$

The lower the value of γ, the fewer the expected fitness function evaluations that occur in each hypermutation. On the other hand, with a small enough value for parameter γ, the number of wasted evaluations can be dropped to the order of $O(1)$ per iteration instead of the linear amount wasted by the traditional FCM operator when improvements are not found. At the same time, it still flips many bits (i.e., hypermutating). Figure 7.1 shows the evaluation probabilities for two different choices of γ. A static HMP that uses FCM$_p$ is formally defined in Definition 6 and its pseudo-code is shown in Function 7.1.

Definition 6 (Static HMP with FCM$_p$)**.** *Static HMP with FCM$_p$ flips at most n distinct bits selected uniformly at random. It evaluates the fitness after the i th bit-flip with probability p_i (as defined in (7.1)) and remembers the last evaluation. FCM$_p$ stops flipping bits when it finds an improvement; if no improvement is found, it will return the last evaluated solution. If no evaluations are made, the parent will be returned.*

In the next section, we will prove the benefits of FCM$_p$ over the standard FCM, when incorporated into a $(1+1)$ framework (by calling Function 7.1 in Algorithm 2.4, page 21). However, for the operator to work effectively in conjunction with ageing, a further modification is required. Instead of stopping the hypermutation at the first constructive mutation, we

[1] Note that $0 < \gamma < 1/e$ is an efficient choice for the results we will present.

Function 7.1: Fast HMP with FCM_p

```
input  : x ∈ {0,1}^n;
output : y;
y := x;
count := 0;
while count < n do
    count := count + 1;
    choose i ∈ {1, ··· , n} uniformly at random without replacement;
    flip y[i];
    evaluate y with probability defined by (7.1);
    if f(y) ≥ f(x) then  // or f(y) > f(x) depending on how constructive
     mutation is defined.
        break;
return y as the offspring;
```

Function 7.2: Fast HMP with BFM_p

```
input  : x ∈ {0,1}^n;
output : y;
y := x;
count := 0;
best := (x, −∞);
while count < n do
    count := count + 1;
    choose i ∈ {1, ··· , n} uniformly at random without replacement;
    flip y[i];
    evaluate y with probability defined by (7.1);
    if f(y) ≥ best[2] then
        best := (y, f(y));
return the best best[1] as the offspring;
```

will execute all n mutation steps, evaluate each bit string with the probabilities in (7.1) and as the offspring, return the best solution evaluated during the hypermutation or the parent itself if no other bit strings are evaluated. We will prove that such a modified operator, which we call BFM_p (Best Found Mutation), may allow the complete Opt-IA to escape local optima more efficiently. This happens because BFM_p can produce solutions of lower quality than the local optimum on which the algorithm is stuck while individuals on the local optimum die due to ageing. Static HMP with BFM_p is shown in Function 7.2 and formally defined as follows.

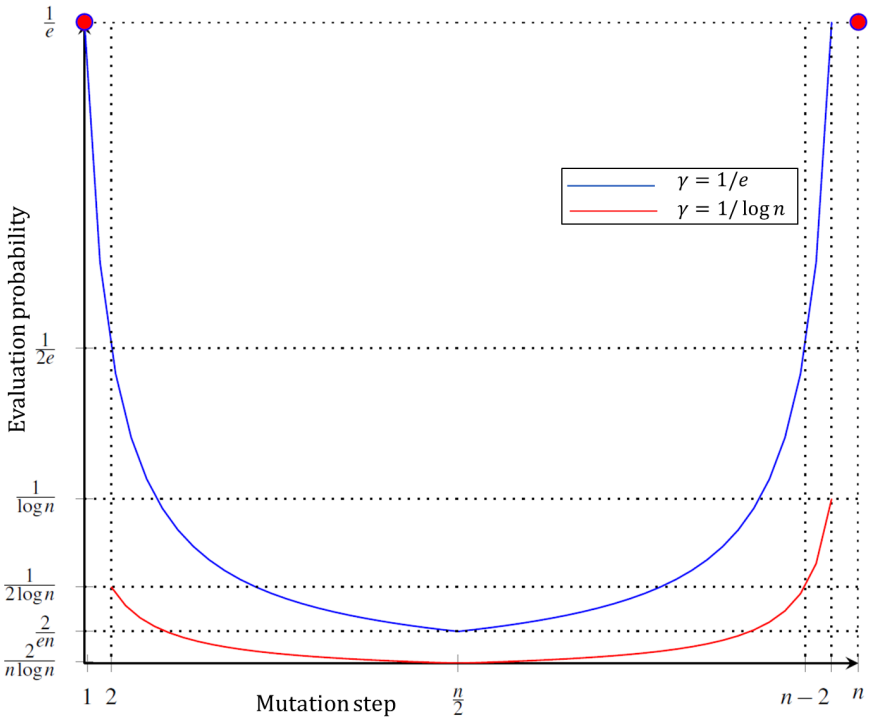

Fig. 7.1 The parabolic evaluation probabilities from (7.1)

Definition 7 (Static HMP with BFM$_p$)**.** *Static HMP with BFM$_p$ flips n distinct bits selected uniformly at random. It evaluates the fitness after the i th bit-flip with probability p_i (as defined in (7.1)) and remembers the best evaluation found so far. BFM$_p$ returns the mutated solution with the best fitness found. If no evaluations are made, the parent will be returned.*

For $\gamma = O(1/(\ln n))$, only one function evaluation per hypermutation is performed in expectation (although all bits will be flipped). Since BFM$_p$ returns the best found one, this solution will be returned by the operator as it is the only one it has encountered. Interestingly, this behaviour is similar to that of the HMP without FCM that also evaluates one point per hypermutation and returns it. However, while HMP without FCM has exponential expected runtime for any function of interest with a polynomial number of optima (see Theorem 12, page 57), we will show that static HMP with BFM$_p$ can be very efficient. From this point of view, BFM$_p$ is a very effective way to perform hypermutations with mutation potential without FCM.

In Section 7.5, we consider static HMP with BFM$_p$ in the complete Opt-IA framework hence analyse its performance combined with cloning and ageing. This algorithm, which we call Fast Opt-IA, is depicted in Algorithm 7.3. After initialising a population of μ b-cells with $age := 0$, at each iteration, the algorithm creates dup copies of each b-cell. These copies are mutated by the fast hypermutation operator, creating a population of mutants called $P^{(\mathrm{hyp})}$

which inherit the age of their parents if they do not improve the fitness; otherwise their age will be set to zero. At the next step, all b-cells with $age \geq \tau$ will be removed with probability p_{die}. If fewer than μ individuals have survived ageing, then the population is filled up with new randomly generated individuals. At the selection phase, the best μ b-cells are chosen to form the population for the next generation.

Algorithm 7.3: Fast Opt-IA

```
1  begin
2      t := 0;
3      initialise P := {x_1, ..., x_μ}, a population of μ b-cells uniformly at random;
4      for all x ∈ P do
5          x.age := 0;
6      while a global optimum is not found do
7          P^(clo) := ∅ and P^(hyp) = ∅;
8          for all x ∈ P do
9              make dup copies of x and add to P^(clo);
10         for all x ∈ (P^(clo) ∪ P) do
11             x.age := x.age + 1;
12         for all x ∈ P^(clo) do
13             call Function 7.2;
14             if f(y) > f(x) then
15                 y.age := 0;
16             else y.age := x.age;
17             add y to P^(hyp);
18         for all x ∈ (P ∪ P^(hyp)) do
19             if x.age > τ then
20                 remove x with probability p_die := 1 − 1/(μ+(μ·dup));
21         P := (P ∪ P^(hyp));
22         if |P| > μ then
23             remove |P| − μ solutions with worst fitness from P breaking ties
                uniformly at random.
24         if |P| < μ then
25             add μ − |P| solutions to P with age := 0 generated uniformly at random;
26         t := t + 1;
```

7.3 Artificial Fitness Levels for Fast Hypermutations

We start our analysis by relating the expected number of fitness function evaluations to the expected number of fast hypermutation operations until an optimum is found. Lemma 14 holds for both FCM_p and BFM_p mechanisms and quantifies the number of expected fitness function evaluated by using these two schemes. To prove the result, we use the following lemma known as Wald's equation.

Lemma 13 (Wald's equation [64])**.** *Let $X_1, X_2, \cdots, X_i$ be non-negative, independent, identically distributed random variables with distribution X. Let T be the stopping time for this sequence. If T and X have bounded expectation, then $E[\sum_{i=1}^{T} X_i] = E[T] \cdot E[X]$.*

Lemma 14. *Let T be the random variable denoting the number of fast hypermutation operations applied until the optimum is found. The expected number of function evaluations in a fast hypermutation operation given that no improvement is found is in the order of $\Theta(1+\gamma \log n)$. Moreover, the expected number of total function evaluations is at most $O(1+\gamma \log n) \cdot E[T]$.*

Proof. Let the random variable X_i for $i \in [T]$ denote the number of fitness function evaluations during the ith execution of the fast hypermutation. Additionally, let the random variable X_i' denote the number of fitness function evaluations at the ith operation assuming that no improvements have been found before. For all i it holds that $X_i \leq X_i'$ since finding an improvement can only decrease the number of evaluations. Thus, the total number of function evaluations $E[\sum_{i=1}^{T} X_i]$ can be bounded from above by $E[\sum_{i=1}^{T} X_i']$ which is equal to $E[T] \cdot E[X']$ due to Wald's equation (stated in Lemma 13) since all X_i' are identically distributed and independent from T.

We now write the expected number of fitness function evaluations in each operation as the sum of n indicator variables $Y_i \in \{0,1\}$ for $i \in [n]$ denoting whether an evaluation occurs right after the ith bit mutation. Referring to the probabilities in (7.1), we get $E[X'] = E\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} P(Y_i = 1) = \frac{1}{e} + \frac{1}{e} + 2\sum_{i=2}^{n/2} \frac{\gamma}{i} = \frac{2}{e} + 2\gamma\Theta(\log n) = \Theta(1+\gamma \log n)$. The second statement is obtained by multiplying this amount with $E[T]$. □

In Lemma 14, γ appears as a multiplicative factor in the expected runtime measured in fitness function evaluations. An intuitive lower bound of $\Omega(1/\log n)$ for γ can be inferred since smaller asymptotic values will not decrease the asymptotic order of expected evaluations per operation. However, in Section 7.5, we will provide an example where a smaller choice of γ reduces $E[T]$ directly. For the rest of our results though, we will rely on $E[T]$ being the same as for the traditional static hypermutations with FCM while the number of wasted fitness function evaluations decreases from n to $O(1+\gamma \log n)$.

We now analyse the fast hypermutation operators in a $(1+1)$ framework, starting by establishing a relationship between RLS and the Fast $(1+1)$ IA (similar to Theorem 13 in Chapter 4 which shows the relationship between RLS and $(1+1)$ IA).

Theorem 45. *Let $E\left(T_A^{AFL}\right)$ be any upper bound on the expected runtime of algorithm A established by the artificial fitness levels method. Then,*

$$E\left(T^{AFL}_{\textit{Fast}\,(1+1)\ \textit{IA with}\ \textit{FCM}_p}\right) \leq E\left(T^{AFL}_{(1+1)\ \textit{RLS}}\right) \cdot O(1+\gamma\log n).$$

Proof. The upper bound on the expected runtime of RLS to solve any function obtained by applying AFL is $E(T^{AFL}_{(1+1)\ RLS}) \leq \sum_{i=1}^{m} 1/p_i$, where p_i is s/n when all individuals in level i have at least s Hamming neighbours which belong to a higher fitness level. The probability of mutating to one of the solutions in the first mutation step is the same for FCM_p. Such a solution will be evaluated with probability $1/e$. If a solution is not found in the first mutation step, then according to Lemma 14 at most $O(1+\gamma\log n)$ fitness function evaluation would be wasted. Since the algorithm is elitist and only accepts individuals of equal or better fitness, each level has to be left only once, independent of whether improvements are achieved by one or more bit-flips. Hence the claim follows. □

Apart from showing the efficiency of the Fast $(1+1)$ IA, the theorem also allows us to easily achieve upper bounds on the runtime of the algorithm, by just analysing the simple RLS. For $\gamma = O(1/\log n)$, Theorem 45 implies the upper bounds of $O(n\log n)$ and $O(n^2)$ for classical benchmark functions ONEMAX and LEADINGONES, respectively. Both of these bounds are asymptotically tight since each function's unary unbiased black-box complexity is in the same order as the presented upper bound [57] (i.e., no unbiased mutation-only black box search heuristic may be asymptotically faster).

7.4 Fast Hypermutations for Benchmark Functions

In this section, we will use the mathematical methodology derived above to show that linear speed-ups may be achieved by the fast hypermutation operators compared to the static HMP using FCM. Afterwards, we show that exponential speed-ups are still achievable for the standard multimodal benchmark functions. We will also show that for some of these functions the fast hypermutations may outperform the static HMPs with FCM.

Corollary 6. *The expected runtime of the Fast $(1+1)$ IA using static HMP with FCM_p to optimise* ONEMAX *and* LEADINGONES *is respectively $\Theta\left(n\log n\left(1+\gamma\log n\right)\right)$ and $\Theta(n^2\left(1+\gamma\log n\right))$. For $\gamma = O(1/\log n)$ these bounds reduce to $\Theta(n\log n)$ and $\Theta(n^2)$.*

FCM_p samples the complementary bit string with probability one if it cannot find any improvements. This behaviour allows an efficient optimisation of the deceptive TRAP function (defined in (3.10), page 44). Since n bits have to be flipped to reach the global optimum from the local optimum, EAs with SBM require exponential runtime w.o.p. [78]. By evaluating the sampled bit strings stochastically, the Fast $(1+1)$ IA provides up to a linear speed-up for small enough γ compared to the $(1+1)$ IA on TRAP as well (i.e., it is a linear factor faster at reaching the local optimum).

Theorem 46. *The Fast* $(1+1)$ *IA using static HMP with* FCM_p *needs* $\Theta(n\log n\,(1+\gamma\log n))$ *expected steps to optimise* TRAP.

Proof. According to Corollary 6, we can conclude that the current individual will reach 1^n in $O(n\log n\cdot(1+\gamma\log n))$ steps in expectation. The global optimum is found in a single mutation operator with probability $1/e$ by evaluating after flipping all bits for which the number of additional fitness evaluations is $O(1+\gamma\log n)$ in expectation. This bound is asymptotically tight since each function's unary unbiased black-box complexity is in the same order as the presented upper bound [57].

□

The results of the $(1+1)$ IA on JUMP_k (see Theorem 16, page 60) and CLIFF_d functions (see Corollary 1, page 61) can also be adapted to the Fast $(1+1)$ IA in a straightforward manner, even though they fall out of the scope of Theorem 45.

Theorem 47. *The expected runtime of the Fast* $(1+1)$ *IA using static HMP with* FCM_p *to optimise* CLIFF_d *is* $O\left(\binom{n}{d}\cdot(d/\gamma)\cdot(1+\gamma\log n)\right)$ *which also holds for the* JUMP *function.*

Proof. As CLIFF_d behaves as ONEMAX for solutions with less than $n-d$ 1-bits, we can use Corollary 6 to conclude that the algorithm will sample a solution with $n-d$ 1-bits in at most $O\left(n\log n\,(1+\gamma\log n)\right)$ fitness function evaluations. The Hamming distance of locally optimal points to the global optimum is d, thus, the probability of reaching the global optimum at the dth mutation step is $\binom{n}{d}^{-1}$ while the probability of evaluating is γ/d. Using Lemma 14, we bound the total expected time to optimise CLIFF by $E(T)=O\left(\binom{n}{d}\cdot(d/\gamma)\cdot(1+\gamma\log n)\right)$. The proof is exactly the same for JUMP. □

For CLIFF_d (and JUMP_k), the superiority of the Fast $(1+1)$ IA in comparison to the deterministic evaluations scheme (i.e., FCM) depends on the function parameter d (or k for JUMP_k). If $\gamma=\Omega(1/\log n)$, the Fast $(1+1)$ IA performs better for $d=o(n/\log n)$ while the deterministic scheme is preferable for larger d. However, for small d the difference between the runtimes can be as large as a factor of n in favour of the Fast $(1+1)$ IA, while even for

Table 7.1 The expected runtimes of the $(1+1)$ IA and the Fast $(1+1)$ IA

Function	$(1+1)$ IA	Fast $(1+1)$ IA
ONEMAX	$\Theta(n^2 \log n)$	$\Theta(n \log n (1+\gamma \log n))$
LEADINGONES	$\Theta(n^3)$	$\Theta\left(n^2 (1+\gamma \log n)\right)$
TRAP	$\Theta(n^2 \log n)$	$\Theta(n \log n (1+\gamma \log n))$
$\text{JUMP}_{k>1}$	$O(n\binom{n}{k})$	$O\left((k/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{k}\right)$
$\text{CLIFF}_{d>1}$	$O(n\binom{n}{d})$	$O\left((d/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{d}\right)$

the largest d, the difference is less than a factor of $\log n$ in favour of the deterministic scheme. Here we should also note that for $d = \Omega(n/\log n)$, the expected time is exponentially large for both algorithms (albeit considerably smaller than that of standard EAs) and the $\log n$ factor has no realistic effect on the applicability of the algorithm. Hence, we believe that the fast variant should be more efficient in practice.

Table 7.1 shows the expected runtimes of the $(1+1)$ IA using static HMP with FCM versus the expected runtime of the Fast $(1+1)$ IA. For $\gamma = O(1/\log n)$, the Fast $(1+1)$ IA is asymptotically at least as fast as the $(1+1)$ EA and faster by a linear factor compared to the $(1+1)$ IA for the unimodal and TRAP functions. For not too large jump and cliff sizes (i.e., $o(n/\log n)$), the Fast $(1+1)$ IA has an asymptotic speed-up compared to the $(1+1)$ IA for the same parameter setting. For not too small jump and cliff sizes both AISs are much faster than the $(1+1)$ EA.

7.5 Fast Opt-IA for Benchmark Functions

In this section, we consider the effect of our proposed evaluation scheme on the complete Opt-IA algorithm. First we start by proving its effectiveness on easy functions like ONEMAX and LEADINGONES.

Theorem 48. *The Fast Opt-IA using static HMP with BFM_p optimises* ONEMAX *and* LEADINGONES *in expected $O(\mu \cdot dup \cdot n \log n \cdot (1+\gamma \log n))$ and $O(\mu \cdot dup \cdot n^2 \cdot (1+\gamma \log n))$ fitness function evaluations respectively for $\tau = c \cdot n \log n$ for some constant c.*

Proof. For ONEMAX, we pessimistically track only the best individual. Considering the improvement by the first bit-flip, in at most $\sum_{i=1}^{n} en/i = O(n \log n)$ generations the optimum will be found where i represents the number of 0-bits. Taking into account the wastage of $O(1+\gamma \log n)$ fitness evaluations in case of failure and the evaluation of $\text{dup} \cdot \mu$ b-cells at

each generation, we get $O(\mu \cdot \text{dup} \cdot n \log n \cdot (1+\gamma \log n))$ as the upper bound on the expected runtime. Since the improvement probability is at least $\frac{1}{en}$ at each step, the improving b-cell would not die w.o.p. before reaching the optimum.

We show the upper bound on LEADINGONES using that the probability of improvement is at least $\frac{1}{en}$ in the first bit-flip, and get the bound of $O(\mu \cdot \text{dup} \cdot n^2 \log n \cdot (1+\gamma \log n))$ on the expected number of fitness function evaluations needed for optimising LEADINGONES since at most n improvements have to be made. □

In Chapter 4, we saw that HIDDENPATH (see Definition 2, page 70) is a function where the use of both ageing and hypermutation operators is necessary to find the optimum in polynomial time. To prove our upper bound, we can follow the proof strategy of Theorem 22 in Chapter 4 (page 71), which established an upper bound of $O(\tau\mu n + \mu n^{7/2})$ for the expected runtime of Opt-IA using static HMP with traditional FCM on HIDDENPATH. We will see that Opt-IA benefits from an $n/\log n$ speed-up (regarding the second term in the runtime which is the dominant term) in the provided upper bound due to BFM_p.

Theorem 49. *The Fast Opt-IA using static HMP with BFM_p needs $O(\tau\mu + \mu n^{5/2} \log n)$ fitness function evaluations in expectation to optimise* HIDDENPATH *with $\mu = O(\log n)$, $\text{dup} = 1$, $\gamma = \Omega(1/\log n) \leq 1/(5 \ln n)$ and $\tau = \Omega(n(\log n)^3)$.*

Proof. For simplicity during the analysis, we call a non-SP point an S_i solution where i is the number of 0-bits. We also pessimistically assume that until the very end, the global optimum point is not evaluated.

After initialisation, an S_{n-1} solution will be found in expected $O(n(\log n)^2)$ generations by hill-climbing the ZEROMAX part of the function according to Theorem 48. This individual creates and evaluates another S_{n-1} search point with probability $\gamma/(2n)$ (i.e., with probability $(n-1)/n$ a 0-bit is flipped and then the 1-bit is flipped with probability $1/(n-1)$, and the solution will be evaluated with probability $\gamma/2$ after the second bit-flip). Hence, after at most $\mu \cdot O(n)$ generations in expectation, the whole population will consist only of S_{n-1} solutions. Considering that the probability of producing two S_{n-1} in one generation is $\binom{\mu}{2} \cdot O(\gamma/n) \cdot O(\gamma/n) = O(1/n^2)$, with probability at least $1 - o(1)$ we see at most one new S_{n-1} per generation for any phase length of $o(n^2)$ generations. Taking into account that the probability of creating a new S_{n-1} individual is $\gamma/(2n)$ and following the proof of Theorem 22 (page 71), we can conclude that in $O(\mu^3 \cdot n/\gamma)$ generations in expectation, the whole population reaches the same age while on the local optimum. Using Markov's inequality iteratively (see Theorem 6, page 35) we can bound the probability that a population that consists of S_{n-1} individuals of the same age will be observed in at most $O(\mu^3 \cdot n(\log n)^2)$ generations with probability $1 - o(1)$. Then, after at most τ generations, with probability $(1 -$

$1/(2\mu))^{2\mu-1} \cdot 1/(2\mu)$, one solution survives and the rest are removed from the population. In the following generation, while $\mu - 1$ randomly initialised solutions are added instead of the removed solutions, we consider the event that the survived solution creates and evaluates an S_1 solution. The expected number of evaluations between the second and $(n-1)$th mutation step is $2\sum_{i=2}^{n/2} \gamma/i \leq 1/2$ since $\gamma \leq 1/(5\ln n)$. The probability that there is at least one evaluation is at most $1/2$ by Markov's inequality, therefore with probability at least $(1-\frac{1}{e}) \cdot \frac{1}{2} \cdot \frac{1}{e} = \Omega(1)$, only the S_1 solution is evaluated. If an S_1 or S_5 solution is not created before the newly generated individuals reach the S_{n-1} level, we repeat the arguments starting from the takeover of the population by S_{n-1} individuals. Since the jump to S_1 occurs with at least constant probability, in expectation we repeat the same process at most a constant number of times and the runtime until the success has the same asymptotic order as the runtime until the first attempt, i.e., $O(\mu^3 \cdot n(\log n)^2)$ hypermutation operations.

After the S_{n-1} individual is added to the population, from this search point, the hypermutation operator finds an S_5 solution by flipping at most six bits and evaluating it with probability at least $\gamma/6$, which requires $O(\log n)$ attempts in expectation which in turn implies that the expected time until this event occurs is $O(\mu^3 \cdot n(\log n)^3)$. This individual will be added to the population with its age set to zero if the complementary bit string (S_{n-1}) is not evaluated with probability $(1-1/e)$. In the same generation the S_1 solution dies with probability $1-\frac{1}{2\mu}$ due to ageing.

Next, we show that the S_5 solutions will take over the population, and the first point of SP will be found before any S_{n-1} is found. An S_5 creates an S_{n-5} individual and an S_{n-5} individual creates an S_5 individual with constant probability $1-(1/e)$ by evaluating complementary bit strings. Thus, it takes $O(1)$ generations until the number of S_5 and S_{n-5} individuals in the population doubles. Since the total number of S_5 and S_{n-5} increases exponentially in expectation, in $O(\log \mu) = O(\log \log n)$ generations the population is taken over by them. After the takeover, since each S_{n-5} solution creates an S_5 solution with constant probability, in the following $O(1)$ generations in expectation each S_{n-5} creates an S_5 solution which have higher fitness value than their parents and replace them in the population. Overall, S_5 solutions take over the population in $O(\log \log n)$ generations in expectation.

For S_5, HIDDENPATH has a gradient towards the SP which favours solutions with more 0-bits in the first (the rightmost) five bit positions. Every improvement on the gradient takes $O((2/\gamma) \cdot n^2)$ generations in expectation since it is enough to flip a precise 1-bit and a precise 0-bit in the worst case. Considering that there are five different fitness values on the gradient, in $O(5 \cdot 2 \cdot n^2/\gamma) = O(n^2/\gamma)$ generations in expectation the first point of the SP will be found. Applying Markov's inequality, this time will not exceed $O(n^{5/2}/\gamma)$ with probability at least $1-(1/\sqrt{n})$.

Now we go back to the probability of finding a locally optimal point before finding an SP point. Due to the symmetry of the hypermutation operator, probability of creating an S_{n-1} solution from an S_5 solution is identical to the probability of creating an S_{n-1} solution from an S_{n-5} solution. The probability of increasing the number of 0-bits by k given that the initial number of 1-bits is i and the number of 0-bits is $n-i$, is at most $(2i/n)^k$ due to the Ballot theorem (see Theorem 11, page 38) since each improvement reinitialises a new ballot game with higher disadvantage. Thus, the probability that a local optimal solution is sampled is $O(n^{-4})$. The probability that such an event never happens before finding SP is $1-o(1)$. After finding SP, in $O(n\log n)$ generations in expectations the global optimum will be found at the end of the SP. The probability of finding any locally optimal point from SP is at most $O(1/n^4)$, hence this event would not happen before reaching the global optimum with probability $1-o(1)$. Overall, the runtime is dominated by $O(\tau+n^{5/2}/\gamma)$ which give us $O((\tau+n^{5/2}/\gamma)\cdot\mu(1+\gamma\log n))$ as the expected number of fitness evaluations. Since $\Omega(1/\log n)=\gamma\leq 1/(5\ln n)$, the upper bound reduces to $O(\tau\mu+\mu n^{5/2}\log n)$. □

HIDDENPATH was artificially constructed to fit the behaviour of the Opt-IA to illustrate its strengths. One of those strengths was the ageing mechanism's ability to escape local optima in two different ways. First, it allows the algorithm to restart with a new random population after it gets stuck at a local optimum. Second, ageing allows individuals with worse fitness than the current best to stay in the population when all the current best individuals are removed by the ageing operator in the same iteration. If an improvement is found soon after the worsening is accepted, then this temporary non-elitist behaviour allows the algorithm to follow other gradients which are accessible by variation from the local optima but leads away from them. On the other hand, even though the FCM mechanism is coupled with ageing in the Opt-IA, it does not allow worsenings. More precisely, for the hypermutation with FCM, the complementary bit string of the local optimum is sampled with probability one if no other improvements are found. Indeed, HIDDENPATH was designed to exploit this ability. However, by only stopping on improving mutations, the traditional hypermutations with FCM do not allow, in general, to take advantage of the power of ageing at escaping local optima. For instance, for the classical benchmark function CLIFF_d with parameter $d=\Theta(n)$, hypermutation with FCM turned out to be a worse choice of variation operator to couple with ageing than both local search and SBM (see Theorem 19, page 68). Ageing coupled with RLS and SBM based EAs can help the algorithm to reach the optimum by local moves, which respectively yields upper bounds of $O(n\log n)$ and $O(n^{1+\varepsilon}\log n)$ for arbitrarily small positive constant ε on the runtimes (see Theorems 17 and 18 in Chapter 4, page 64). However, hypermutations with FCM require an increase in the number of 1-bits in the current solution by d at least once before the hypermutation stops. This requirement implies the

exponential lower bound on the runtime for the CLIFF function in Theorem 19 in Chapter 4 (page 68), regardless of the evaluation scheme as long as the hypermutation only stops on a constructive mutation.

The following theorem demonstrates how BFM_p (which instead of stopping the hypermutation at the first constructive mutation, executes all n mutation steps and evaluates each bit string with the probabilities in (7.1) and returns the best found solution) allows ageing and hypermutation to work in harmony in Opt-IA.

Theorem 50. *The Fast Opt-IA using static HMP with BFM_p with $\mu = 1$, $dup = 1$, $\gamma = 1/(n\log^2 n)$ and $\tau = \Theta(n\log n)$ needs $O(n\log n)$ fitness function evaluations in expectation to optimise CLIFF_d with any linear $d \leq n/4 - \varepsilon$ for an small constant ε.*

Proof. With $\gamma = 1/(n\log^2 n)$, the expected number of fitness function evaluations per iteration $O(1+\gamma\log n)$ would be in the order of $\Theta(1)$. On the first ONEMAX slope, the algorithm improves by the first bit-flip with probability at least $(n-d)/n = \Theta(1)$ and then evaluates this solution with probability $p_1 = 1/e = \Theta(1)$. This implies that the local optimum will be found in $O(n)$ fitness evaluations in expectation after initialisation.

A solution at the local optimum can only improve by finding the unique globally optimum solution, which requires the hypermutation to flip precisely d 0-bits in the first d mutation steps which occurs with probability $\binom{n}{d}^{-1}$. We pessimistically assume that this direct jump never happens and assume that once a solution at the local optimum is added to the population, it reaches age τ in some iteration t_0. We consider the following chain of events that starts at t_0. First a solution with $(n-d+1)$ 1-bits would be added to the population with probability $\frac{1}{e}\cdot\frac{d}{n}$. Then, the locally optimal solution will die due to ageing with probability $1/2$ while the post-cliff solution will survive with probability $1/2$. In the next iteration, the post-cliff solution would improve the fitness with constant probability and hence resets its age to zero. If all of these events occur consecutively (which happens with constant probability), the algorithm can start climbing the second ONEMAX slope with local moves (i.e., by considering only the first mutation steps) which are evaluated with constant probability. Then, the CLIFF_d function is optimised in $O(n\log n)$ expected fitness function evaluations like ONEMAX unless a pre-cliff solution (i.e., a solution with less than $n-d$ 1-bits) replaces the current individual. The rest of our analysis will focus on the probability that a pre-cliff solution is sampled and evaluated given that the algorithm has a post-cliff solution with age zero at iteration t_0+1.

If the current solution is a post-cliff solution, then the final bit string sampled by the hypermutation has a worse fitness level than the current individual. The probability that BFM_p evaluates at least one solution between mutation steps two and $n-1$ (event $\mathscr{E}_{nv}$),

is bounded from above by $\sum_{n-1}^{i=2} \gamma/i < 2\gamma \cdot \log n = 2/(n \log n)$. We consider the $O(n \log n)$ generations until a post-cliff solution (i.e., solutions with more than $n-d$ 1-bits) with age zero reaches the global optimum. The probability that event $\mathscr{E}_{mv}$ never occurs in any iteration until the optimum is found is at least $(1-2/(n \log n))^{O(n \log n)} = e^{-O(1)} = \Omega(1)$, a constant probability. Thus, every time we create a post-cliff solution with age zero, there is at least a constant probability that the global optimum is reached before any solution that is not sampled at the first or the last mutation step gets evaluated. The first mutation step cannot yield a pre-cliff solution, and the last mutation step cannot yield a solution with better fitness value. Thus, with a constant probability the post-cliff solution finds the optimum. If it fails to do so (i.e., a pre-cliff solution takes over as the current solution or a necessary event does not occur at iteration t_0+1), then in at most $O(n \log n)$ iterations another chance to create a post-cliff solution comes up and the process is repeated. In expectation, a constant number of trials will be necessary until the optimum is found and since each trial takes $O(n \log n)$ fitness function evaluations, our claim follows. □

Note that the above result requires a γ in the order of $\Theta(1/(n \log^2 n))$, while Lemma 14 implies that any $\gamma = \omega(1/\log n)$ would not decrease the expected number of fitness function evaluations below the asymptotic order of $\Theta(1)$. However, having $\gamma = 1/(n \log^2 n)$ allows Opt-IA, with constant probability, to complete its local search before any solution with larger Hamming distance is ever evaluated. In Theorem 50, we observe that this opportunity allows Opt-IA to hill-climb the second slope before jumping back to the local optima. The following theorem rigorously proves that a very small choice for γ in this case is necessary (i.e., $\gamma = \Omega(1/\log n)$ leads to exponential expected runtime).

Theorem 51. *At least $2^{\Omega(n)}$ fitness function evaluations in expectation are executed before the Fast Opt-IA using static HMP with BFM$_p$ with $\gamma = \Omega(1/\log n)$ finds the optimum of* CLIFF$_d$ *for $d = (1-c)n/4$, where c is a constant $0 < c < 1$.*

Proof. Consider a current solution with more than $n-d$ and less than $n-d+2\sqrt{n}$ 1-bits. We show that w.o.p., BFM$_p$ will yield a solution with less than $n-d$ and more than $n-2d+2\sqrt{n}$ 1-bits before the initial individual is mutated into a solution with more than $n-d+2\sqrt{n}$ 1-bits. This observation will imply that a pre-cliff solution with better fitness will replace the post-cliff solution before the post-cliff solution is mutated into a globally optimal solution. We will then show that it is also exponentially unlikely that any pre-cliff solution mutates into a solution with more than $n-d+\sqrt{n}$ 1-bits to complete our proof.

We first provide a lower bound on the probability that BFM$_p$ with post-cliff input solution x yields a pre-cliff solution with higher fitness value than x. We start by determining

the earliest mutation step, r_{min}, that a pre-cliff solution with better fitness than x can be sampled. For any post-cliff solution x, $\textsc{Cliff}_d(x) = \textsc{OneMax}(x) - d + (1/2)$, and any pre-cliff solution y with $\textsc{OneMax}(x) - d + 1$ 1-bits, it holds that $\textsc{Cliff}_d(y) > \textsc{Cliff}_d(x)$. We obtain the rough bound of $r_{min} \geq d - 2\sqrt{n}$ by considering the worst-case event that static HMP with BFM_p picks d 1-bits to flip consecutively. Let $\ell(x)$ denote the number of extra 1-bits a post-cliff solution has in comparison to a locally optimal solution (i.e, $\textsc{OneMax}(x) = n - d + \ell(x)$). Now, we use Serfling's inequality (see Theorem 9, page 36) to show that with a constant probability, static HMP with BFM_p will find a pre-cliff solution before $3\ell(x)$ mutation steps and it will keep sampling pre-cliff solutions until r_{min}.

For the input bit string of BFM_p, x, let the multi-set of weights $C := \{c_i | i \in [n]\}$ be defined as $c_i := (-1)^{x_i}$ (i.e., $c_i = -1$ when $x_i = 1$, and $c_i = 1$ when $x_i = 0$). Thus, for a permutation π of bit-flips over $[n]$, the number of 1-bits after the kth mutation step is $\textsc{OneMax}(x) + \sum_{j=1}^{k} c_{\pi_j}$ since flipping the position i implies that the number of 1-bits changes by c_i. Let $\bar{\mu} := (1/n)\sum_{j=1}^{n} c_i$ be the population mean of C and $\bar{X} := (1/3\ell(x))\sum_{j=1}^{3\ell(x)} c_{\pi_j}$ the sample mean. Since the $\textsc{Cliff}_d$ parameter d is less than $n/4$, then $\bar{\mu} \leq (1/n)\left((-3n/4) + (n/4)\right) = -1/2$.

In order to have a solution with at least $n - d + 1$ 1-bits at mutation step $3\ell(x)$, the following must hold:

$$
\begin{aligned}
& 3\ell(x)\bar{X} \geq -\ell(x) \iff \bar{X} \geq -\frac{1}{3} \\
& \implies \bar{X} - \bar{\mu} \geq -\frac{1}{3} + \frac{1}{2} = \frac{1}{6} \iff \sqrt{3\ell(x)}\,(\bar{X} - \bar{\mu}) \\
& \geq \frac{\sqrt{3\ell(x)}}{6}.
\end{aligned}
$$

The probability that a pre-cliff solution will not be found in mutation step $3\ell(x)$ follows from Serfling's inequality, with sample mean $\bar{X}$, population mean $\bar{\mu}$, sample size $3\ell(x)$, population size n, $c_{min} = -1$ and $c_{max} = 1$:

$$
\begin{aligned}
& \mathrm{P}\left(\sqrt{3\ell(x)}\,(\bar{X} - \bar{\mu}) \geq \frac{\sqrt{3\ell(x)}}{6}\right) \\
& \leq \exp\left(-\frac{2\left(\frac{\sqrt{3\ell(x)}}{6}\right)^2}{\left(1 - \left(\frac{3\ell(x)-1}{n}\right)\right)(1-(-1))^2}\right) \leq e^{-\Omega(\ell(x))}.
\end{aligned}
$$

Thus, with probability $(1-e^{-\Omega(\ell(x))})$, we will sample the first pre-cliff solution after $3\ell(x)$ mutation steps. We focus our attention to post-cliff solutions with $1 \leq \ell(x) \leq 2\sqrt{n}$ and can conclude that for such solutions the above probability is in the order of $\Omega(1)$. Since the number of 0-bits changes by one at every mutation step, the event of finding a solution with at most $n-d$ bits implies that at some point a solution with exactly $n-d$ 1-bits has been sampled. Let $k_0 \leq 3\ell(x)$ be the mutation step where a locally optimal solution is found for the first time. Due to the Ballot theorem the probability that a solution with more than $n-d$ 1-bits is sampled after k_0 is at most $2d/n \leq 1/2$. So, with probability at least $1/2$, the static HMP with BFM_p will keep sampling pre-cliff solutions until $r_{min} \leq d-2\sqrt{n} = \Omega(n)$. We will now consider the probability that at least one of the solutions sampled between k_0 and r_{min} is evaluated. Since the evaluation decisions are taken independently from each other, the probability that none of the solutions are evaluated is:

$$\begin{aligned}
&\prod_{i=k_0}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \leq \prod_{i=3\ell(x)}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \leq \prod_{i=6\sqrt{n}}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \\
&\leq \prod_{i=6\sqrt{n}}^{r_{min}} \left(1-\frac{1}{(c_1 \log n) i}\right),
\end{aligned}$$

for some constant c_1 since $\gamma = \Omega(1/\log n)$. We will separate this product into $\lfloor \log(r_{min}/6\sqrt{n}) \rfloor$ smaller products and show that each smaller product can be bounded from above by $e^{-1/(2c_1 \log)}$. The first subset contains the factors with indices $i \in \{(r_{min}/2)+1, \ldots, r_{min}\}$, the second set $i \in \{(r_{min}/4)+1, \ldots, r_{min}/2\}$ and jth set (for any $j \in [\lfloor \log(r_{min}/6\sqrt{n}) \rfloor]$) $i \in \{r_{min}2^{-j}+1, \ldots, r_{min}2^{-j+1}\}$. If some indices are not covered by these sets due to the floor operator, we will ignore them since they can only make the final product smaller. Note that we assume any logarithm's base is two unless it is specified otherwise;

$$\begin{aligned}
&\prod_{j=1}^{\lfloor \log(r_{min}/6\sqrt{n}) \rfloor} \prod_{i=r_{min}2^{-j}+1}^{r_{min}2^{-j+1}} \left(1-\frac{1}{(c_1 \log n) i}\right) \\
&\leq \prod_{j=1}^{\lfloor \log(r_{min}/6\sqrt{n}) \rfloor} \left(1-\frac{2^{j-1}}{(c_1 \log n) r_{min}}\right)^{r_{min}2^{-j}} \\
&\leq \prod_{j=1}^{\lfloor \log(r_{min}/6\sqrt{n}) \rfloor} e^{-1/(2c_1 \log n)} \\
&\leq e^{-\lfloor \log(r_{min}/6\sqrt{n}) \rfloor/(2c_1 \log n)} = e^{-\Omega(1)},
\end{aligned}$$

where in the second line we made use of the inequality $(1-cn^{-1})^n \leq e^{-c}$ and in the final line our previous observation that $r_{min} = \Omega(n)$. This implies that at least one of the sampled

pre-cliff individuals will be evaluated at least with constant probability. At this point, we have established that a pre-cliff solution will be added to the population with constant probability if the initial post-cliff solution x has a distance between $\sqrt{n}$ and $2\sqrt{n}$ to the local optima.

Let $\mathscr{E}_{i,k}$ for $k > 1$ denote the event that hypermutation samples a solution with $i-k$ 0-bits given that the initial solution has i 0-bits. Since the number of 0-bits change by one at every mutation, $\mathscr{E}_{i,k}$ implies $\mathscr{E}_{i,k-1}$. In particular, for $\mathscr{E}_{i,k}$ to happen first $\mathscr{E}_{i,k-1}$ must happen and then another improvement must be found. Given $\mathscr{E}_{i,k-1}$, the probability that a new improvement is found is less than $2i/n$ because of two reasons. First, the number of 0-bits has decreased with respect to the initial solution and second, there are fewer solutions to be sampled before BFM$_p$terminates. Thus, we can conclude $P(\mathscr{E}_{i,k}) \leq P(\mathscr{E}_{i,k-1})\frac{2i}{n} \leq \left(\frac{2i}{n}\right)^k$. This implies that it is exponentially unlikely that a pre-cliff solution is mutated into a solution with more than $n-d+\sqrt{n}$ 1-bits. Moreover, the probability that post-cliff solutions are improved by more than $n^{1/6}$ is less than $4^{-n/6}$, which implies that w.o.p. it takes at least $n^{(1/2)-(1/6)} = n^{1/3}$ iterations before a solution x with $\ell(x) < \sqrt{n}$ is mutated into a solution x' with $\ell(x') > 2\sqrt{n}$. Since we established that a pre-cliff solution is evaluated with constant probability at each iteration, we can conclude that at least one such individual is sampled in $n^{1/3}$ iterations w.o.p. Since the Fast Opt-IA cannot follow the post-cliff gradient to the optimum w.o.p., it relies on making the jump from the local optima to global optimum. Given an initial solution with $y \in \{(n/3), \ldots, n-d+2\sqrt{n}\}$ 1-bits, the probability of jumping to the unique global optimum is $2^{\Omega(-n)}$ as well, thus our claim follows. □

7.6 Fast Hypermutations for Classical Combinatorial Optimisation Problems

In this section, we analyse the performance of Fast $(1+1)$ IA using static HMP with FCM$_p$ for two combinatorial optimisation problems, PARTITION (see Definition 4, page 88) and VERTEX COVER. Starting by analysing the performance on PARTITION, we first prove that the Fast $(1+1)$ IA can efficiently solve the worst case instance of PARTITION for the $(1+1)$ EA (defined in Subsection 5.4, page 93). In Chapter 5, it was proven that the $(1+1)$ IA can optimise this instance in $O(n^2)$ evaluations. In this section we will show that Fast $(1+1)$ IA can optimise the same instance in $O(n \log n)$ evaluations. Then, we show that the Fast $(1+1)$ IA can obtain a $(1+\varepsilon)$ approximation for any instance of PARTITION in $O(2en^2 \cdot (2^{2/\varepsilon}+1) \cdot (1+\gamma \log n) + (1/\varepsilon^{(2/\varepsilon+1)}(1-\varepsilon)^{-2}e^3 2^{2/\varepsilon}\frac{n}{2\gamma} \cdot (1+\gamma \log n))$ expected fitness function evaluations, i.e., a linear speed-up compared to the upper bound provided for standard static HMPs with FCM.

Afterwards, we analyse the performance of the Fast $(1+1)$ IA for VERTEX COVER for identifying a vertex cover for any graph. It has been shown in the literature that RLS and the $(1+1)$ EA can find a cover in expected $\Theta(n \log n)$ fitness function evaluations [34] while the contiguous somatic hypermutations of B-cell require expected $\Theta(n^2 \log n)$ evaluations [47].

7.6.1 Performance Analysis of Fast Hypermutations for PARTITION

Theorem 52. *The Fast* $(1+1)$ *IA using static HMP with* FCM_p *optimises the worst case instance of* PARTITION *for the* $(1+1)$ *EA (i.e.,* P^*_ε*) in* $O(\frac{n}{\gamma} \cdot (1+\gamma \log n))$ *expected fitness function evaluations.*

Proof. For $s = 2$, the proof idea is identical to that of the Theorem 32 in Chapter 5 (page 101) which proves an upper bound of $O(n^2)$ for the $(1+1)$ IA to optimise the class of worst case instances for EAs.

By Property 3 of Lemma 7 in Chapter 5 (page 94), we know that for any non-optimal solution, either there are a linear number of small jobs on the fuller machine that can be moved to the emptier one, or the number of small jobs on each machine differ from the optimal configuration (i.e., half and half) by a linear factor.

For the case where there are at least a linear number of small jobs that can be moved from the fuller machine to the emptier one, as the Fast $(1+1)$ IA evaluates the first bit-flip with probability $1/e$, the probability of moving each of these linear number of small jobs (which improves the makespan, hence the mutation operator stops) is $\frac{\Omega(n)}{n} \cdot \frac{1}{e}$ until no move is accepted. Since according to Property 1 of Lemma 7 (page 94), there are at most $O(n)$ different makespan values for the problem, the expected number of hypermutation operations with constant improvement probability is at most $O(n)$. For the second case, Theorem 8 in Chapter 5 (page 98) implies that for at least a linear number of bit-flips, the two large jobs are on different machines with probability at least $\Omega(1)$. During this interval of linear length, we will sample an optimal solution where both the large and the small jobs are evenly distributed with overwhelmingly probability according to Lemma 31 in Chapter 5 (page 99). By considering that the Fast $(1+1)$ IA evaluates a solution with probability at least $2\gamma/n$, the optimum is sampled at most $O(n/\gamma)$ times in expectation before it is evaluated. Taking the wasted fitness function evaluations occurring in case of failure into account, i.e., $O(1+\gamma \log n)$ according to Lemma 14, we get an upper bound of $O(\frac{n}{\gamma} + n \log n)$. □

In Theorem 34 in Chapter 5 (page 106), we proved that the $(1+1)$ IA can find arbitrarily good constant approximations for any PARTITION problem instance in $O(n^3)$ expected fitness function evaluations. In this subsection, we show that, although the Fast $(1+1)$ IA has a low probability of evaluating bit strings with approximately half of the bit-flips, it still finds the

same approximation as that of the $(1+1)$ IA in an expected runtime which is smaller by a linear factor compared to the upper bound of static HMP. The reason is that the leading term in the runtime is due to the exploitation phases where new local optima have to be identified. In such phases, the Fast $(1+1)$ IA is a linear factor faster than static HMPs with FCM.

Before proving the approximation result, we state the following helper lemma which was originally introduced in Chapter 5 for the $(1+1)$ IA and the $(1+1)$ EA (see Lemma 6, page 91) stating the required expected number of iterations to find the next local optimum starting from a non-locally optimal point for any PARTITION instance. Similarly to Chapter 5, by local optimum we mean a solution with makespan that cannot be improved by moving one single job. Moreover, $\ell_1, \ell_2, \cdots \ell_L$ denote local optima of a PARTITION instance where L is the number of local optima and for any $i \in L, f(\ell_i) \geq f(\ell_{i+1})$.

The proof of the following lemma is identical to that of Property 1 of Lemma 6 (page 91) except that here, the probability of moving a specific job to the emptier machine in the first mutation step is $1/(en)$ instead of $1/n$. Therefore, the proof is omitted.

Lemma 15. *Let $x \in \{0,1\}^n$ be a non-locally optimal solution to the partition instance such that $f(\ell_i) > f(x) \geq f(\ell_{i+1})$ for some $i \in [L-1]$. Then, the Fast $(1+1)$ IA using FCM_p with current solution x samples a solution y such that $f(y) \leq f(\ell_{i+1})$ in at most $2en^2$ expected generations.*

The following theorem shows that the Fast $(1+1)$ IA can efficiently find arbitrarily good constant approximations to any PARTITION instance. Similarly to before, by large jobs we mean $s := \lceil 2/\varepsilon \rceil - 1 \leq n/2$ largest jobs. Clearly, by choosing a smaller ε (i.e., better approximation ratios), the number of jobs labelled as *large* (i.e., s) will increase and the expected time to find such approximation will be exponential in $1/\varepsilon$.

Theorem 53. *The Fast $(1+1)$ IA using static HMP with FCM_p finds a $(1+\varepsilon)$ approximation for any* PARTITION *instance in $O(2en^2 \cdot (2^{2/\varepsilon}+1) \cdot (1+\gamma \log n) + \frac{1}{2\gamma} ne^3 \varepsilon^{-(\frac{2}{\varepsilon}+1)} (1-\varepsilon)^{-2} (1+\gamma \log n))$ expected fitness function evaluations for any $\varepsilon = \omega(1/\sqrt{n})$.*

Proof. The proof idea is similar to that of Theorem 34 in Chapter 5 for the $(1+1)$ IA (page 106).

We denote the current solution with X_t and assume the algorithm stops as soon as it finds a $(1+\varepsilon)$ approximation. By Lemma 15, we know that any non-locally optimal point would find a point such that its makespan is at least as good as the next local optimum in at most $2en^2$ generations in expectation. As the number of local optima with different fitness which are not $(1+\varepsilon)$ approximations is at most $2^{2/\varepsilon}$ according to Property 4 of Lemma 10 in Chapter 5 (page 105), before finding a $(1+\varepsilon)$ approximation the expected number of generations where $X_t \notin \mathscr{L}$ is $2en^2 \cdot (2^{2/\varepsilon}+1)$. If $\sum_{i=s+1}^{n} p_i \geq \frac{1}{2} \sum_{i=1}^{n} p_i$, then any of these

local optima would be $(1+\varepsilon)$ and the algorithm would stop. Therefore, for the rest of this proof, we assume $\sum_{i=s+1}^{n} p_i < \frac{1}{2}\sum_{i=1}^{n} p_i$.

Now, we want to find the expected number of generations where $X_t \in \mathscr{L}$ before finding a $(1+\varepsilon)$ approximation. Hence, for the rest of the proof, we only assume locally optimal solutions which are not $(1+\varepsilon)$ approximations and we show the expected number of generations before such solutions would eventually find a $(1+\varepsilon)$ approximation. To do this, we use a specific configuration of large jobs for which we know that, when reached, the small jobs would be also distributed in a way that a $(1+\varepsilon)$ approximation would be achieved. This configuration is the optimal configuration of only large jobs assuming there is no small job. We call such configuration H and denote its makespan with h. While h might be different than the makespan of the large jobs in the optimal makespan f_{opt}, we know that $h \leq f_{opt}$. W.l.o.g, we assume both the fuller machines of X_t and H are M_1.

By Lemma 8 in Chapter 5 (page 98), the large jobs would have the same configuration as in H between $n(\varepsilon-\varepsilon^2)$th and $n-n(\varepsilon-\varepsilon^2)$th bit-flips with probability at least $\left(\frac{n(\varepsilon-\varepsilon^2)-(2/\varepsilon)+1}{n-(2/\varepsilon)+1}\right)^{2/\varepsilon} \geq \frac{(\varepsilon-\varepsilon^2)^{2/\varepsilon}}{e}$. By Property 4 of Lemma 10 (page 105), if there is any small job on the fuller machine, then this solution is already a $(1+\varepsilon)$ approximation. So we assume there is no small job on M_1. Since $\sum_{i=s+1}^{n} p_i < \frac{1}{2}\sum_{i=1}^{n} p_i = W/2$, using Lemma 11 in Chapter 5 (page 106), by $n(\varepsilon-\varepsilon^2)$th bit-flip, the expected total processing time of small jobs moved from M_2 to M_1 is at most $(\varepsilon-\varepsilon^2)\cdot W/2$. Using Markov's inequality (see Theorem 6, page 35), the probability of moving at most $\varepsilon W/2$ jobs to M_1 is at least $1-(\frac{(\varepsilon-\varepsilon^2)W}{\varepsilon W}) = \varepsilon$. This is the probability that the makespan is at most $h+\varepsilon W/2$. As the optimal makespan, f_{opt}, is larger than h and $f_{opt} \geq W/2$, we get $\frac{h+\varepsilon W/2}{f_{opt}} \leq (1+\varepsilon)$.

Overall, with probability:

$$p \geq \frac{(\varepsilon-\varepsilon^2)^{2/\varepsilon}}{e}\cdot\frac{\gamma}{n/2}\cdot\varepsilon = \frac{2\gamma\varepsilon^{1+(2/\varepsilon)}(1-\varepsilon)^2}{ne^3},$$

a $(1+\varepsilon)$ approximation is found unless an improvement is achieved before (i.e., the first term of the runtime in the theorem statement). Considering that at most $(1+\gamma\log n)$ evaluations are performed during each iteration, the approximation is found in at most $O(2en^2\cdot(2^{2/\varepsilon}+1)\cdot(1+\gamma\log n)+\frac{1}{2\gamma}ne^3\varepsilon^{-(\frac{2}{\varepsilon}+1)}(1-\varepsilon)^{-2}(1+\gamma\log n))$ expected fitness function evaluations. □

For $\gamma = 1/\log n$, and some constant ε, the runtime is dominated by $2en^2 2^{2/\varepsilon}$. For such a setting, the Fast $(1+1)$ IA is by a linear factor faster than the $(1+1)$ IA for finding a $(1+\varepsilon)$ approximation for any PARTITION instance.

7.6.2 Performance Analysis of Fast Hypermutations for VERTEX COVER

In this section, we consider the classical NP-hard VERTEX COVER problem to analyse the performance of the Fast $(1+1)$ IA. In VERTEX COVER, the input is a graph and the problem is to find a minimal set of vertices such that all edges of the graph are covered by at least one vertex in that set.

Definition 8. *Given an undirected graph $G = (V,E)$, the* VERTEX COVER *problem asks to find a minimal subset of vertices, $V' \subseteq V$, such that every edge $e \in E$ is adjacent to one of the vertices in V'.*

To use AISs (or EAs) to optimise VERTEX COVER instances, each vertex of the graph is denoted by a bit in the bit string; If $x_i = 1$, then it means that vertex i is included in the cover set. We use the fitness function used in [56, 71] (for minimisation) to evaluate the fitness of candidate solutions:

$$f(x) = \sum_{i=1}^{n} \left(x_i + n(1-x_i) \sum_{j=1}^{n} (1-x_j) e_{i,j} \right),$$

where $e_{i,j}$ is an edge connection between vertex i and vertex j and is one if there is an edge between these vertices in the graph G. This fitness function sums the number of vertices in the cover (the first term) and gives a large penalty to the number of uncovered edges (the second term)[2].

In the following theorems, we analyse both the $(1+1)$ IA with FCM and the Fast $(1+1)$ IA with FCM_p for finding a cover for any given graph. We will prove that the Fast $(1+1)$ IA is by a linear factor faster than the $(1+1)$ IA.

Theorem 54. *The expected time until the $(1+1)$ IA using static HMP with FCM finds a feasible solution to vertex cover is $\Theta(n^2 \log n)$.*

Proof. To prove the upper bound, we use the same proof idea as Theorem 1 in [34] which provides an upper bound on the expected runtime of the $(1+1)$ EA and RLS for finding a feasible solution to vertex cover.

Let k denote the number of vertices that are incident to at least one uncovered edge and u be the total number of uncovered edges. The optimisation goal is to find the expected time until the number of uncovered edges is $u = 0$.

Looking at only the first bit-flip, the probability of improvement is at least k/n and each accepted offspring decreases the number of uncovered edges by u/k in expectation. The

[2]Note that edge based presentations for VERTEX COVER have also been considered in the literature [48].

reason is that, on average, each of the k vertices are connected to u/k uncovered edges in expectation, hence, at the end of each improving step, the expected number of uncovered edges is at most $u_{t+1} := (u_t - \frac{k}{nk}u_t) = (u_t - \frac{1}{n}u_t) = u_t(1-\frac{1}{n})$, hence the drift (i.e., the expected number of uncovered edges) is $\delta := 1/n$. Now, we can use multiplicative drift analysis (see Theorem 10 in Chapter 3, page 37) to compute the expected time until all the edges are covered (i.e., $u = 0$). Assuming g to be the number of uncovered edges, g_{max} is $n(n-1)/2$ for a complete graph. Hence, we get an expected runtime of $E(T) \leq \frac{1}{\delta}(1+\ln(n\cdot(n-1)) \leq O(n\log n)$ to cover all edges. By pessimistically assuming that in the case of a failure at improving in the first mutation step, n fitness function evaluations are wasted, the overall expected runtime is $O(n^2\log n)$.

To prove a lower bound on the expected time to find a cover, we assume that the given graph is complete (i.e., the number of edges is $n\cdot(n-1)/2$). The size of a cover for such a graph is $n-1$. By Chernoff bounds (see Theorem 8, page 36), we know that w.o.p. the initialised solution (cover) includes at most $2n/3$ of vertices (i.e., the number of 1-bits). To compute the expected time until all $n-1$ vertices are selected, we use the Ballot theorem. Considering the number of 0-bits as $i = q$ and the number of 1-bits as $n-i = p$, the probability of an improvement is at most $1-(p-q)/(p+q) = 1-(n-2i)/n = 2i/n$ by the Ballot theorem (see Theorem 11, page 38). Since the operator stops at the first constructive mutation, it is necessary that at least $n/3$ improving hypermutation occur. Hence, the expected runtime for the cover to be identified is at least $\sum_{i=1}^{n/3}\left(\frac{n}{2i}\cdot 1 + (\frac{n}{2i}-1)\cdot n\right) = \Omega(n^2\log n)$.

□

Theorem 55. *The expected time until the Fast* $(1+1)$ *IA using static HMP with* FCM_p *finds a feasible solution to vertex cover is* $\Theta(n\log n\cdot(1+\gamma\log n))$.

Proof. The proof for both upper bound and lower bound is similar to that of Theorem 54. We start with the upper bound first. For the Fast $(1+1)$ IA, the probability of improvement in the first bit-flip is $k/(ne)$ which does not change the runtime asymptotically. However, in case of failure at improving the fitness, the Fast $(1+1)$ IA wastes at most $O(1+\gamma\log n)$ fitness function evaluations in expectation. This yields an expected runtime of $E(T) = O((1+\gamma\log n)\cdot n\ln n)$. Note that this time is $O(n\ln n)$ for the choice of $\gamma = 1/\log n$.

Regarding the lower bound, the proof is identical to that of Theorem 54 except that the expected wastage when failing to find an improvement is $\Omega(1+\gamma\log n)$, which makes the expected runtime larger than $\sum_{i=1}^{n/3}(\frac{n}{2i}\cdot 1 + (\frac{n}{2i}-1)\cdot(1+\gamma\log n)) = \Omega(n\log n\cdot(1+\gamma\log n))$.

□

7.7 Conclusion

We have presented two alternatives for the FCM mechanism, FCM_p and BFM_p, and have rigorously proved for several significant benchmark problems from the literature that they maintain the exploration characteristics of the traditional operator while outperforming them up to linear factor speed-ups in the exploitation phase.

The main modification that allows the algorithm to achieve the presented improvements is to sample the solution after the ith bit-flip stochastically with probability roughly $p_i = \gamma/i$, rather than deterministically with probability one. The analyses show that the parameter γ can be set easily. Concerning FCM_p, that returns the first sampled constructive mutation and is suggested to be used in isolation, any $\gamma = O(1/\log(n))$ allows optimal asymptomatic exploitation time (based on the unary unbiased black box complexity of ONEMAX and LEADINGONES) while maintaining the traditional exploration capabilities. Concerning BFM_p, which does not stop at first constructive mutation and is designed to work harmonically with ageing as in the standard Opt-IA, considerable lower values of the parameter (i.e., $\gamma = 1/(n\log^2 n)$) are required to escape from difficult local optima efficiently (e.g., CLIFF) such that the hypermutations do not return to the local optima with high probability. While these low values for γ still allow optimal asymptotic exploitation in the unbiased unary black box sense, they considerably reduce the capability of the operator to perform the large jumps required to escape the local optima of functions with characteristics similar to JUMP, i.e., where ageing is ineffective due to the second slope of decreasing fitness. Additionally, Fast AIS still suffer on $\text{HYPERTRAP}_{c/8}$ introduced in Chapter 4 (see Definition 3, page 76).

We have also shown that linear speed-ups are achieved by the fast hypermutation operators for the PARTITION and VERTEX COVER combinatorial optimisation problems that have many real world applications [81]. Future work should consider an adaptation of the parameter γ to allow it to automatically increase and decrease throughout the run [30, 27].

Part IV

Conclusion

Chapter 8

Summary and Final Remarks

Various general-purpose optimisation algorithms have been designed with the aim of identifying high quality solutions by imitating the principles of the adaptive immune system of vertebrates [24]. While several successful supplications to complex optimisation problems have been reported for different clonal selection inspired AISs [20, 17, 18], it is still largely unclear when AISs will have better performance than more traditional search heuristics, such as stochastic local search and evolutionary algorithms (EAs), hence when they should be preferred. This thesis has shed considerable amount of light on this question.

As the theoretical analyses of some well-known AISs, e.g., Clonalg [23] and B-cell [55], have proved some serious drawbacks regarding their general applicability [93, 50], the main focus of this thesis was Opt-IA [19, 20] which was the least investigated among the well-known AISs, yet seemed the most promising to us. Opt-IA applies three different immune system operators, i.e., cloning, hypermutations with mutation potential, and ageing, which make it probably the closest existing computational model to the natural immune system designed for optimisation problems. In this thesis, we investigated the performance of Opt-IA operators to identify their advantages and disadvantages particularly in comparison to those used by traditional EAs.

We started by analysing hypermutation with static mutation potential (static HMP) in isolation for optimising standard benchmark problems to highlight its behaviour and performance in different settings. The first important finding was the necessity of using the HMP operator in conjunction with the FCM (stopping at first constructive mutation) mechanism. We proved that if FCM is not used, then AISs applying hypermutations need at least expected exponential time to optimise any reasonable function. Concerning exploitation capabilities, we proved that static HMPs are always at most by a linear factor slower than RLS for any function that can be analysed with the artificial fitness levels method. This result was proven with the introduction of a methodology that allows us to derive upper bounds on

the runtime of complicated HMP operators by analysing the simpler RLS algorithm. The derived bounds are a linear factor larger than those of RLS. Proving that these bounds are tight for some easy hill-climbing benchmark functions, we concluded that EAs and RLS are better options than static HMP for easy unimodal functions. After showing HMPs are not as efficient as EAs for exploitation, we then rigorously proved their significant superiority at escaping local optima of standard multimodal benchmark functions. We also proved that for functions where very large jumps in the search space are needed, HMPs outperform SBM making a difference between polynomial and exponential runtimes.

By analysing the (hybrid) ageing operator in isolation (i.e., without any other immune inspired operator), we showed how it allows EAs and RLS to have an impressive performance for escaping difficult local optima of standard benchmark problems. Compared to what was known about (hybrid) ageing's efficiency previously (i.e., that it is efficient on specific examples of a complex search space from dynamic optimisation and only when combined with RLS [74]), our analyses revealed that ageing can be very effective for general classes of functions and when applied with both RLS and SBM. In particular, we proved that ageing coupled with RLS can optimise difficult CLIFF functions in $O(n \log n)$ time in expectation which is the smallest runtime achievable by RLS and SBM for any function with unique optimum [57]. However, we surprisingly discovered the opposite effect when ageing is coupled with HMP, i.e., in the original framework they are supposed to work together. The reason behind such failure is that, while ageing helps the algorithm to escape from local optima by accepting individuals of lower fitness, HMP only stops at a first constructive mutation (i.e., it cannot identify solutions of lower quality).

After analysing both operators in isolation, we then presented the first ever analysis of the complete population-based Opt-IA. We highlighted problem characteristics for which the combination of ageing and HMP is either crucial or detrimental by presenting example landscapes. If a given landscape includes a local optimum which, once discovered, requires a large jump to find new parts of the search space of lower fitness leading to the global optimum, then the use of Opt-IA is crucial (i.e., both ageing and hypermutations are needed to optimise such landscape). However, a landscape with a large area of local optima (with many improving points) far away from the global optimum, can trap Opt-IA easily after being discovered by large jumps. While FCM makes the algorithm stop after each improvement inside the local optima area, ageing cannot be helpful as after the algorithm escapes, the probability of HMP sampling the local optima again is very high. On the other hand, EAs and RLS rarely find such local optima due to the their very low probability of making big jumps.

After investigating the Opt-IA operators both in isolation and in combination for standard benchmark functions, we showed that the conclusions drawn from our analysis for such functions also hold for a classical combinatorial problem with ties to real world applications. Considering the ability of hypermutations to make large jumps, we looked at the NP-hard PARTITION problem. Previously, the performance of EAs and RLS for finding a $(1+\varepsilon)$ approximation for any PARTITION instance was analysed by Witt [90]. He showed that SBM and RLS can achieve arbitrarily good constant approximation in expected polynomial time. However, his results are only obtainable when a problem specific restart scheme is adapted, or alternatively, a problem specific number of parallel runs are used. Our analyses of AISs using HMP revealed that these algorithms are able to find a $(1+\varepsilon)$ approximation in an expected polynomial time, which is slightly larger than the expected time proved for EAs. However, the approximation is achieved efficiently in every single run rather than requiring problem-specific restarts. Our analysis of the ageing operator in isolation also proved that it guarantees $(1+\varepsilon)$ approximations in polynomial expected time by automatically restarting the search. The two results are achieved in different ways. The ageing operator locates more promising basins of attraction by restarting the optimisation process after implicitly detecting that it has found a local optimum. Hypermutations find improved approximate solutions efficiently by performing large jumps in the search space. These results were the first time performance guarantees for any AIS derived for a problem from combinatorial optimisation. In contrast with EAs, the AISs do not need problem specific information to reach the desired approximation ratio, i.e., neither the knowledge that the tackled problem is PARTITION nor a decision on the desired approximation ratio to be made in advance. If knowledge that the tackled problem is PARTITION was available, then using the $(1+1)$ EA would not be ideal in the first place.

The described analyses for both benchmark problems and classic combinatorial optimisation problems revealed two significant drawbacks of the considered AISs. First, the exploration effectiveness of HMPs comes at the expense of a considerable slow-down in the exploitation phase. Second, the impressive capabilities of ageing coupled with local mutations at escaping local optima disappear when the operator is coupled with HMP. To target these issues, we then introduced alternative ways for AIS to perform hypermutations.

One reasonable way to speed up the exploitation of HMPs is to use the idea of dynamically decreasing the mutation potential as solutions get closer to the global optimum. The obvious issue of such scheme is that of setting a suitable measure to decide whether the algorithm is close to the global optimum. This idea was originally used by both Clonalg and Opt-IA. While the analysis of Clonalg showed that it needs expected exponential time to optimise the easy ONEMAX function [93], the inversely proportional HMP of Opt-IA was proved to have

an expected runtime which is a linear factor slower than that of EAs for easy hill-climbing functions [50], i.e., the asymptotic result on easy functions matches those of static HMP which we proved in this thesis. These results motivated us to develop effective strategies to define inversely proportional HMPs that provably decrease the mutation rate as the optimum is approached. The analyses revealed that the proposed HMP operator based on Hamming distance and exponential decay, successfully obtains asymptotic speed-ups in exploitation compared to static HMP. Using the proposed operator in combination with ageing, the more the search space is explored, the better the algorithm approximates the ideal behaviour of the inversely proportional hypermutations (i.e., that the mutation rate decreases inversely to the distance to the global optimum). However, using such inversely proportional mutation potentials puts some limits on hypermutations: the low mutation rates on local optima mean that they will be able to efficiently escape local optima only when applied together with ageing.

As using the idea of inversely proportional HMPs introduces new issues, we suggested alternative strategies for FCM to reduce the number of fitness evaluations wasted. An AIS using static HMP with the alternative schemes, which is called Fast AIS, samples the solution after the ith bit-flip with a probability approximately γ/i rather than deterministically evaluating after each bit flip. Hence, the algorithm still hypermutates but wastes fewer fitness function evaluations. We proved that such a scheme can speed up the expected runtime of the traditional static HMP by a linear factor at hill-climbing while it still escapes local optima more efficiently than traditional EAs. In addition to some standard multimodal benchmark problems, we proved the latter by analysing the Fast AIS performance in obtaining a $(1+\varepsilon)$ approximation for any PARTITION instance and showing that it achieves such approximation ratios in an expected time which is by a linear factor smaller than the upper bound we provided for static HMP using FCM. However, the suggested Fast AIS was not efficient at escaping local optima while used alongside ageing, and hence suffered from the same issue as the original static HMP using FCM. The Fast AIS can be made to work alongside ageing efficiently to escape local optima by not stopping at the first constructive mutation and instead using a much smaller probability of evaluation (to make it unable to discover the local optima after escaping them via ageing). While such a strategy successfully achieves its goal, we found that it is not efficient on its own for escaping local optima which need large jumps. In other words, it is only effective for landscapes similar to the CLIFF function where there is a second slope with increasing fitness leading away from the local optima.

The ever growing successful uses of elitism and high mutation rates in EAs and steady-state GAs [29, 72, 8, 21, 28], neither of which occur in the natural (Darwinian) evolution,

are evidence of how the immune system paradigm may be more suited for general purpose optimisation than Darwinian evolution.

8.1 Future Work

In this thesis, we have concentrated mainly on the effectiveness of AIS operators for single-individual algorithms (i.e., we have not analysed the effects of b-cell interactions in populations of antibodies in the natural immune system). Indeed only recently have some advantages of populations over single-individuals been shown concerning the more traditional evolutionary and genetic algorithms [8, 9, 21]. Exploiting antibody interactions in the natural immune system in population-based AISs is a promising research direction where natural features of an AIS such as elitism, high mutation rates, and memory cells may prove to be very effective at maintaining a diverse population and hence to have a significant impact on the general-purpose multimodal optimisation typically present in real world applications.

Another potential future work is comparing the performance of hypermutations with heavy-tailed mutation operators [29, 36] that are gaining momentum in the evolutionary computation field. These mutation operators have been recently exploited in the design of so-called fast EAs to allow a larger number of bit flips more often than the standard bit mutations (SBM) traditionally used in EAs and GAs. Since these analogies are very similar to hypermutations in AIS, the comparison between them will provide insights to decide which operator to use for a particular search space. This can also potentially lead to the design of more powerful mutation operators.

The next promising future work is to identify classical combinatorial optimisation problems where AISs provably outperform EAs and other randomised search heuristics. In this thesis, we showed the efficiency of AISs for classical combinatorial problems where EAs are also proven to work efficiently. Additionally, assessing the performance of the proposed modifications to AISs (in Chapters 6 and 7) experimentally on both classical combinatorial problems and real-world applications and comparing their performance with the traditional AISs would potentially give more insights into the effectiveness of the proposed modifications. An example problem can be the protein folding test-beds used in [20] where the original Opt-IA was introduced.

Regarding the proposed modification to the hypermutation with mutation potential operator, some improvements are needed to make the operator adaptive. As we saw in Chapter 7, in the proposed fast hypermutation operators, different γ values are needed to escape local optima depending on the use of ageing operators. Hence, automatically adapting this parameter is a natural future work.

References

[1] Auger, A. and Doerr, B. (2011). *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc.

[2] Bruno, J., Coffman, E. G., and Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing-time. *Communications of the ACM*, 17:382–387.

[3] Burnet, F. M. (1959). *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press.

[4] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education.

[5] Corus, D., Dang, D.-C., Eremeev, A. V., and Lehre, P. K. (2017a). Level-based analysis of genetic algorithms and other search processes. *IEEE Transactions on Evolutionary Computation*, 22(5):707–719.

[6] Corus, D., He, J., Jansen, T., Oliveto, P. S., Sudholt, D., and Zarges, C. (2015). On easiest functions for somatic contiguous hypermutations and standard bit mutations. In *Proc. of GECCO 2015*, pages 1399–1406.

[7] Corus, D., He, J., Jansen, T., Oliveto, P. S., Sudholt, D., and Zarges, C. (2017b). On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica*, 78:714–740.

[8] Corus, D. and Oliveto, P. S. (2017). Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(5):720–732.

[9] Corus, D. and Oliveto, P. S. (2019). On the benefits of populations on the exploitation speed of standard steady-state genetic algorithms. In *Proc. of GECCO 2019*, pages 1452–1460.

[10] Corus, D., Oliveto, P. S., and Yazdani, D. (2017c). On the runtime analysis of the Opt-IA artificial immune system. In *Proc. of GECCO 2017*, pages 83–90.

[11] Corus, D., Oliveto, P. S., and Yazdani, D. (2018a). Artificial immune systems can find arbitrarily good approximations for the NP-hard partition problem. In *Proc. of PPSN 2018*, pages 16–28.

[12] Corus, D., Oliveto, P. S., and Yazdani, D. (2018b). Fast artificial immune systems. In *Proc. of PPSN 2018*, pages 67–78.

[13] Corus, D., Oliveto, P. S., and Yazdani, D. (2019a). Artificial immune systems can find arbitrarily good approximations for the NP-hard number partitioning problem. *Artificial Intelligence*, 247:180–196.

[14] Corus, D., Oliveto, P. S., and Yazdani, D. (2019b). On inversely proportional hypermutations with mutation potential. In *Proc. of GECCO 2019*, page 215–223.

[15] Corus, D., Oliveto, P. S., and Yazdani, D. (2019c). When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theoretical Computer Science (In press)*.

[16] Covantes Osuna, E. (2019). *Theoretical and Empirical Evaluation of Diversity-preserving Mechanisms in Evolutionary Algorithms: On the Rigorous Runtime Analysis of Diversity-preserving Mechanisms in Evolutionary Algorithms*. PhD thesis, University of Sheffield.

[17] Cutello, V. and Nicosia, G. (2006). A clonal selection algorithm for coloring, hitting set and satisfiability problems. *Neural Nets*, 3931:324–337.

[18] Cutello, V., Nicosia, G., and Pavone, M. (2003). A hybrid immune algorithm with information gain for the graph coloring problem. In *Proc. of GECCO 2003*, pages 171–182.

[19] Cutello, V., Nicosia, G., and Pavone, M. (2004). Exploring the capability of immune algorithms: a characterization of hypermutation operators. In *Proc. of ICARIS 2004*, pages 263–276.

[20] Cutello, V., Nicosia, G., Pavone, M., and Timmis, J. (2007). An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation*, 11:101–117.

[21] Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2018). Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22(3):484–497.

[22] de Castro, L. N. and Timmis, J. (2002a). Artificial immune systems: a novel approach to pattern recognition. In *Artificial Neural Networks in Pattern Recognition*, pages 67–84.

[23] de Castro, L. N. and Zuben, F. J. V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.

[24] de Castro, L. R. and Timmis, J. (2002b). *Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag.

[25] Doerr, B. (2011). Analyzing randomized search heuristics: tools from probability theory. In Auger, A. and Doerr, B., editors, *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, chapter 1, pages 1–20. World Scientific.

[26] Doerr, B. (2019). Probabilistic tools for the analysis of randomized optimization heuristics. In Doerr, B. and Neumann, F., editors, *Theory of Randomized Search Heuristics in Discrete Search Spaces*, chapter 1, pages 1–87. Springer.

[27] Doerr, B. and Doerr, C. (2018). Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709.

[28] Doerr, B., Doerr, C., and Ebel, F. (2015). From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104.

[29] Doerr, B., Le, H. P., Makhmara, R., and Nguyen, T. D. (2017). Fast genetic algorithms. In *Proc. of GECCO 2017*, pages 777–784.

[30] Doerr, B., Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2018). On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proc. of GECCO 2018*, page 1015–1022.

[31] Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the $(1+1)$ evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81.

[32] Feller, W. (1968). *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons.

[33] Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R. (1994). Self-nonself discrimination in a computer. In *IEEE Symposium on Security and Privacy*, pages 202–212.

[34] Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., and Witt, C. (2010). Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633.

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

[36] Friedrich, T., Quinzan, F., and Wagner, M. (2018). Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In *Proc. of GECCO 2018*, pages 293–300.

[37] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[38] Graham, R. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:263–269.

[39] Hayes, B. (2002). The easiest hard problem. *American Scientist*, 90:113–117.

[40] Hedberg, S. (1996). Combating computer viruses: IBM's new computer immune system. *IEEE Parallel and Distributed Technology: Systems and Applications*, 4(2):9–11.

[41] Hochbaum, D. S. and Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162.

[42] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press.

[43] Horoba, C., Jansen, T., and Zarges, C. (2009). Maximal age in randomized search heuristics with aging. In *Proc. of GECCO 2009*, pages 803–810.

[44] Jägersküpper, J. and Storch, T. (2007). When the plus strategy outperforms the comma strategy and when not. In *Proc. of FOCI 2007*, pages 25–32.

[45] Janeway, C. (2011). *Janeway's Immunobiology*. Garland Science, 8th edition.

[46] Jansen, T. (2013). *Analyzing Evolutionary Algorithms: the Computer Science Perspective*. Springer.

[47] Jansen, T., Oliveto, P. S., and Zarges, C. (2011). On the analysis of the immune-inspired B-Cell algorithm for the vertex cover problem. In *Proc. of ICARIS 2011*, pages 117–131.

[48] Jansen, T., Oliveto, P. S., and Zarges, C. (2013). Approximating vertex cover using edge-based representations. In *Proc. of FOGA 2013*, pages 87–96.

[49] Jansen, T. and Zarges, C. (2010). Aging beyond restarts. In *Proc. of GECCO 2010*, pages 705–712.

[50] Jansen, T. and Zarges, C. (2011a). Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theoretical Computer Science*, 412(6):517–533.

[51] Jansen, T. and Zarges, C. (2011b). On benefits and drawbacks of aging strategies for randomized search heuristics. *Theoretical Computer Science*, 412(6):543–559.

[52] Jansen, T. and Zarges, C. (2011c). Variation in artificial immune systems: hypermutations with mutation potential. In *Proc. of ICARIS 2011*, pages 132–145.

[53] Jansen, T. and Zarges, C. (2012). Computing longest common subsequences with the B-Cell algorithm. In *Proc. of ICARIS 2012*, pages 111–124.

[54] Karp, R. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103.

[55] Kelsey, J. and Timmis, J. (2003). Immune inspired somatic contiguous hypermutation for function optimisation. In *Proc. of GECCO 2003*, pages 207–218.

[56] Khuri, S. and Bäck, T. (1994). An evolutionary heuristic for the minimum vertex cover problem. In *KI-94 Workshops (Extended abstracts)*, pages 86–90.

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

[58] Lengler, J. (2019). Drift analysis. In Doerr, B. and Neumann, F., editors, *Theory of Randomized Search Heuristics in Discrete Search Spaces*, chapter 2, pages 89–126. Springer.

[59] Li, Z., Woo, C. J., Iglesias-Ussel, M. D., Ronai, D., and Scharff, M. D. (2004). The generation of antibody diversity through somatic hypermutation and class switch recombination. *Genes and Development*, 18:1–11.

[60] Lissovoi, A., Oliveto, P. S., and Warwicker, J. A. (2019). On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *Proc. of AAAI 2019*, pages 2322–2329.

[61] Mayforth, R. (1993). *Designing Antibodies*. Academic Press.

[62] Merkle, R. C. and Hellman, M. E. (1978). Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530.

[63] Mertens, S. (2006). The easiest hard problem: number partitioning. *Computational Complexity and Statistical Physics*, 125(2):125–139.

[64] Mitzenmacher, M. and Upfal, E. (2005). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

[65] Mucha, M., Wundefinedgrzycki, K., and Włodarczyk, M. (2019). A subquadratic approximation scheme for partition. In *Proc. of SODA 2019*, page 70–88.

[66] Neumann, F., Oliveto, P. S., and Witt, C. (2009). Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *Proc. of GECCO 2009*, pages 835–842.

[67] Neumann, F., Sudholt, D., and Witt, C. (2008). Rigorous analyses for the combination of ant colony optimization and local search. In *Proc. of International Conference on Ant Colony Optimization and Swarm Intelligence 2008*, volume 5217, pages 132–143.

[68] Neumann, F. and Witt, C. (2010). *Bioinspired Computation in Combinatorial Optimization*. Springer-Verlag.

[69] Neumann, F. and Witt, C. (2015). On the runtime of randomized local search and simple evolutionary algorithms for dynamic makespan scheduling. In *Proc. of IJCAI 2015*, pages 3742–3748.

[70] Oliveto, P. S., He, J., and Yao, X. (2007). Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. *Automation and Computing*, pages 100–106.

[71] Oliveto, P. S., He, J., and Yao, X. (2009a). Analysis of the $(1+1)$ EA for finding approximate solutions to vertex cover problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1006–1029.

[72] Oliveto, P. S., Lehre, P. K., and Neumann, F. (2009b). Theoretical analysis of rank-based mutation-combining exploration and exploitation. In *Proc. of CEC 2009*, pages 1455–1462.

[73] Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2018). How to escape local optima in black box optimisation: when non-elitism outperforms elitism. *Algorithmica*, 80:1604—-1633.

[74] Oliveto, P. S. and Sudholt, D. (2014). On the runtime analysis of stochastic ageing mechanisms. In *Proc. of GECCO 2014*, pages 113–120.

[75] Oliveto, P. S., Sudholt, D., and Zarges, C. (2019). On the benefits and risks of using fitness sharing for multimodal optimisation. *Theoretical Computer Science*, 773:53–70.

[76] Oliveto, P. S. and Witt, C. (2014). On the runtime analysis of the simple genetic algorithm. *Theoretical Computer Science*, 545:2–19.

[77] Oliveto, P. S. and Witt, C. (2015). Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science*, 605:21–41.

[78] Oliveto, P. S. and Yao, X. (2011). Runtime analysis of evolutionary algorithms for discrete optimisation. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, chapter 2, pages 21–52. World Scientific.

[79] Osuna, E. C. and Sudholt, D. (2017). Analysis of the clearing diversity-preserving mechanism. In *Proc. of FOGA 2017*, pages 55–63.

[80] Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenova, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. In *Proc. of GECCO 2015*, pages 1455–1462.

[81] Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, 5th edition.

[82] Rogozin, I. B. and Diaz, M. (2004). Cutting edge: Dgyw/wrch is a better predictor of mutability at g:c bases in ig hypermutation than the widely accepted rgyw/wrcy motif and probably reflects a two-step activation-induced cytidine deaminase-triggered process. *Journal of Immunology*, 172:3382–3384.

[83] Rogozin, I. B., Pavlov, Y. I., Bebenek, K., Matsuda, T., and Kunkel, T. A. (2001). Somatic mutation hotspots correlate with DNA polymerase eta error spectrum. *Nature Immunology*, 2:530–536.

[84] Rudolph, G. (1997). *Convergence properties of evolutionary algorithms*. PhD thesis, Hamburg.

[85] Sahni, S. K. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127.

[86] Serfling, R. J. (1974). Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48.

[87] Sudholt, D. (2012). A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435.

[88] Sudholt, D. (2019). The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. In Doerr, B. and Neumann, F., editors, *Theory of Randomized Search Heuristics in Discrete Search Spaces*, chapter 8, pages 359–400. Spinger.

[89] Whitley, D. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proc. of the Third International Conference on Genetic Algorithms*, pages 116–121.

[90] Witt, C. (2005). Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS 2005*, pages 44–56.

[91] Witt, C. (2006). Runtime Analysis of the $(\mu + 1)$ EA on Simple Pseudo-Boolean Functions. *Evolutionary Computation*, 14(1):65–86.

[92] Xia, X. and Zhou, Y. (2018). On the effectiveness of immune inspired mutation operators in some discrete optimization problems. *Information Science*, 426(C):87–100.

[93] Zarges, C. (2008). Rigorous runtime analysis of inversely fitness proportional mutation rates. In *Proc. of PPSN*, pages 112–122.

[94] Zarges, C. (2009). On the utility of the population size for inversely fitness proportional mutation rates. In *Proc. of FOGA 2009*, pages 39–46.

[95] Zarges, C. (2019). Theoretical foundations of immune-inspired randomized search heuristics for optimization. In Doerr, B. and Neumann, F., editors, *Theory of Randomized Search Heuristics in Discrete Search Spaces*, chapter 10, pages 443–474. Springer.