Self-adaptive Parameter Control Mechanisms in Evolutionary Computation

By

Xiaoyu Qin

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
August 2023

ABSTRACT

Evolutionary algorithms (EAs) are effective solvers for a variety of discrete black-box optimisation problems. However, their performance heavily relies on the proper selection of algorithmic parameters, such as mutation rate, crossover rate, and selection pressure, which are often problem- and instance-specific. One promising approach is self-adaptive parameter control mechanisms, where the parameters are encoded in the individuals and evolved along with their solutions through variation operators. Although there have been some theoretical studies demonstrating the efficiency of self-adaptive EAs on certain functions with unknown structure, the potential benefits of this approach in other scenarios remain unknown. Moreover, there is a great opportunity for creativity in designing elegant self-adaptive EAs.

In this thesis, we first examine the benefits of self-adaptation in noisy optimisation, where the presence of noise can significantly affect the algorithm performance. We provide a mathematical analysis of the runtime of 2-tournament EAs with self-adapting mutation rates and two other parameter strategies on a theoretical benchmark optimisation problem with and without symmetric noise. Our results demonstrate that self-adaptation consistently achieves the lowest runtime compared to using fixed mutation rates. This is confirmed regardless of the presence of noise, through additional experiments with other noise types and self-adaptation mechanisms. Next, we investigate the performance of self-adaptive EAs on a natural tracking dynamic optima problem, known as the *Dynamic Substring Matching* (DSM) problem. This problem requires the algorithm to track a sequence of structure-changing optima, and our analysis reveals that mutation-based EAs with a fixed mutation

rate have small chance of succeeding, while self-adaptive EAs can track the dynamic optima with high probability. Finally, we propose a novel self-adaptive EA for single-objective optimisation, which incorporates multi-objective optimisation principles to jointly optimise the fitness and mutation rates. We call this algorithm multi-objective self-adaptive EA (MOSA-EA). Our runtime analysis shows that the MOSA-EA can efficiently escape the local optima with unknown sparsity, where fixed mutation rate EAs may become trapped. In an empirical study on complex combinatorial problems, the MOSA-EA outperforms twelve other algorithms on NK-Landscape and Max-k-Sat problem instances. Overall, this thesis suggests that the self-adaptation has great potential for controlling parameters in EAs, especially in challenging optimisation scenarios, such as uncertain optimisation.

ACKNOWLEDGMENTS

I wish to express my profound gratitude to the many individuals who played pivotal roles throughout my PhD journey.

Foremost, my deepest thanks are due to my supervisor, Prof Per Kristian Lehre. His unwavering patience, meticulousness, and steadfast sense of duty have been a beacon of inspiration. The regular consultations we had, particularly during vital stages of the project, combined with his comprehensive guidance on my research, significantly fuelled my achievements. It was under his tutelage that I produced more than nine papers during my PhD studies, a feat I could not have accomplished without him.

I extend heartfelt thanks to the respected members of my Thesis Group: Dr Rajesh Chitnis, Prof Ata Kaban, and Prof Jonathan Rowe. During our six intensive thesis group meetings, their keen insights and invaluable suggestions greatly refined my research and overall PhD journey. Additionally, I am deeply indebted to the research community in the field of Evolutionary Computation. Their spirit of collaboration and their robust support system for emerging researchers like myself were immeasurable. The engaging dialogues and constructive feedback I received profoundly enriched my research perspective.

A special acknowledgment goes to BlueBear, the high-performance computing facility at the University of Birmingham. Their generous provision of over 47 years of CPU time was pivotal for the computational elements of my PhD project. This resource was crucial in propelling my research forward, and I am profoundly thankful for their support. I'd also like to express my genuine gratitude to the committed staff of the School of Computer Science.

Their consistent administrative and technical assistance ensured the seamless progression of my academic endeavours.

I am fortunate to have been encircled by an exceptional group of colleagues and friends. Their camaraderie, insights, and steadfast support were foundational throughout my PhD. While I cannot capture the entirety of their contributions here, special mention goes to the following individuals, alphabetically arranged by surname: Dr Alistair Benford, Mr Xinxing Cheng, Dr Zitai Chen, Dr Mingjie Chen, Mr Haoxuan Ding, Ms Zixuan Han, Ms Shanshan He, Dr Mario Alejandro Hevia Fajardo, Mr Xi Jia, Mr Chaoze Liu, Ms Di Liu, Mr Shishen Lin, Dr Miqing Li, Mr Zimin Liang, Mr Wei Li, Mr Yuhang Qu, Ms Haixi Shan, Mr Hao Tong, Mr Chenguang Xiao, Ms Xinyue Xue, Mr Xiankun Yan, Dr Jing Zhang, Mr Mo Zhang, Mr Tianyang Zhang, Ms Sijia Zhou. To everyone, both mentioned and unmentioned, I offer my heartfelt thanks.

Lastly, and most profoundly, my deepest gratitude goes to my parents, Mr Shuyi Qin and Prof Huan Song, and to my entire family. Their unending love, encouragement, and unwavering faith in me have been the bedrock of my academic pursuits.

Contents

List of Figures

List of Tables

Acronyms

ACO Ant Colony Optimisation.

cGA compact Genetic Algorithm.

EA Evolutionary Algorithm.

EC Evolutionary Computation.

ECAI European Conference on Artificial Intelligence.

EDA Estimation of Distribution Algorithm.

ES Evolutionary Strategy.

FOGA Foundations of Genetic Algorithms.

GA Genetic Algorithm.

GECCO Genetic and Evolutionary Computation Conference.

GP Genetic Programming.

MMAS Max-Min Ant System.

MOSA-EA Multi-objective Self-adaptive EA.

PPSN Parallel Problem Solving from Nature.

RLS Random Local Search.

RS Random Search.

SAEA Self-adaptive EA.

TSP Traveling Salesman Problem.

UMDA Univariate Marginal Distribution Algorithm.

Chapter One

Introduction

1.1 General Introduction

Optimisation is a fundamental task in computer science and operations research. In general, an optimisation problem can be expressed in terms of a relation $\mathcal{P} \subseteq \mathcal{I} \times \mathcal{S}$, where $\mathcal{I}$ represents the set of problem instances, and $\mathcal{S}$ denotes the set of problem solutions. For any instance $z \in \mathcal{I}$, a feasible solution $x \in \mathcal{S}$ must satisfy $(z, x) \in \mathcal{P}$. Given a problem instance $z \in \mathcal{I}$, an optimisation problem is the problem of finding the "best" solution $x^* \in \mathcal{S}$ based on a specific measure among all feasible solutions $y \in \mathcal{S}$ (Ausiello et al., 1999). This measure of the "quality" of the solution is typically referred to as the *objective function* $f_z : \mathcal{S}_z \to \mathbb{R}$, where the set $\mathcal{S}_z := \{y \mid y \in \mathcal{S} \wedge (z, y) \in \mathcal{P}\}$, known as the *search space*, consists of all feasible solutions, referred to as *search points*. The solution x^* of the problem instance z, called the *optimal solution*[*], is the one where $f_z(x^*) \geq f_z(x)$ for all $x \in \mathcal{S}_z$. We have *constrained optimisation* if there exists infeasible solutions. For certain problem instances, identifying feasible solutions may be difficult. This thesis considers unconstrained optimisation. Typically, the search space of an optimisation problem is comprised of a sequence of variables, also referred to as dimensions, with each having its own domain. The size of the problem instance is directly related to the number of these variables. If these variables are continuous (continuous search space), the optimisation problem is classified as a *continuous optimisation problem*; if discrete (discrete search space), it is considered a *discrete* or *combinatorial optimisation problem* (Papadimitriou and Steiglitz, 1998). Both types of optimisation problems are essential in various fields, such as science (Pintér, 2006), engineering (Gen and Cheng, 1999), and economics (Intriligator, 2002).

For numerous optimisation problems, finding an optimal solution efficiently can be challenging, as they are often NP-hard (Fogel, 2000). For example, the optimisation version of the *traveling salesman problem* (TSP) is a well-known NP-hard problem, which aims to find

[*]In this thesis, we assume that the objective is to maximise the objective function.

the path with smallest weight among all paths that visits each city exactly once and returns to the origin, of a weighted graph. To narrow down the research scope, the thesis focus on optimising *pseudo-Boolean functions* where search spaces are binary $\{0,1\}^n$ (Boros and Hammer, 2002), as formally described in Section 2.2. The choice of pseudo-Boolean functions in this thesis is motivated by their role as objective functions in many combinatorial optimisation problems, coupled with the extensive research available on their optimisation. Another reason is that there are almost no previous analysis of self-adaptation, it makes sense to initiate the analysis in the best-studied search domain within the theory of evolutionary algorithms, which is pseudo-Boolean optimisation.

Uncertainty can further complicate the optimisation task, as it refers to the presence of unknown factors that influence objective functions, leading to variations in the mapping from a search point to the objective value during optimisation. There are various types of uncertainties in the context of optimisation, such as *noise* and *dynamic* optimisation (Jin and Branke, 2005). In the case of noise, the objective values of a search point are random variables, while in the case of dynamic optimisation, the objective function changes over time. For instance, in the TSP, noise could preclude obtaining the precise total weight of a path, while dynamics could cause the weights of edges to change as a function of time. Uncertainty models are introduced and discussed in Sections 2.2.5 and 2.3.3, respectively.

Furthermore, objective functions may contain *local optima* which can pose challenges for algorithms. A given local optimum, denoted by $\bar{x}$, is considered "better" than other "nearby" search points, such that $f(\bar{x}) \geq f(x)$ for all $x \in \mathrm{N}_k(\bar{x})$, where $\mathrm{N}_k(\bar{x})$ represents the set of "neighbouring" search point x. Despite this, $\bar{x}$ could still be "worse" than the optimal solution x^*, as $f(\bar{x}) < f(x^*)$. It is important to note that the definition of "neighbours" of a search point may vary among problem instances and algorithms. Further details can be found in Section 2.3.4. The term *landscape* is used to characterise the solutions and their respective fitness values of an objective function (Jones, 1995). When multiple local optima are present,

the objective function is deemed a *multi-modal landscape*. Specifically, when there is only one local optimum, it is referred to as a bi-modal objective function. The challenge in many optimisation problems is that the objective function has many local optima, but the goal is to find the global optimum x^*, the best overall solution, not just the best nearby solution.

Many optimisation problems are captured by the *black-box scenario*, where the internal structure or specific details of the problem instance are not known to the algorithm, and only the input-output relationship can be evaluated (Droste et al., 2006). The *black-box optimisation problem* is formally described as follows. Let $\mathcal{S}$ be a finite set (search space) and x_t be the search point in $\mathcal{S}$ at the t-th evaluation, where $t \in \mathbb{N}$. Let $\mathcal{F}$ be a class of functions $\mathcal{F} \subseteq \{f : \mathcal{S} \to \mathbb{R}\}$. The objective of *black-box optimisation algorithms*, such as EAs, is to identify the optimal solution $x^* \in \mathcal{S}$ for the objective function f, such that for all search points $x \in \mathcal{S}, f(x^*) \geq f(x)$. They use the information $(x_1, f(x_1)), \ldots, (x_t, f(x_t))$ observed at each time $t \in \mathbb{N}$, where the search point x_t they queried is based on observations from the $t-1$ previous search points. Note that f belongs to a class of functions $\mathcal{F}$. The algorithm has no prior knowledge of the objective function f of the problem instance, but can make use of the knowledge of the problem class $\mathcal{F}$ and obtain information about it by evaluating search points (retrieving objective values of search points) within the search space (Droste et al., 2006). Additionally, we introduce a related concept in black-box scenario, termed as *black-box complexity* (Droste et al., 2006). This measure signifies the difficulty of solving a black-box optimisation problem using black-box optimisation algorithms. The black-box complexity is the minimum worst-case of the number of queries required to identify the optimum of any black-box algorithm on class $\mathcal{F}$.

Evolutionary algorithms (EAs) are a type of randomised heuristic algorithms that are widely used in black-box optimisation aiming to identify good solutions, although not necessarily optimal ones. They are inspired by Darwinian evolution (Darwin, 1859), which involves *selection* and *variation*. The concept of selection posits that individuals who are

better adapted to the environment within a population have a higher chance of producing offspring. New traits appear in populations through random variations in the genomes of individuals, such as mutation and recombination, ultimately contributing to the creation of fitter individuals and driving the evolution of the population. EAs simulate natural evolution with these two key processes. EAs maintain a population of individuals. The number of individuals in a population is referred to as the *population size*. Each individual specifies the genome, i.e., the *genotype* of the individual, and represents a search point of the problem instance z, i.e., the *phenotype* of the individual. Typically, EAs possess a mapping from the genotype to the phenotype of an individual, denoted as $\phi : \mathcal{Q}_z \rightarrow \mathcal{S}_z$, where $\mathcal{Q}_z$ is a set containing all possible genotypes for the problem instance z, and $\mathcal{S}_z$ is the search space of the problem instance z. The fitness value of an individual is determined by the objective value of the search point represented by the individual x, i.e., $f(\phi(x))$. Random variation acts on the genotype of the individual. In one generation, the term *parent* denotes the chosen individual, while *offspring* refers to the varied individual derived from the parent. The sets of parents and offsprings are referred to as the *parent population* and *offspring population*, respectively. In order to optimise an objective function, often referred to as the *fitness function*, EAs execute the following steps (Bäck et al., 1997):

1. Initialisation: Create a initial population (parent population).

2. Variation: Produce an offspring population by introducing variations in individuals from the parent population.

3. Selection: Choose individuals from the parent and offspring populations to establish a new parent population. This selection is based on fitness values of individuals in populations and utilises various selection mechanisms.

4. Repeat steps 2-3 until the optimal solution is found or the stopping criterion is met.

A generation of EAs refers to one repetition of steps 2-3. The field that investigates EAs and other bio-inspired randomised heuristic algorithms is known as *evolutionary computation* (EC). The history of EC dates back to the 1950s, marked by numerous independent contributions (Fogel, 1998). For a comprehensive review of the early work in this field, we recommend the survey by Fogel (1998). In the field of EC, there are numerous algorithms. Some EAs include: *evolution strategies* (ESs), developed by Rechenberg (1978) and Schwefel (1981); *genetic algorithms* (GAs), introduced by Holland (1975) and further studied by De Jong (1975); *genetic programmings* (GPs) as presented by Koza (1989). More details about EAs can be found in Section 2.2.1.

In contrast to many other optimisation methods, EAs typically only assume black-box access, while mathematical optimisation methods often require a closed-form description of the objective function, such as in linear programming (Vanderbei, 2020) or mixed-integer programming (Achterberg and Wunderling, 2013). Furthermore, many optimisation methods are designed for specific problems, e.g., interior point methods for convex optimisation (Renegar, 2001). Gradient descent represents a popular class of optimisation algorithms in machine learning, can be effective when the objective function is smooth and possesses a well-defined gradient. Nevertheless, it is inapplicable when no gradient is available, as in discrete search spaces (Ruder, 2017). In contrast, EAs do not require a gradient, they are so-called gradient-free methods.

When evaluating the performance of an EA, we want to understand the amount of computational resources. Specifically, following the standard approach in computer science, we want to understand the relationship between the problem size and the time to find desired solutions (usually an optimal solution) to a given problem instance by the algorithm. In general, runtime is the number of primitive operations performed before the algorithm terminates. In term of EAs, the computational resources needed on fitness function evaluations is often much greater than that on the rest of operations of the algorithm in each generation

(Jansen, 2013). As such, the *runtime* of an EA on a problem instance is usually defined in terms of the number of objective function evaluations required to find the optimal solution for the first time. A formal definition of runtime can be found in Section 2.2.3.

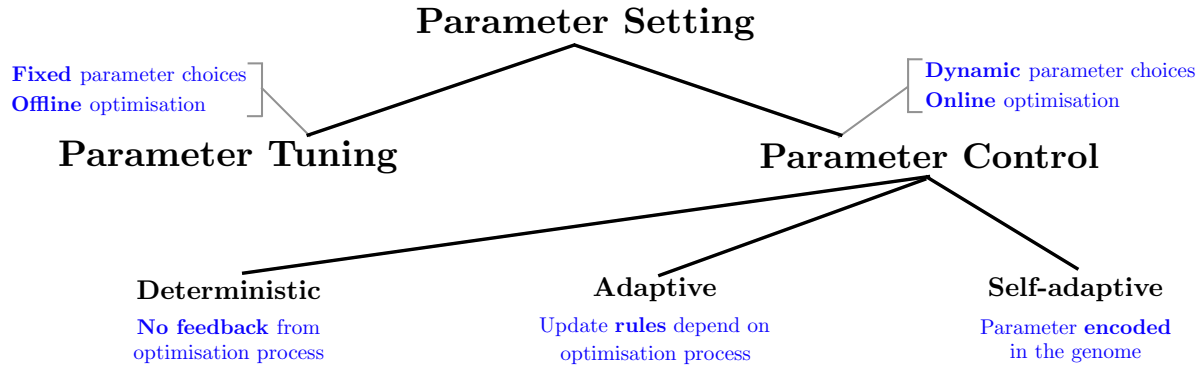

Figure 1.1: Classification of parameter setting method in EAs (Eiben et al., 1999)

The performance of EAs is significantly dependent on algorithm parameter settings which can determine the behaviour of an algorithm, such as parameters controlling the strength of mutation, the probability of recombination, and the population size (Lobo et al., 2007). Section 2.2.1 presents some classical EAs and their parameters. Proper parameter settings can lead to reduced runtime of the EA on the objective function. These settings are often instance-specific (Lobo et al., 2007), meaning the proper parameter setting for one problem instance may differ considerably from the settings needed for another problem instance. Given our interest in reducing the runtime of an algorithm across various problem instances, it is crucial to configure the parameters appropriately. To configure parameters appropriately, a large number of parameter setting techniques have been developed. A popular taxonomy proposed by Eiben et al. (1999) (see Figure 1.1) categorises these techniques into two main classes, *parameter tuning* and *parameter control*, depending on whether parameters are *static*, i.e., the parameter setting do not change during runtime. A common method for parameter tuning involves conducting initial tests on a set of benchmark problems, and selecting the most promising static parameter setting based on a statistical analysis of the test outcomes. Dynamically adapting parameters during the optimisation process is called

parameter control. There are different mechanisms for parameter control, including *deterministic*, *adaptive*, and *self-adaptive* mechanisms (Eiben et al., 1999). One promising approach to configuring parameters is self-adaptive parameter control mechanism (Bäck, 1992; Schwefel, 1981), where both the solution and the parameters are encoded within genomes of individuals and evolve together through variation operators, so that the genotype of each individual not only determines a search point (phenotype) but also its own parameters. The hypothesis is that an individual with an appropriate parameter setting has greater potential to improve its fitness value, thus creating an indirect selective bias towards individuals with better parameter settings. We refer to an EA using the self-adaptive parameter control mechanism as a self-adaptive EA, with its example implementation shown in Section 2.2.2. Detailed background information about parameter settings and self-adaptive parameter control can be found in Sections 2.3.1 and 2.3.2. Although numerous theoretical and empirical studies exist on self-adaptive parameter control mechanisms, the benefits of these mechanisms in EAs remain uncertain in various optimisation scenarios, including those involving noisy and dynamic functions, as well as multi-modal functions.

1.2 Research Questions

This thesis aims to understand self-adaptive parameter control mechanisms in EAs through theoretical analysis and subsequently improve self-adaptive EAs. This thesis addresses four research questions related to self-adaptive parameter control mechanisms in EC. These questions will be investigated through rigorous mathematical proofs and supplemental experimental analyses.

Research Question 1: *Is it possible for self-adaptive parameter control mechanisms to determine the "good" parameter setting relative to noise?*

In the subsequent discussion detailed in Section 2.3.3.1, the robustness of static EAs relies on "right" parameter settings that are associated with the presence and intensity of noise. In a black-box scenario, both the type of noise and the level of noise remain unidentified for the algorithms. Consequently, the primary research question emerges: Can self-adaptive EAs determine the "right" parameter settings, thereby facilitating rapid optimisation under noise?

Research Question 2: *Can self-adaptive parameter control mechanisms adapt parameter settings in dynamic optimisations?*

In dynamic optimisation scenarios, the "good" parameter settings may vary over time, potentially leading to unsuccessful optimisations for static EAs. For additional information about dynamic optimisation, please refer to Section 2.3.3.2. Self-adaptation controls parameters during the optimisation process. Thus, the subsequent research question arises: Can self-adaptive parameter control mechanisms facilitate the adaptation of parameter settings while optimising dynamic functions?

Research Question 3: *Can self-adaptive parameter control mechanisms assist in escaping from local optima?*

Referencing our discussion in Section 2.3.4, there is evidence suggesting that "right" parameter settings are crucial for escaping from local optima in optimisation. Thus, the third research question is posed: Can self-adaptive parameter control mechanisms facilitate escape from local optima?

Research Question 4: *Are self-adaptive EAs efficient in addressing complex combinatorial optimisation problems?*

Referring to Section 2.3.2, although a few empirical studies suggest that self-adaptive EAs can effectively solve complex combinatorial optimisation problems, a comprehensive em-

pirical study is still lacking. It would be intriguing to understand potential benefits of self-adaptation in more complicated optimisation problems, rather than solely on theoretical benchmarking functions. Therefore, the final research question arises: Can EAs with self-adaptive parameter control mechanisms significantly enhance the optimisation of complex combinatorial problems?

1.3 Contributions and Outline

This thesis contributes to the understanding of self-adaptive parameter control mechanisms in EAs primarily through rigorous mathematical proofs, complemented by experiments. Firstly, we delve into the significance of appropriate parameter settings for non-elitist EAs under the influence of uncertainty, which provides a compelling motivation for using self-adaptation to configure suitable parameter settings. Secondly, we undertake theoretical and empirical analyses of self-adaptive EA in noisy environments. Thirdly, we explore the potential of self-adaptation in dynamic optimisation, providing theoretical evidence of its capability to enhance the performance of EAs. Fourthly, we propose a novel self-adaptive EA from a multi-objective optimisation perspective, called the *multi-objective self-adaptive EA* (MOSA-EA) and prove that it efficiently escapes a local optimum on a bi-model function. Finally, we conduct an extensive empirical analysis of the MOSA-EA, showcasing its superior performance in complex combinatorial optimisation problems. As an ancillary discovery, we propose a variant of the level-based theorem, providing a lower bound for the probability of finding the current optimum by algorithm within a specified evaluation budget. Each chapter is the result of collaborative work with Lehre, and the associated publication for every chapter is mentioned at the beginning of the respective chapter. We now provide an overview of all the subsequent chapters.

In Chapter 2, we provide the necessary background for the studies conducted in this thesis. We introduce essential notations, algorithms, analysis tools, benchmark functions, and uncertainty models. Additionally, we present related work pertaining to the theme of this thesis, encompassing parameter settings in EAs, self-adaptive parameter control mechanisms, and pertinent results of EAs optimising under uncertainty and on multi-modal functions.

In Chapter 3, we analyse the runtime of non-elitist EAs using fixed parameter settings on two classical benchmark functions, OneMax and LeadingOnes, under one-bit, bitwise, Gaussian, and symmetric noise models, as well as a dynamic optimisation problem DBV (see Sections 2.2.1, 2.2.4, and 2.2.5 for definitions of algorithms, benchmarking functions, noise models, and the dynamic optimisation problem, respectively). Our analyses are more comprehensive and precise compared to the previous study on non-elitist EAs by Dang and Lehre (2015). We demonstrate that, in several scenarios, the non-elitist EAs outperform the current state-of-the-art results. Furthermore, we provide more precise guidance on how to choose the mutation rate, the selective pressure, and the population size as a function of the level of uncertainty. One of the significant conclusions derived from this chapter is that appropriate parameter settings for non-elitist EAs under the influence of noise should be adjusted in relation to the level of noise. Nonetheless, in real-world optimisation problems, for instance, in a black-box scenario, the noise level is typically unknown. This provides a compelling motivation for the application of a self-adaptive parameter control mechanism.

The main contribution of Chapter 4 is the first theoretical analysis of the self-adaptive EA in a noisy environment. The rigorous runtime analysis on the LeadingOnes problem shows that the 2-tournament EA with self-adapting from high/low mutation rates can guarantee the lowest runtime among the fixed high/low mutation rates and the uniformly chosen mutation rate from high/low rates, regardless of the presence of symmetric noise. In addition, we extend to more types of noise, one-bit and bit-wise noise, and a self-adaptation mechanism that adapts the mutation rate from a given interval $(0, 1/2]$ in the empirical study. The

experimental results show that self-adaptive EAs can adapt to noise levels and outperform EAs with fixed mutation rates. In conclusion, we provide both theoretical and empirical evidence that self-adaptation can enhance the noise-tolerance of EAs, addressing Research Question 1.

In Chapter 5, we explore whether self-adaptation can be beneficial in dynamic optimisation. We specifically examine a tracking dynamic optima problem with changing structure that require adjustable parameter settings. The structure refers to the number of relevant bits (discussed in Section 2.3.2). This problem requires algorithms to successively find and hold the solutions that match a sequence of bit-flipping and length-varying target bitstrings (structure-changing optima) within specified evaluation budgets. We show that EAs with any fixed mutation rate get lost with constant probability somewhere during tracking this dynamic optimisation problem, resulting in an exponentially small probability of achieving the final optimum. Therefore, the variable mutation rates may be necessary to reliably track the moving optimum. The main contribution is the first rigorous study of self-adaptive parameter control mechanisms on dynamic optimisation. We demonstrate that the (μ, λ) self-adaptive EA proposed by Case and Lehre (2020) (described in Section 2.2.2) can track every optimum in this dynamic optimisation problem and reach the final optimum with an overwhelmingly high probability. Another significant contribution of this chapter is the introduction of a level-based theorem with tail bounds. Level-based theorems serve as some of the most important theoretical tools for deriving the runtime of non-elitist population-based EAs (Corus et al., 2018; Dang et al., 2021b; B. Doerr and Kötzing, 2021), as introduced in Section 2.2.3.1. However, existing level-based theorems merely provide the expected runtime. To evaluate the proficiency of a self-adaptive EA in tracking dynamic optima, it is essential to determine a lower bound for the probability of achieving the current optimum within a specified evaluation budget. To fulfil these requirements, we develop a new variant of the level-based theorem. The primary conclusion drawn from this chapter is that self-

adaptation of mutation rates can enhance the performance of EAs in dynamic optimisation, thereby answering Research Question 2.

In Chapter 6, we propose a novel self-adaptive EAs for single-objective optimisation, the MOSA-EA. This algorithm approaches parameter control from a multi-objective optimisation perspective, simultaneously maximising both fitness and mutation rates. Due to the nature of this approach, individuals in "dense" fitness valleys can survive high mutation rates, while individuals on "sparse" local optima can only survive with lower mutation rates, thereby allowing them to co-exist on a non-dominated Pareto front. The concepts of "sparsity" and "density" are further elaborated in Section 2.3.4. Runtime analyses demonstrate that the MOSA-EA efficiently escapes a local optimum of unknown sparsity on a bi-modal function, unlike certain fixed mutation rate EAs, which can become entrapped. The most significant contribution of this chapter is the proposal of a new method for self-adapting parameter settings in EAs. This method has been proven to efficiently escape a particular type of local optima, thereby addressing Research Question 3.

In Chapter 7, we extend our study of the MOSA-EA through a comprehensive empirical analysis. Our findings reveal that the MOSA-EA delivers performance comparable to that on unimodal functions, and significantly surpasses eleven randomised search heuristics on a bi-modal function with "sparse" local optima. Regarding complex combinatorial optimisation problems, the MOSA-EA increasingly outperforms other algorithms on more challenging instances of NK-Landscape and Max-k-Sat optimisation problems. Most notably, the MOSA-EA outperforms a problem-specific MaxSat solver on several difficult Max-k-Sat instances. These results suggest that self-adaptation through multi-objectivisation can be effectively utilised to control parameters in non-elitist EAs. The significance lies in the fact that self-adaptive EAs demonstrate strong performance not only on theoretical benchmarking functions but also on complex combinatorial optimisation problems. They show promise in solving real-world problems, thereby addressing Research Question 4.

In Chapter 8, we conclude this thesis by highlighting the most significant findings from our results and providing an overview of the open questions in the field of self-adaptation.

1.4 Publications

During the PhD study, the author of this dissertation has published his research through various publications, as enumerated in Table 1.1. Note that, in adherence to the convention within the field of theoretical computer science, the authorship in publications 1-4 and 6-8 is listed in alphabetical order.

Table 1.1: Publications during the PhD study

#	**Title**	**Authors**	**Jour./Conf.**	**Status**
1.	*More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments.*	Lehre, Qin	GECCO 2021	Published
2.	*More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments.*	Lehre, Qin	*Algorithmica*	Published
3.	*Self-adaptation via Multi-objectivisation: A Theoretical Study*	Lehre, Qin	GECCO 2022	Published
4.	*Fast Non-elitist Evolutionary Algorithms with Power-law Ranking Selection.*	Dang, Eremeev, Lehre, Qin	GECCO 2022	Published
5.	*Self-adaptation via Multi-objectivisation: An Empirical Study.*	Qin, Lehre	PPSN 2022	Published
6.	*Fast Non-elitist Evolutionary Algorithms with Power-law Ranking Selection*	Dang, Eremeev, Lehre, Qin	*Algorithmica*	Submitted
7.	*Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms.*	Lehre, Qin	FOGA 2023	Published
8.	*Self-adaptation Can Help Evolutionary Algorithms Track Dynamic Optima.*	Lehre, Qin	GECCO 2023	Published
9.	*Optimizing Chance-Constrained Submodular Problems with Variable Uncertainties.*	Yan, Do, Shi, Qin, Neumann	ECAI 2023	Published

Chapter Two

Background

2.1 Introduction

This chapter provides an overview of the thesis background, focusing on preliminaries and relevant studies related to self-adaptive parameter control mechanisms. The chapter is structured as follows: Section 2.2 introduces notations, algorithms (Section 2.2.1), runtime analysis (Section 2.2.3), uncertainty models (Section 2.2.5), and benchmarking functions (Section 2.2.4). Section 2.2 primarily focuses on definitions of notations and algorithms, with further discussion appearing in Section 2.3, which presents related work. This section covers parameter settings (Section 2.3.1), self-adaptation in EAs (Section 2.3.2), and EAs in uncertain environments and on multi-modal landscapes (Sections 2.3.3 and 2.3.4, respectively).

2.2 Preliminaries

This section introduces the algorithms, runtime analyses, uncertainty models, and benchmarking functions studied in this thesis. Note that the proposed algorithms, analysis tools, models, and functions will not be covered in this section. For clarity and convenience, we first introduce some notations in this thesis. We use $\mathbb{N} := \{1, 2, \ldots\}$ to denote positive integers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ to denote non-negative integers, and define the following sets of integers: $[n] := \{i \in \mathbb{N} \mid i \leq n\}$ for $n \in \mathbb{N}$, $[0..n] := [n] \cup \{0\}$ for $n \in \mathbb{N}$, and $[a..b] := [b] \setminus [a-1]$ where $a, b \in \mathbb{N}$ and $a \leq b - 1$. This thesis focuses on the optimisation of pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$ where $n \in \mathbb{N}$ is called the problem instance size. The search space of pseudo-Boolean functions is binary, i.e., $\{0,1\}^n$. We call elements of $\{0,1\}^n$ bitstrings of length n with bit values $x = (x_1, \ldots, x_n)$. We use 1^n and 0^n to denote all 1-bit and all 0-bit bitstrings of length n, respectively. We use $x_{1:m}$ to denote the substring $x_{1:m} = (x_1, \ldots, x_m)$ for any $m \in [n]$. Table 2.1 presents a summary of notations used in this thesis.

Table 2.1: Notation used in this thesis.

Notation	Definition
$\Pr(\cdot)$	Probability
$\Pr(\cdot \mid \cdot)$	Conditional probability
$E[\cdot]$	Expectation
$\mathrm{Unif}(A)$	Uniform distribution over a set A
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2, where $\mu, \sigma \in \mathbb{R}$
$\mathrm{Bin}(n, p)$	Binomial distribution with n trials and success probability p, where $n \in \mathbb{N}$ and $p \in [0, 1]$
$\ln(\cdot)$	Natural logarithm
$\log(\cdot)$	Logarithm with base 2
$\lvert x \rvert$	Length of a bitstring $x \in \{0,1\}^n$ for $n \in \mathbb{N}$
$\lvert A \rvert$	Cardinality of a set A
$\cdot \diamond \cdot$	Bit-string concatenation

In the analysis of optimisation algorithms, it is often necessary to use metrics that quantify the neighbourhood relationship between two search points within a search space. For example, we may need to measure how "close" the current solution is to some optimum. In binary search spaces, the *Hamming distance* is a widely accepted method for evaluating the distance between two bitstrings. This concept is also fundamental in constructing benchmarking functions and is defined as follows.

Definition 2.2.1. *Given two bitstrings $x, y \in \{0,1\}^n$ with bit values $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ for any $n \in \mathbb{N}$, then the Hamming distance between x and y is*

$$\mathrm{H}(x, y) := \sum_{i=1}^{n} |x_i - y_i|. \tag{2.1}$$

We also define the *Hamming shell* for a bitstring, which includes all bitstrings with a

specified Hamming distance.

Definition 2.2.2. *Given a bitstring $x \in \{0,1\}^n$ and $k \in [n]$ for any $n \in \mathbb{N}$, then the k-Hamming shell of x are*

$$\mathrm{N}_k(x) := \{y \mid \mathrm{H}(x,y) = k\}, \tag{2.2}$$

in particular, for the special case $k = 1$, we use the notation

$$\mathrm{N}(x) :=: \mathrm{N}_1(x) := \{y \mid \mathrm{H}(x,y) = 1\}. \tag{2.3}$$

Chapter 5 examines the dynamic substring matching problem to evaluate whether there are benefits of self-adaptive parameter control mechanisms. In this context, the algorithms must identify solutions where the bit value of each bit position aligns with the corresponding bit value of a changing target substring. To formalise this, we define the match function between two bitstrings as follows.

Definition 2.2.3. *Given two bitstrings $x \in \{0,1\}^n$ and $y \in \{0,1\}^\ell$ where $\ell \in [n]$ for any $n \in \mathbb{N}$, define the matching function*

$$\mathrm{M}(x,y) := \begin{cases} 1 & \text{if } \mathrm{H}(x_{1:|y|}, y) = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Additionally, a bitstring x is said to match a substring y if $\mathrm{M}(x,y) = 1$.

2.2.1 Evolutionary Algorithms (EAs)

EAs are inspired by Darwinian evolutionary theory and have been applied to a wide range of optimisation problems across various fields, including engineering, economics, planning, and computer science (Chen, 2002; Fleming and Purshouse, 2002; Harman et al., 2012; Slowik and Kwasnicka, 2020; Tong et al., 2022). As mentioned in Section 1.1, EAs maintain a

population of individuals. These individuals are subject to random variations and selection based on a fitness function, which is used to evaluate the quality of each individual, and it guides the selection mechanism which selects the individuals that will be reproduced to form a new generation of individuals.

Variation is one of two primary processes of EAs, which typically comprises mutation and recombination. Mutation involves modifying some portion of the genotype of an individual to generate a new individual. In contrast, recombination, such as *crossover*, involves merging genotypes of two or more individuals to generate a new individual. Using EAs to optimise pseudo-Boolean functions, genotypes and phenotypes of individuals are typically identical, represented as bitstrings. The mutation operator for bitstring is commonly described as a random mapping from bitstrings to bitstrings $\mathrm{M} : \{0,1\}^n \to \Omega \to \{0,1\}^n$, where Ω represents the underlying sample space. For convenience, we use $\mathrm{M}(x)$ to denote the bitstring obtained from mutating bitstring x, i.e., a random variable. The *bit-wise mutation operator* is one of the most widely used mutation operators. It independently flips each bit of a bistring x with probability χ/n, where $\chi/n \in (0, 1/2]$. We refer to the probability χ/n as the *mutation rate* and χ as the *mutation parameter*. We call EAs which use a recombination operator genetic algorithms (GAs). We direct readers to (Bäck et al., 1997) for a comprehensive exploration of recombination operators. One of the goals of this thesis is to investigate the behaviour and performance of self-adaptation. As a starting point for theoretical study, this thesis focuses on simple EAs that utilise only the mutation operator and self-adapting mutation rate mechanisms. By studying EAs that rely solely on mutation, we can gain valuable insights into the essential role of self-adaptation in the evolutionary process and its impact on the optimisation. We leave the study of self-adaptation involving crossover to future work.

Selection is the second essential process in EAs. The main purpose of selection mechanisms is to identify the better solutions in a population, which refers to both individuals with high

fitness values and those with high potential to achieve high fitness values (individuals with lower fitness values that are, nonetheless, a short Hamming distance from those with high fitness values). There are various selection mechanisms that can be categorised as either *elitist* or *non-elitist*, also known as preservative and extinctive, respectively (Bäck, 1992; Bäck and Hoffmeister, 1991; Baker, 1989). Elitist selection mechanisms always preserve the best individuals from the current population for the next generation. In contrast, non-elitist selection mechanisms do not always copy one or more of the fittest individual from a generation to the next generation. There exist several comparative theoretical studies between elitism and non-elitism in EAs (Dang et al., 2021a,b; B. Doerr, 2022; Jagerskupper and Storch, 2007). Although it may appear counterintuitive to discard the fittest individuals during optimisation, recent studies have shown that non-elitism can be advantageous in facilitating escape from local optima (Dang et al., 2021a,b; Dang and Lehre, 2016b). However, careful tuning of parameters, such as the mutation rate, is crucial (Lehre, 2010). Further discussion about parameter settings of non-elitist EAs is provided in the rest of this section.

Algorithm 1 (1+1) EA

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Mutation rate $\chi/n \in (0, 1/2]$.

Require: Initial individual $x_0 \in \{0,1\}^n$.

1: **for** $\tau = 0, 1, 2, \ldots$ until termination condition met **do**

2: Create x' by independently flipping each bit of x_t with probability χ/n.

3: **if** $f(x') \geq f(x_\tau)$ **then**

4: Set $x_{\tau+1} \leftarrow x'$.

5: **else**

6: Set $x_{\tau+1} \leftarrow x_\tau$.

As demonstrated in the preceding framework in Section 1.1, EAs are typically population-based. Due to the difficulty of analysing stochastic processes involving multiple individuals,

the simplest variant of EAs, which contains only one individual, can serve as a starting point for theoretical analysis (Droste, 2002). The (1+1) EA shown in Algorithm 1 is one of the most extensively studied elitist algorithms in the theory of EC. It is designed to optimise pseudo-Boolean functions. This single-individual algorithm begins with the initialisation of an individual x_0 usually sampled uniformly at random from the search space $\{0,1\}^n$. In each generation $\tau \in \mathbb{N}_0$, an offspring x' is produced by bit-wise mutation of the parent x_τ. If the fitness value of the offspring x' is equal to or greater than that of the parent x_τ, then x' becomes the new parent $x_{\tau+1}$ in the next generation; otherwise, set the new parent $x_{\tau+1}$ to x_τ. This simple algorithm has only one parameter, the mutation rate, which has a significant impact on the performance of the algorithm (B. Doerr and C. Doerr, 2020; B. Doerr et al., 2013; Witt, 2013).

Algorithm 2 $(\mu + \lambda)$ EA

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population sizes $\lambda, \mu \in \mathbb{N}$.

Require: Mutation rate $\chi/n \in (0, 1/2]$.

Require: Initial population $P_0 \in (\{0,1\}^n)^\mu$.

1: **for** $\tau = 0, 1, 2, \ldots$ until termination condition met **do**

2: **for** $i = 1, \ldots, \lambda$ **do**

3: Set $z \leftarrow P_\tau(k)$ where $k \sim \text{Unif}([\mu])$.

4: Set $P(i) \leftarrow y$,

where y is created by independently flipping each bit of z with probability χ/n.

5: Set $P' \leftarrow P_t \cup P$.

6: Sort P' such that $f(P'(1)) \geq \ldots \geq f(P'(\lambda + \mu))$.

7: Set $P_{\tau+1} \leftarrow (P'(1), \ldots, P'(\mu))$.

We can further extend the (1+1) EA to a general version of elitist EAs, such as the $(\mu + \lambda)$ EA presented in Algorithm 2, which involves multiple individuals, i.e., a population.

We use P to denote a population and $P(i)$ to denote the i-th individual from population $P \in (\{0,1\}^n)^\lambda$, where $i \in [\lambda]$. The $(\mu + \lambda)$ EA involves a parent population $P_\tau \in (\{0,1\}^n)^\mu$ of size $\mu \in \mathbb{N}$, and an offspring population $P \in (\{0,1\}^n)^\lambda$ of size $\lambda \in \mathbb{N}$. Similar to the (1+1) EA, each individual of the initial parent population P_0 is usually sampled uniformly at random from the search space $\{0,1\}^n$. In each generation $\tau \in \mathbb{N}_0$, each individual of the offspring population P is produced by selecting a parent uniformly at random with replacement from the parent population P_τ and applying bit-wise mutation. After fitness evaluations, the μ best individuals of the parent and the offspring populations $P_\tau \cup P$ form the next parent population $P_{\tau+1}$. Some variations of this algorithm include the $(\mu$+1) EA and the (1+λ) EA, where the offspring population size is specified as $\mu = 1$ and the parent population size is specified as $\lambda = 1$, respectively. The elitist EAs are used as baseline or comparative algorithms in this thesis.

Algorithm 3 Non-elitist EA (Lehre, 2011)

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Selection mechanism $P_{\text{sel}} : (\{0,1\}^n)^\lambda \to \Omega \to [\lambda]$, where Ω is the underlying sample space.

Require: Mutation rate $\chi/n \in (0, 1/2]$.

Require: Initial population $P_0 \in (\{0,1\}^n)^\lambda$.

1: **for** $\tau = 0, 1, 2, ...$ until termination condition met **do**

2: **for** $i = 1$ to λ **do**

3: Sample $I_\tau(i) \sim P_{sel}(P_\tau)$; set $z := P_\tau(I_\tau(i))$.

4: Set $P_{\tau+1}(i) \leftarrow y$,

where y is created by independently flipping each bit of z with probability χ/n.

Non-elitist EAs differ from elitist EAs in that they replace the entire current parent population with the offspring population, rather than keeping the best individuals. Algorithm 3

presents a framework for non-elitist EAs. In each generation $\tau \in \mathbb{N}_0$, to create an individual y for the next population $P_{\tau+1}$, the algorithm first produces an individual z via a random mapping from the current population P_τ to a index $P_{\text{sel}} : (\{0,1\}^n)^\lambda \to \Omega \to [\lambda]$, where Ω is the underlying sample space. The algorithm then applies a bit-wise mutation operator. Analogously to the mutation operator, we omit the sample space parameter, and let $P_{\text{sel}}(P)$ refer to the random variable which is the index of selected individual from the population P. Each individual in the next population $P_{\tau+1}$ is created independently. The notation $I_\tau(i)$ represents the index of the i-th selected individual in the τ-th generation. There are numerous selection mechanisms available in the literature, and interested readers may refer to (D. E. Goldberg and Deb, 1991) for further exploration.

Algorithm 4 k-tournament selection mechanism

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Tournament size $k \in [\lambda]$.

Require: Population $P \in (\{0,1\}^n)^\lambda$.

1: **for** $j = 1$ to k **do**

2: Set $I_j \backsim \text{Unif}([\lambda])$.

3: Set $i \leftarrow \arg\max_{\ell \in \{I_1 \dots I_k\}} f\left(P(\ell)\right)$.

4: **return** i.

In Chapter 3, we analyse the performance of non-elitist EAs to demonstrate the importance of parameter tuning under uncertainty. This analysis involves two commonly used selection mechanisms: k-tournament selection and (μ, λ) selection. Algorithm 4 presents the k-tournament selection mechanism, which selects the fittest individual from k uniformly at random with replacement chosen individuals $P(I_1), P(I_2), \dots, P(I_k)$ from the population P, with tournament size $k \in [\lambda]$ and population size $\lambda \in \mathbb{N}$. For the convenience of later analysis, we illustrate the special case of k-tournament selection, called 2-tournament selection

Algorithm 5 2-tournament selection mechanism

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Population $P \in (\{0,1\}^n)^\mu$.

1: Set $x_1 \leftarrow P(I_1)$ where $I_1 \backsim \text{Unif}([\lambda])$

2: Set $x_2 \leftarrow P(I_2)$ where $I_2 \backsim \text{Unif}([\lambda])$

3: **if** $f(x_1) \geq f(x_2)$ **then**

4: Set $i \leftarrow I_1$.

5: **else**

6: Set $i \leftarrow I_2$.

7: **return** i.

or binary tournament selection. In this case, only two individuals enter the tournament for each selection, as shown in Algorithm 4. Algorithm 6 defines the (μ, λ) selection mechanism, which uniformly at random selects an individual from the μ individuals with the highest fitness. It is worth noting that in practice, we usually sort population P once in each generation for convenience. For clarity, we refer to the 2-tournament EA as Algorithm 3 using Algorithm 5 as the selection mechanism, and the (μ, λ) EA as Algorithm 3 using Algorithm 6 as the selection mechanism.

Algorithm 6 (μ, λ) selection mechanism

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Selection parameter $\mu \in \mathbb{N}$, where $\lambda \geq \mu \geq 1$.

Require: Population $P \in (\{0,1\}^n)^\lambda$.

1: Sort P such that $f(P(1)) \geq \ldots \geq f(P(\lambda))$.

2: $i \backsim \text{Unif}([\mu])$.

3: **return** i.

2.2.2 Self-adaptive EAs

Parameter settings play a crucial role in EAs (B. Doerr et al., 2013; Lehre and Yao, 2012; Lobo et al., 2007). *Parameter control* aims to dynamically configure algorithm parameters during optimising. This will be discussed in detail in Section 2.3.1. We say an EA is *static* when it uses a fixed parameter setting, meaning the parameter setting remains constant throughout the entire optimisation process. In contrast, we say an EA is *dynamic* when the parameter setting adapts during the process. All the EAs mentioned in Section 2.2.1 are classified as static. Self-adaptation is a promising parameter control mechanism where genotypes of individuals contain not only the solution but also the parameters, and the population evolves via variation operators and selection. As a result, each individual in a *self-adaptive population* possesses not only its represented solution but also its parameters. This thesis focuses on the theoretical analysis of self-adaptation, and this section introduces several previously studied algorithms. To the best of our knowledge, there are only three papers in the theory of EC that focus on self-adaptive mechanisms, and they all solely concentrate on self-adapting mutation rates (Case and Lehre, 2020; Dang and Lehre, 2016b; B. Doerr et al., 2021). The self-adaptive population in these papers can be represented as $P \in \mathcal{Y}^\lambda$, where the state space $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$ represents the binary search space and the interval of mutation rate. To facilitate the analysis, we define the set of solutions (phenotypes) of a self-adaptive population P as $Q := (x_i)_{i\in[\lambda]}$, where $P = (x_i, \chi_i/n)_{i\in[\lambda]}$, and use $Q(i)$ to denote the solution of the i-th individual from self-adaptive population P, where $i \in [\lambda]$ and $\lambda \in \mathbb{N}$ population size. In the rest of this section, we will present self-adaptive EAs from existing papers. The results and conclusions of these papers will be discussed in Section 2.3.2.

The aforementioned papers proposed and studied three self-adaptive EAs: the 2-tournament self-adaptive EA using two mutation rates proposed by Dang and Lehre (2016b), the (μ, λ)

Algorithm 7 Framework of self-adaptive EAs

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Sorting partial order $\succeq_{P,f}$.

Require: Selection mechanism $P_{\text{sel}} : \mathcal{Y}^\lambda \to \Omega \to [\lambda]$, where Ω is the underlying sample space and $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$.

Require: Self-adapting mutation rate strategy $D_{\text{mut}} : (0, 1/2] \to \Omega \to (0, 1/2]$, where Ω is the underlying sample space.

Require: Initial self-adaptive population $P_0 \in \mathcal{Y}^\lambda$.

1: **for** $\tau = 0, 1, 2, \ldots$ until termination condition met **do**
2: Sort P_τ such that $P_\tau(1) \succeq_{P_\tau,f} \ldots \succeq_{P_\tau,f} P_\tau(\lambda)$
3: **for** $i = 1, \ldots, \lambda$ **do**
4: Sample $I_\tau(i) \sim P_{sel}(P_\tau)$; set $(x, \chi/n) := P_\tau(I_\tau(i))$.
5: Sample $\chi'/n \sim D_{\text{mut}}(\chi/n)$.
6: Create x' by independently flipping each bit of x with probability χ'/n.
7: Set $P_{\tau+1}(i) := (x', \chi'/n)$.

self-adaptive EA proposed by Case and Lehre (2020), and the $(1, \lambda)$ self-adaptive EA proposed by B. Doerr et al. (2021). In any generation $\tau \in \mathbb{N}_0$, these algorithms essentially sort the self-adaptive population P_τ by fitness function f and mutation rate χ/n utilising a specific *sorting partial order*. Despite the importance of fitness values, the parameter values of individuals are also crucial for self-adaptive EAs. Therefore, the sorting partial order is used to define the relationship between two individuals in the self-adaptive population. Each individual in the next population $P_{\tau+1}$ is then produced via selection and mutation, with the selection mechanism based on the order of the sorted population. The mutation rate of the selected individual changes based on the some self-adapting mutation rate strategy, and is then bit-wisely flipped with the probability of the new mutation rate. To describe these pro-

cesses, we provide a framework for self-adaptive EAs (Algorithm 7) which covers all studied self-adaptive EAs in this thesis. The framework presented allows users to customise several components, including the sorting partial order $\succeq_{P,f}$ (e.g., Definitions 2.2.4(a)-(c)), the selection mechanism $P_{\text{sel}} : \mathcal{Y}^\lambda \to \Omega \to [\lambda]$, where Ω is the underlying sample space (e.g., Algorithms 8 and 9), and the self-adapting mutation rate strategy $D_{\text{mut}} : (0, 1/2] \to \Omega \to (0, 1/2]$, where Ω is the underlying sample space (e.g., Algorithm 10). For convenience, we omit the sample space parameter, and let $P_{\text{sel}}(P)$ and $D_{\text{mut}}(\chi/n)$ refer to random variables which is the index in a sorted self-adaptive population P of the selected individual, and the self-adapted mutation rate, respectively. Similarly to the static non-elitist framework shown in Algorithm 3, the notation $I_\tau(i)$ represents the index of the i-th selected individual in the τ-th generation.

Algorithm 8 2-tournament selection mechanism (self-adaptive EAs)

Require: Pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Selection parameter $\mu \in \mathbb{N}$, where $\lambda > \mu \geq 1$.

Require: Sorted self-adaptive population $P_t \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0, 1\}^n \times (0, 1/2]$.

1: $i_1 \backsim \text{Uniform}([\lambda])$

2: $i_2 \backsim \text{Uniform}([\lambda])$

3: **if** $i_1 \leq i_2$ **then**

4: $\quad i \leftarrow i_1$.

5: **else**

6: $\quad i \leftarrow i_2$.

7: **return** i.

There are two types of sorting partial orders used in the papers mentioned above. Specifically, the 2-tournament self-adaptive EA using two mutation rates (Dang and Lehre, 2016b) uses a fitness-only sorting partial order (Definition 2.2.4(a)), which only sorts the self-

Algorithm 9 (μ, λ) selection mechanism (self-adaptive EAs)

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Selection parameter $\mu \in \mathbb{N}$, where $\lambda > \mu \geq 1$.

Require: Sorted self-adaptive population $P_t \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$.

1: $i \backsim \text{Uniform}([\mu])$

2: **return** i.

adaptive population based on their fitness values, without considering their mutation rates. On the other hand, the (μ, λ) self-adaptive EA (Case and Lehre, 2020) and the $(1, \lambda)$ self-adaptive EA (B. Doerr et al., 2021) sort the self-adaptive population by fitness values and then by mutation rates of individuals. The former prefers higher values for both fitness and mutation rates, while the latter prefers high fitness values and lower mutation rates, which are described in Definitions 2.2.4(b) and 2.2.4(c), respectively. The sorting partial orders defined in Definition 2.2.4 rely solely on fitness and mutation rate. However, the notation $\succeq_{P,f}$ involves the entire population P. This is because, in Chapter 6, we propose a new self-adaptive EA with a sorting partial order that involves the entire population. For generality, we use the notation $\succeq_{P,f}$ with P for sorting partial order.

Definition 2.2.4. *Consider a self-adaptive population $P \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$ and $n \in \mathbb{N}$. Let $f : \{0,1\}^n \to \mathbb{R}$ be a Pseudo-Boolean function, then for all $(x, \chi/n), (x', \chi'/n) \in \mathcal{Y}$,*

(a) the fitness-only sorting partial order (Dang and Lehre, 2016b) is defined as

$$(x, \chi/n) \succeq_{P,f} (x', \chi'/n) \iff f(x) \geq f(x') . \tag{2.5}$$

(b) the fitness-first sorting partial order preferring higher mutation rate (Case and Lehre, 2020) is defined as

$$(x, \chi/n) \succeq_{P,f} (x', \chi'/n) \iff f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \geq \chi') . \tag{2.6}$$

(c) The fitness-first sorting partial order preferring lower mutation rate (B. Doerr et al., 2021) is defined as

$$(x, \chi/n) \succeq_{P,f} (x', \chi'/n) \Longleftrightarrow f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \leq \chi') . \tag{2.7}$$

The self-adapting mutation rate strategy is a crucial component of self-adaptive EAs. In the 2-tournament self-adaptive EA using two mutation rates (Dang and Lehre, 2016b), only two mutation rates are used, and the self-adapting mutation rate switches to the other mutation rate with probability p_{c}. In the (μ, λ) self-adaptive EA (Case and Lehre, 2020), the selected individual inherits an increased mutation parameter $A\chi/n$ with probability p_{inc}, and a reduced mutation parameter $b\chi/n$ otherwise. This self-adapting mutation rate strategy (shown in Algorithm 10) allows mutation rates to self-adapt from ϵ to $1/2$. Here, $A > 1$, $\epsilon \in (0, 1/2)$ and $b, p_{\mathrm{inc}} \in (0, 1)$ are algorithm parameters. In the $(1, \lambda)$ self-adaptive EA (B. Doerr et al., 2021), the new mutation rate χ'/n is sampled uniformly at random from $\{\chi/(nF), F\chi/n\}$ where $F > 1$, with a half chance to increase and a half chance to decrease.

Algorithm 10 Self-adapting mutation rate strategy in (Case and Lehre, 2020)

Require: Increasing factor $A > 1$.

Require: Decreasing factor $b > 1$.

Require: Minimal mutation rate $\epsilon \in (0, 1/2)$.

Require: Probability for increasing $p_{\mathrm{inc}} \in (0, 1)$.

Require: Mutation rate χ/n.

1: $\chi'/n := \begin{cases} \min(A\chi/n, 1/2) & \text{with probability } p_{\mathrm{inc}}, \\ \max(b\chi/n, \epsilon) & \text{otherwise.} \end{cases}$

2: **return** χ'/n.

Overall, the self-adaptive EAs can be instantiated by using the framework of self-adaptive EAs and customised components. For instance, the (μ, λ) self-adaptive EA proposed by Case and Lehre (2020) can be represented as Algorithm 7, using the fitness-first sorting

partial order with preferring higher mutation rate (Definition 2.2.4(b)), the (μ, λ) selection (Algorithm 9), and the self-adapting mutation rate strategy presented in Algorithm 10.

2.2.3 Runtime Analysis

As discussed in Section 1.1, EAs primarily address black-box optimisation problems, and runtime is a crucial metric for evaluating algorithm performance. The runtime of an EA on a problem instance is usually defined in terms of the number of objective function evaluations required to find the optimal solution for the first time. *Runtime analysis* is a popular theoretical approach for mathematically evaluating the performance of EAs and other randomised heuristic algorithms. Many powerful mathematical techniques for runtime analysis of EAs have since been developed. We recommend readers consult books authored or edited by Auger and Doerr (2011), B. Doerr and Neumann (2019), Jansen (2013), Neumann and Witt (2010), and Zhou et al. (2019) for further investigation into theoretical analysis in EC. In this section, we will introduce some runtime analysis tools used in this thesis. We first provide a formal definition of runtime of black-box algorithms in Definition 2.2.5.

Definition 2.2.5 (Runtime (Droste et al., 2006))**.** *The runtime T of black-box optimisation algorithm A on fitness function $f : \mathcal{S} \to \mathbb{R}$, where $\mathcal{S}$ is a finite search space and $x_t \in \mathcal{S}$ is the search point at the t-th evaluation for $t \in \mathbb{N}$, is*

$$T_{A,f} := \min_{t \in \mathbb{N}} \{t \mid \forall y \in \mathcal{S}, f(x_t) \geq f(y)\}. \tag{2.8}$$

Compared to deterministic algorithms, EAs and other randomised heuristic algorithms make some random choices. As a result, it is possible to have two identical runs, but it is unlikely. Furthermore, they may not produce the same result on a given input if run multiple times. Therefore, the runtime is a random variable because the algorithm makes random decisions (e.g., mutation). The fact that two runs can be different is a consequence of the fact

that the algorithm makes random decisions. Hence, we are usually interested in the *expected runtime* $E[T_{A,f}]$. Asymptotic notations (also known as *Bachmann-Landau notations*) are commonly used to classify functions based on their growth rates. Asymptotic notations are also frequently used in the analysis of EAs to describe the expected runtime as a function of the problem size (Jansen, 2013). Definition A.1.1 defines asymptotic notations rigorously which is shown in Appendix A.1. Often, we are more interested in obtaining bounds that hold with high probability rather than focusing solely on the expected runtime (Auger and Doerr, 2011). For this purpose, the *tail bound* on the runtime, sometime referred to as *success probability*, is introduced to describe the probability of finding the optimal solution within a given runtime budget $\mathcal{T}$, e.g., $\Pr(T_{A,f} < \mathcal{T})$. This concept is commonly employed to describe the inefficiency of algorithms in the theory of EC. For example, we say the algorithm fails on the function if the probability of the runtime of the algorithm on the function being within e^{cn} (exponential runtime) is at most $e^{-\Omega(n^{\varepsilon})}$, i.e., $\Pr(T_{A,f} < e^{cn}) = e^{-\Omega(n^{\varepsilon})}$, where $c, \varepsilon > 0$ are constants.

Drift theorems are powerful tools for estimating the expected runtime and tail bound on the runtime, which are widely used in the theory of EC (B. Doerr and L. A. Goldberg, 2013; B. Doerr et al., 2012; Hajek, 1982; He and Yao, 2004; Oliveto and Witt, 2011). We recommend readers consult the chapter (Lengler, 2020) in a textbook for a comprehensive survey of drift analysis. Based on drift theorems, many advanced analysis tools have been developed. For instance, *level-based theorems* (Corus et al., 2018; Dang et al., 2021b; B. Doerr and Kötzing, 2021) are employed to obtain an upper bound on the expected runtime of non-elitist population-based algorithms, while the *negative drift theorem for populations* is applied to determine tail bounds on the runtime of such algorithms. These methods are presented in Sections 2.2.3.1 and 2.2.3.2, respectively, and are frequently utilised throughout this thesis. In Chapter 3, we utilise the level-based theorem (Corus et al., 2018) to propose a general theorem for obtaining the upper bounds of expected runtime for 2-tournament EAs

optimising under uncertainty. We also employ the negative drift theorem for populations (Lehre, 2010) to identify when non-elitist EAs fail under noise. Moreover, we use level-based theorems (Corus et al., 2018; Dang et al., 2021b) to derive the upper bounds of expected runtime for self-adaptive EAs on dynamic and multi-modal optimisation in Chapters 5 and 6. Additionally, Section 2.2.3.3 introduces a tool for obtaining the lower bound of the expected runtime of mutation-only EAs as a function of mutation rate. In Chapter 4, we use this tool to demonstrate that static EAs with low mutation rates can be slow, serving as motivation for employing self-adaptive mutation rates.

2.2.3.1 Level-based Theorems

The level-based theorems (Corus et al., 2018; Dang et al., 2021b; B. Doerr and Kötzing, 2021) are general tools that provide an upper bound of the runtime of non-elitist algorithms on a wide variety of optimisation problems (Case and Lehre, 2020; Dang et al., 2021b; Dang et al., 2019; Lehre and P. T. H. Nguyen, 2021; P. T. H. Nguyen, 2021), which follow the scheme of Algorithm 11 with a population $P_\tau \in \mathcal{X}^\lambda$, where $\mathcal{X}$ is a finite state space. In this thesis, we first introduce the level-based theorem proposed by Corus et al. (2018) for runtime analysis. This theorem is one of the most widely used versions in runtime analysis and is presented in Theorem 2.2.1. Assume that the search space $\mathcal{X}$ is partitioned into ordered disjoint subsets (called *levels*) $A_1, \ldots, A_m$. Let $A_{\geq j} := \cup_{k=j}^{m} A_k$ be the search points in level j and higher, and let D be some random mapping from the set of all possible populations $\mathcal{X}^\lambda$ into the space of probability distributions of $\mathcal{X}$, i.e., $D : \mathcal{X}^\lambda \to \Omega \to \mathcal{X}$, where Ω is the underlying sample space. For convenience, we omit the sample space parameter, and let $D(P)$ refer to the random variable of individual. Given any subset $A \subseteq \mathcal{X}$, we define $|P_\tau \cap A| := |\{i \mid P_\tau(i) \in A\}|$, i.e., the number of individuals in P_τ that belong to A. To estimate an upper bound on the time of existing individuals at the final level A_m using Theorem 2.2.1, three conditions must be satisfied: (G1) requires that the probability of

level "upgrading", i.e., sampling an individual in higher levels, is non-zero; (G2) requires the probability of the number of individuals in higher levels "growing"; (G3) requires a sufficient population size.

Algorithm 11 Population-based Algorithm (Corus et al., 2018)

Require: Finite state space $\mathcal{X}$.

Require: Population size $\lambda \in \mathbb{N}$.

Require: Map $D : \mathcal{X}^\lambda \to \Omega \to \mathcal{X}$, where the underlying sample space Ω.

Require: Initial population $P_0 \in \mathcal{X}^\lambda$.

1: **for** $\tau = 0, 1, 2, ...$ until termination condition met **do**

2: **for** $i = 1$ to λ **do**

3: Sample $P_{\tau+1}(i) \sim D(P_\tau)$.

Theorem 2.2.1 (Level-based theorem (Corus et al., 2018)). *Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$, let $T := \min\{\tau\lambda \mid |P_\tau \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_τ of Algorithm 11. If there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0, 1]$, and $\gamma_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$,*

(G1) for all $j \in [m-1]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ then

$$\Pr_{y \sim D(P)}(y \in A_{\geq j+1}) \geq z_j,$$

(G2) for all $j \in [m-2]$, and all $\gamma \in (0, \gamma_0]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda$ then

$$\Pr_{y \sim D(P)}(y \in A_{\geq j+1}) \geq (1+\delta)\gamma,$$

(G3) and the population size $\lambda \in \mathbb{N}$ satisfies

$$\lambda \geq 4/(\gamma_0\delta^2) \ln\left(128m/(z_*\delta^2)\right), \text{ where } z_* := \min\{z_j\},$$

then

$$E[T] \leq \frac{8}{\delta^2} \sum_{j=1}^{m-1} \left(\lambda \ln\left(\frac{6\delta\lambda}{4 + z_j\delta\lambda}\right) + \frac{1}{z_j}\right).$$

To address the issue of "deceptive" regions B that contains individuals with a higher selection probability but at a lower level, Dang et al. (2021b) proposed a new level-based theorem (Theorem 2.2.2). Theorem 2.2.2 includes an additional condition (G0) that requires the probability of producing a "deceptive" individual to decrease if there are many individuals in the "deceptive" region. Conditions (G1) and (G2) are relaxed to only hold when there are sufficiently few "deceptive" individuals in the population. The "deceptive" region is a concept in the study of fitness landscapes, which will be discussed in Section 2.3.4.

Theorem 2.2.2 (New level-based theorem (Dang et al., 2021b)). *Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$ and a subset $B \subset \mathcal{X}$, let $T := \min\{\tau\lambda \mid |P_\tau \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of Algorithm 11. If there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0, 1]$, and $\gamma_0, \psi_0 \in (0, 1)$ such that for any population $P \in \mathcal{X}^\lambda$,*

(G0) for all $\psi \in [\psi_0, 1]$, if $|P \cap B| \leq \psi\lambda$ then

$$\Pr_{y \sim D(P)}(y \in B) \leq (1 - \delta)\psi,$$

(G1) for all $j \in [m-1]$, if $|P \cap B| \leq \psi_0\lambda$ and $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ then

$$\Pr_{y \sim D(P)}(y \in A_{\geq j+1}) \geq z_j,$$

(G2) for all $j \in [m-2]$, and all $\gamma \in (0, \gamma_0]$, if $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda$ then

$$\Pr_{y \sim D(P)}(y \in A_{\geq j+1}) \geq (1 + \delta)\gamma,$$

(G3) and the population size $\lambda \in \mathbb{N}$ satisfies

$$\lambda \geq 12/(\gamma_0\delta^2) \ln\left(300m/(z_*\delta^2)\right), \text{ where } z_* := \min\{z_j\},$$

then

$$E[T] \leq \frac{12\lambda}{\delta} + \frac{96}{\delta^2} \sum_{j=1}^{m-1} \left(\lambda \ln\left(\frac{6\delta\lambda}{4 + z_j\delta\lambda}\right) + \frac{1}{z_j}\right).$$

The target algorithm of level-based theorems (Algorithm 11) is a general algorithm with a population size of $\lambda \in \mathbb{N}$. In each generation τ, an individual from the next population P_τ is independently sampled from a given probability distribution D over the current population P_τ. This framework can be applied to many algorithms in Section 2.2.1. For instance, if we assume that the finite state space $\mathcal{X}$ of Algorithm 11 is $\{0,1\}^n$, then the map D can represent processes of selection and mutation (Lines 3-4) of Algorithm 3. Similarly, if we divide the space $\{0,1\}^n \times (0,1/2]$ of Algorithm 11 into a finite state space as $\mathcal{X}$, then the map D can also represent Lines 3-7 of Algorithm 7.

2.2.3.2 Negative Drift Theorem for Populations

In the analysis of EAs, apart from the upper bounds of runtime, identifying the situation when the runtime of algorithms is exponential is another critical aspect. The *negative drift theorem for populations* (Lehre, 2010) is applied to obtain exponential tail bounds on the runtime of population selection-variation algorithms with a finite state space, such as Algorithm 11. In this thesis, it is sufficient to utilise Corollary 2.2.1 (Lehre, 2010), which is a corollary of the negative drift theorem for populations. This corollary is specific to non-elitist EAs which can be described in Algorithm 3. Before stating the corollary, we define the *reproductive rate* α_0 (Lehre, 2010) of the individual $P_\tau(i)$ in Algorithms 3 and 7 as the expected number of times the individual is sampled from $P_{sel}(P_\tau)$, i.e., $E[R_\tau(i) \mid P_\tau]$, where $R_\tau(i) := \sum_{j=1}^{\lambda} \delta_{I_\tau(j),i}$ for $\tau \in \mathbb{N}$ and $i \in [\lambda]$, where

$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b, \\ 0 & \text{if } a \neq b. \end{cases}$$

It is easy to compute the maximal reproductive rates of the 2-tournament selection and the (μ,λ) selection (without uncertainty), which are

$$\begin{aligned}\alpha_0 &= \lambda\left(\left(\frac{1}{\lambda}\right)^2 + 2\left(1-\frac{1}{\lambda}\right)\lambda\right)\\ &= 2(1-1/\lambda)\\ &\approx 2\end{aligned}$$

if $\lambda \to \infty$, e.g., $\lambda \in \Omega(\log(n))$, and

$$\begin{aligned}\alpha_0 &= \lambda\frac{1}{\mu}\\ &= \frac{\lambda}{\mu},\end{aligned}$$

respectively (Lehre and Yao, 2012). The corollary is then stated in Corollary 2.2.1.

Corollary 2.2.1 ((Lehre, 2010))**.** *The probability that Algorithm 3 with population size $\lambda = \text{poly}(n)$, mutation rate χ/n and maximal reproductive rate bounded by $\alpha_0 < e^{\chi} - \delta$, for a constant $\delta > 0$, optimises any function with a polynomial number of optima within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Based on Corollary 2.2.1, we can determine the tolerance limit of the mutation rate, which is known as the *error threshold* (Lehre, 2010; Ochoa, 2006). If the mutation rate is greater than the error threshold, the probability that the runtime of the non-elitist EA is exponentially large on any function is close to 1. This threshold value is related to the reproductive rate of the algorithm. Specifically, the non-elitist EA has an error threshold when the mutation rate $\chi/n > \ln(\alpha_0 - \delta)$, where α_0 is the maximal reproductive rate of the algorithm, and $\delta > 0$ is a constant. For example, the error thresholds of the k-tournament EA and the (μ, λ) EA are approximately $\ln(k)$ and $\ln(\lambda/\mu)$, respectively.

2.2.3.3 Tool to Derive Lower Bounds of Runtime

To justify the use of the self-adaptive parameter setting instead of static one in this thesis, we would like to determine the "speed limit" of EAs using static mutation rates, that is, a lower bound for the expected runtime. Algorithm 12 is a framework for EAs that use only the bit-wise mutation operator. It can be used to represent both elitist and non-elitist EAs, such as Algorithms 1, 2, and 3. Consequently, Theorem 2.2.3 (Sudholt, 2013) gives the lower bound on the runtime for such EAs with respect to the mutation rate.

Algorithm 12 Framework of a mutation-based EA (Sudholt, 2013)

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Initial population $\{x_1, \dots x_\mu\}$.

1: **for** $t = \mu, \mu+1, \mu+2, ...$ until termination condition met **do**

2: Select a parent x from $\{x_1, \dots, x_t\}$ according to t and $f(x_1), \dots f(x_t)$.

3: Create x_{t+1} by independently flipping each bit of x with probability χ'/n.

Theorem 2.2.3 ((Sudholt, 2013))**.** *The expected runtime of every mutation-based EA using mutation rate χ/n on every function with a unique optimum is at least $\left(\frac{\ln(n)-\ln(\ln(n))-3}{\chi(1-\chi/n)^n}\right) n$, if $2^{-n/3}/n \leq \chi \leq 1$.*

2.2.4 Benchmarking Functions

We first present the simple theoretical functions OneMax and LeadingOnes that not only serve as well-known pseudo-Boolean benchmarking functions but also serve as the foundation for constructing more complex functions. In the theoretical analysis of EAs, OneMax and LeadingOnes functions assume a significant role, often employed to compare the performance of algorithms.

Definition 2.2.6. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textsc{OneMax}(x) :=: OM(x) := \sum_{i=1}^{n} x_i. \tag{2.9}$$

Definition 2.2.7. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textsc{LeadingOnes}(x) :=: LO(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j. \tag{2.10}$$

We additionally introduce two widely-used bi-modal benchmarking functions, namely $\textsc{Jump}_k$ and Cliff.

Definition 2.2.8. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textit{Jump}_k(x) := \begin{cases} n - OM(x) & \text{if } n - k < OM(x) < n, \\ k + OM(x) & \text{otherwise.} \end{cases} \tag{2.11}$$

Definition 2.2.9. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textit{Cliff}(x) := \begin{cases} OM(x) & \text{if } OM(x) \leq 2n/3, \\ OM(x) - n/3 + 1/2 & \text{otherwise.} \end{cases} \tag{2.12}$$

We consider two complex combinatorial optimisation problems to evaluate performance of EAs in empirical study of this thesis (see Chapter 7), the random NK-Landscape problem (Kauffman and Weinberger, 1989) and the random maximum k-satisfiability (Max-k-Sat) problem. The NK-Landscape model (Kauffman and Weinberger, 1989) was constructed as hard fitness landscapes which involving complex dependencies among decision variables, resulting in many local optima. The NK-Landscape problem can be defined as:

Definition 2.2.10. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, $k \in \mathbb{N}$ satisfying $k \leq n$, and a set of sub-functions $f_i : \{0,1\}^k \to \mathbb{R}$ for $i \in [n]$, then*

$$\textit{NK-Landscape}(x) := \sum_{i=1}^{n} f_i\left(\Pi\left(x, i\right)\right) \tag{2.13}$$

where the function $\Pi : \{0,1\}^n \times [n] \to \{0,1\}^k$ *returns a bit-string containing* k *right side neighbours of the* i*-th bit of* x*, i.e.,* $x_i, \ldots, x_{(i+k) \mod n}$.

The parameter k of the NK-Landscape problem represents the length of bit-strings for each sub-function. Typically, each sub-function is defined by a lookup table with 2^k entries, each in the interval $(0, 1)$. To generate a random instance of a landscape, a uniform sampling method can be applied to each value within the lookup table, where values are sampled from a range between 0 and 1. Moreover, changing k can vary the difficulty of an instance (Ochoa et al., 2008). As a general rule, instances are considered to become more challenging for larger values of k (Ochoa et al., 2008).

The Max-k-Sat problem is a complex combinatorial optimisation problem used as a benchmarking function in this thesis. The Sat problem was the first known NP-complete problem (Cook, 1971). The optimisation version of this problem, the Max-k-Sat problem, aims to find an assignment that maximises the number of satisfied clauses (or equivalently, minimises the number of unsatisfied clauses) in a given Boolean formula in conjunctive normal form (Achlioptas and Moore, 2006; Coja-Oghlan, 2014; Gottlieb et al., 2002). The definition of the Max-k-Sat problem as a pseudo-Boolean function optimisation is as follows: Let the logic values TRUE and FALSE be represented by the integers 1 and 0, respectively, so that each clause in conjunctive normal form can be represented as a function $C_i : [k] \to [n]$, We say the k is the length of clause C_i.

Definition 2.2.11. *Given a bitstring* $x \in \{0,1\}^n$ *for* $n \in \mathbb{N}$*, a set* $\mathcal{C}$ *containing* k*-length clauses where* $|\mathcal{C}| = m$ *for all* $m, k \in \mathbb{N}$ *satisfying* $k \leq n$*,*

$$\textit{Max-k-Sat}_{\mathcal{C}}(x) := \sum_{i=1}^{m} L_i(x) \tag{2.14}$$

where the functions $L_i(x)$ indicate the truth value of the i-th clause:

$$L_i(x) := \begin{cases} 1 & \text{if } \bigvee_{j=1}^{k} x_{C_i(j)} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

To generate a random instance of MAX-k-SAT, m clauses are randomly created, and each clause consists of k literals, which are uniformly and randomly selected without replacement from the set $[n]$. According to (Coja-Oghlan, 2014) the probability of generating a satisfiable instance decreases from almost 1 to almost 0 when the ratio of the number of clauses m to the problem size n exceeds a threshold, denoted by $r_{k-\text{SAT}}$. The threshold is computed to be $2^k \ln(2) - \frac{1}{2}(1 + \ln(2)) + o_k(1)$, where $o_k(1)$ represents a term that tends to 0 as k increases. In this thesis, a random MAX-k-SAT instance is considered hard if the ratio m/n is above $r_{k-\text{SAT}}$. For example, the threshold $r_{k-\text{SAT}}$ for MAX-k-SAT is approximately 2133 if we ignore the $o_k(1)$ term.

2.2.5 Noise and Dynamic Models

Real-world optimisation problems often involve uncertainty, such as noise and dynamics, which can be categorised as noisy optimisation and dynamic optimisation problems, etc. This section aims to introduce noise models and dynamic functions studied in this thesis. In Section 2.3.3, we will discuss related work on uncertain optimisation. In Chapter 3, we demonstrate that non-elitist EAs can handle uncertainty effectively with proper parameter settings. In Chapter 4, we show that 2-tournament EAs can efficiently solve optimisation problems under symmetric noise without the need for tuning parameters. Additionally, in Chapter 5, we propose a dynamic optimisation problem to demonstrate the efficiency of the self-adaptive EA, where static mutation-only EAs fail.

In noisy optimisation problems, the exact objective values of a search point cannot be

precisely obtained, which we refer to as the noisy fitness function. A noisy evaluation can be described as a random mapping from search points to real numbers, i.e., $f^n : \mathcal{X} \to \Omega \to \mathbb{R}$, where Ω is a sample space and $\mathcal{X}$ is a search space. For convenience, we omit the sample space Ω and denote $f^n(x)$ as the noisy fitness value of the objective function f for a search point $x \in \mathcal{X}$. Normally, sampled noise fitness values are independent of each other. The noise level is an indicator that describes the degree of noise affecting the evaluation in noisy optimisation, and we will define it on a case-by-case basis. A higher noise level typically indicates a greater impact of noise on the evaluation, making optimisation more challenging for the algorithm.

Noise models in pseudo-Boolean optimisation can be categorised into two classes: the *prior noise* randomly flips one or several bits in the search point before each evaluation, e.g. one-bit noise and bit-wise noise, while the *posterior noise* makes some changes on the fitness value after each evaluation, e.g. Gaussian noise and symmetric noise (Gießen and Kötzing, 2016). The one-bit noise model (q) (Dang and Lehre, 2015; Droste, 2004; Friedrich et al., 2016; Lehre and P. T. H. Nguyen, 2019; Qian et al., 2019; Sudholt, 2021) serves as the simplest starting point for theoretical analysis. It flips at most one bit of a bitstring (search point) then evaluates it using the function f, which can be defined as follows:

Definition 2.2.12 (One-bit noise model)**.** *Given a probability $q \in [0, 1]$ which represents the noise level, and a bistring $x \in \{0, 1\}^n$, then*

$$f^n(x) = \begin{cases} f(x) & \text{with probability } 1 - q, \\ f(x') & \text{with probability } q \end{cases} \tag{2.16}$$

where $x' \sim \text{unif}(\mathrm{N}(x))$.

Real-world optimisation problems are often subject to noise that can affect multiple bits rather than just one. Thus, the bit-wise noise model (p) (Gießen and Kötzing, 2016; Qian et al., 2019; Sudholt, 2021) can more closely imitate such reality. It flips each bit of a bitstring

independently with some probability (similar to the bitwise mutation operator) and then evaluates it using the function f, as defined below:

Definition 2.2.13 (Bit-wise noise model)**.** *Given a probability $p \in [0,1]$ which represents the noise level, and a solution $x \in \{0,1\}^n$, then*

$$f^n(x) = f(x') \tag{2.17}$$

where x' is obtained by independently flipping each bit of x with probability p.

The Gaussian noise model (σ^2) (Dang and Lehre, 2015; Friedrich et al., 2015, 2016; Gießen and Kötzing, 2016; Qian et al., 2018; Rowe and Aishwaryaprajna, 2019) is a type of posterior noise that adds a value independently sampled from a normal distribution to the objective value of each evaluation. It can be defined as follows:

Definition 2.2.14 (Gaussian noise model)**.** *Given a parameter $\sigma \geq 0$ which represents the noise level, and a bistring $x \in \{0,1\}^n$, then*

$$f^n(x) = f(x) + \mathcal{N}(0, \sigma^2). \tag{2.18}$$

The symmetric noise model as a posterior noise was first introduced by Qian et al. (2021). They showed that the (1+1) EA with a resampling strategy can fail in the symmetric noise model with a fixed noise level of $q = 1/2$, while using a population can be helpful. In this thesis, we propose a more general symmetric noise model using a variable noise level q. The model is defined as follows:

Definition 2.2.15 (Symmetric noise model)**.** *Given a probability $q \in [0,1]$ which represents the noise level, and an arbitrary number $C \in \mathbb{R}$ which is a parameter of the noise model, and a bistring $x \in \{0,1\}^n$, then*

$$f^n(x) = \begin{cases} f(x) & \text{with probability } 1-q, \\ C - f(x) & \text{with probability } q. \end{cases} \tag{2.19}$$

In dynamic optimisation problems, the objective function changes over time. It can be described as follows: there is a sequence of functions $f_t : \mathcal{X} \to \mathbb{R}$, where a search space $\mathcal{X}$ and $t \in \mathbb{N}$, with f_t representing the objective function applied in the t-th evaluation. Additionally, a sequence of *optimal regions* exists, which are sets of acceptable search points corresponding to the function f_t. A formal description of dynamic optimisation is shown in Definition 2.2.16. The goal of an algorithm solving dynamic optimisation problems is to find search points in optimal regions.

Definition 2.2.16 (Dynamic optimisation (Dang et al., 2017))**.** *A dynamic function F is a random sequence of functions $(f_t)_{t\in\mathbb{N}}$, where $f_t : \mathcal{X} \to \mathbb{R}$ and a search space $\mathcal{X}$ for all $t \in \mathbb{N}$. The optimal regions associated with F is the sequence $(\mathrm{OPT}_t)_{t\in\mathbb{N}}$, where $\mathrm{OPT}_t = \arg\max_x f_t(x)$.*

Dynamic optimisation is more challenging than static optimisation, as the optimal solution or the functions may change over time. For instance, the weight of each bit position in the dynamic linear function (Lengler and Schaller, 2018) changes over time. Weights of each objective function is independently sampled from a distribution. The DBV problem is a special case of dynamic linear functions, which was first proposed by (Lengler and Meier, 2020). It can be considered as a dynamic version of the classical function $\textsc{BinVal}(x) := \sum_{i=1}^{n} 2^{n-i} \cdot x_i$, which is defined as:

Definition 2.2.17. *Given a bistring $x \in \{0,1\}^n$,*

$$DBV_\tau(x) := \sum_{i=1}^{n} 2^{n-i} \cdot x_{\pi_\tau(i)} \tag{2.20}$$

*where $\tau \in \mathbb{N}_0$ is the number of rounds * and $\pi_\tau : [n] \to [n]$ is a permutation drawn uniformly at random from all possible permutations of $[n]$.*

In practice, it is common to evaluate each individual in population P_t once before selection and mutation. However, in such cases, each comparison under uncertainty in the

*Typically, a round contains all evaluations in a generation of the algorithm.

2-tournament and $(1 + 1)$ selection is not independent, making the analysis challenging. Therefore, we can apply the reevaluation strategy (Dang and Lehre, 2015; Friedrich et al., 2016; Gießen and Kötzing, 2016; Qian et al., 2019; Qian et al., 2021; Qian et al., 2018) in this situation, which involves reevaluating the noisy fitness value of an individual every time it enters a tournament. Note that this strategy is only applicable to the (1+1) EA and the 2-tournament EA in this thesis.

2.3 Related Work

2.3.1 Parameter Settings in EAs

As shown in Section 2.2.1, EAs are parameterised algorithms, which require several parameters such as mutation rate and selection parameter to be identified. The configuration of these parameters has a significant impact on the performance of EAs, which has been extensively studied through runtime analysis. For example, considering any linear function and a constant $c > 0$, the runtime of the (1+1) EA is as follows: super-polynomial if the mutation rate $\chi/n \in \omega\left(\log(n)/n\right)$ or $\chi/n = o(n^{-c})$; polynomial if $\chi/n \in \Omega(n^{-c})$ and $\chi \in O(\log(n)/n)$; and $O(n\log(n))$ if $\chi/n = c/n$ (Witt, 2013). The appropriate mutation rate is also influenced by uncertainty. The runtime of the (1+1) EA algorithm is no longer $O(n\log(n))$ for all mutation rates $\chi/n = c/n$ where the constant $c > 0$ on dynamic linear functions. If the mutation rate is above a threshold value $\chi/n = c/n > c_0/n \approx 1.59/n$, then the runtime becomes super-polynomial (Lengler and Schaller, 2018). Parameter settings in non-elitist EAs are also crucial, as there is a "balance" between the reproduction rate and the mutation rate, which refers to the error thresholds introduced in Section 2.2.1. For instance, the runtime of the (μ, λ) EA on any function with a polynomial number of optima becomes exponential if the mutation parameter χ is above λ/μ (Lehre, 2010).

Parameter settings typically depend on the problem instance and optimisation environment, making the identification of appropriate parameter settings a challenging task (B. Doerr and C. Doerr, 2020; Lobo et al., 2007). As mentioned in Section 1.1, Eiben et al. (1999) classified parameter setting techniques into parameter tuning and parameter control (see Figure 1.1). A prevalent approach for parameter tuning involves conducting preliminary tests, evaluating their performance, and selecting the most promising static parameter setting for optimisation. Meanwhile, parameter control, which includes deterministic, adaptive, and self-adaptive mechanisms, dynamically adjusts parameters during the optimisation process. Deterministic parameter control mechanisms adjust algorithm parameter settings based on deterministic rules, typically time-based. Adaptive parameter control mechanisms update parameter settings depending on the optimisation process. In self-adaptive parameter control mechanisms, parameters are encoded within the genomes of individuals and evolve together through variation operators. The related work of them will be presented in Sections 2.3.1.1, 2.3.1.2 and 2.3.2, respectively.

2.3.1.1 Parameter Tuning

Parameter tuning aims to identify the fixed (nearly) optimal parameter setting for given problems during the whole optimisation process. Through rigorous mathematical proofs, we can determine the global optimal parameter setting. Theoretical results such as runtime analyses usually require knowledge of the optimisation problem, e.g., the instance, structure, and environment. However, obtaining this information is usually difficult and the information about the fitness function is typically unknown in real-world optimisation. Therefore, in practice, the main idea of parameter tuning is to run some initial tests and observe their performance. Then we configure the most promising parameter values for the algorithm. There are many parameter tuning tools that can automatically identify reasonable parameter settings. One of the representative methods is *irace*, an automated configuration tool for

optimisation algorithms built by López-Ibáñez et al. (2016). It uses iterated racing steps to identify the appropriate parameter values. Similar algorithm configuration frameworks include *ParamILS* (Hutter et al., 2009) and *CALIBRA* (Adenso-Diaz and Laguna, 2006), etc. Alternatively, Hutter et al. (2011) and Ansotegui et al. (2015) applied machine learning to predict good parameter settings and proposed the model-based approaches *SMAC* and *GGA++*, respectively. For a comprehensive review of parameter tuning, we direct readers to the survey by Huang et al. (2020).

In general, parameter tuning methods can be challenging to use in real optimisation problems because they often require a significant amount of time to run. Furthermore, the recommended parameter values are static, and the optimal parameter setting may change during the optimisation process. For instance, the optimal mutation rate of the (1+1) EA on LeadingOnes depends on the current best fitness value achieved, which is given by $1/(\mathrm{LO}(x)+1)$, where x is the current parent individual (Böttcher et al., 2010). In a similar vein, Jansen and Wegener (2006) presented a function illustrating that the (1+1) EA algorithm, employing a series of time-dependent mutation rates, discovers the optimum in polynomial time. Conversely, using any fixed mutation rate needs more than polynomial time with an extremely high probability. Moreover, the fitness function may change over time in practice, i.e., dynamic optimisation, requiring that the parameter settings change accordingly. Therefore, it is necessary to change the parameters with respect to the current optimisation situation.

2.3.1.2 Deterministic and Adaptive Parameter Control

Exploration and *exploitation.* Exploration involves searching in completely new areas of the search space, while exploitation involves searching in areas of the search space that are close to previously promising regions (Črepinšek et al., 2013). All search algorithms should bal-

ance both exploration and exploitation to efficiently search the entire search space. In early studies, it was believed that for exploration, low selective pressure and high-intensity variation were required, while for exploitation, high selective pressure and conservative variation were required (Črepinšek et al., 2013). Deterministic parameter control mechanisms adjust the parameters of algorithms to adapt selective pressure and strength of variation based on deterministic rules that are typically based on time (Eiben et al., 1999). For example, simulated annealing (Kirkpatrick et al., 1983) is a famous heuristic algorithm with a deterministic parameter control mechanism, in which the selective pressure increases with time.

However, the schedule should be problem or instance-specific, making it challenging to choose a specialised schedule for each independent problem. Therefore, we can update rules depending on the optimisation process, called adaptive parameter control mechanism or self-adjusting parameter control mechanism. Intuitively, the algorithm can be in the exploration phase if the individuals are easy to improve after variation. The algorithm could also be stuck at some local optimum far from the global optimum. Based on this intuition, Rechenberg (1978) proposed the well-known $1/5^{th}$-success-rule. This algorithm increases its mutation rate if the observed success rate is larger than 1/5, i.e., at least one out of five solutions is improved; otherwise, it decreases the mutation rate. In the rest of this section, we will focus on previous studies of adaptive parameter control mechanisms within the theory of EC. It is worth mentioning that two PhD theses by Rajabi (2022) and Hevia Fajardo (2023) have conducted outstanding work in theory on adapting mutation rate and population size, respectively.

The first prominent topic in this field concerns adapting the mutation rate. A straightforward method for tuning the mutation rate on-the-fly involves basing the mutation rate on the current fitness value. For example, the mutation rate may be inversely proportional to the fitness value, meaning that as the fitness value of a population increases, the mutation rate decreases, shifting from exploration to exploitation. The motivation for adjusting the

mutation rate relative to fitness is to balance exploration and exploitation (Črepinšek et al., 2013). In the referenced study (Böttcher et al., 2010), the expected runtime of the (1+1) EA, when utilising a dynamic fitness-dependent mutation rate, is significantly lower than that of the optimal static mutation rate on LeadingOnes. Similarly, Lehre and Sudholt (2020) demonstrated that the (1+λ) EA using the optimal fitness-dependent mutation rate has an asymptotically optimal expected runtime on OneMax, i.e., the black-box complexity among all λ-parallel mutation-based black-box algorithms. However, this fitness-dependent adaptive parameter control method might be difficult to apply in real-world optimisation, since it usually heavily relies on knowledge of optimisation problems. That is, we need to know the relationship between search points and their objective values of the objective function to design a proper adaptation. Nonetheless, these studies provide valuable insights for designing more general adaptive parameter control methods.

Another type of approach involves adapting the mutation rate based on the "progress" of the previous generation. For instance, B. Doerr et al. (2019) proposed a self-adjusting mutation rate mechanism that results in the (1+λ) EA exhibiting a better asymptotic expected runtime on OneMax than that of the optimal fitness-dependent mutation rate. Using this self-adjusting mutation rate mechanism, in each generation, 50% of the offsprings are mutated with a mutation rate of $2\chi/n$, while the remaining 50% are mutated with a rate of $\chi/(2n)$, where χ/n represents the current generation mutation rate. Subsequently, the new mutation rate is determined by the mutation rate utilised by the best offspring. Simultaneously, B. Doerr et al. (2019, 2021) employed the fundamental concept of the $1/5^{\text{th}}$-success rule to control the mutation rate of the (1+1) EA. In their algorithm, the mutation rate is increased by multiplying a factor $A > 1$ if the offspring outperforms the parent; otherwise, it is decreased by multiplying a factor $0 < b < 1$. However, using the $1/5^{\text{th}}$-success rule to adjust the mutation rate could lead to a small mutation rate when trapped in local optima, since it is challenging to produce fitter individuals, which results in a continuous reduction

of the mutation rate. When optimising multi-modal functions, it is generally believed that a high mutation rate can be advantageous for escaping local optima.

To compensate for this disadvantage, Rajabi and Witt (2022) proposed a learning-inspired mechanism called *stagnation detection*, which increases the mutation rate after a specific number of iterations without improvement to enhance exploration. This number of iterations ensures that the neighbourhood has been thoroughly searched with a high probability. They incorporated the stagnation detection mechanism into the (1+1) EA algorithm and demonstrated that such an algorithm can optimise the well-studied multi-modal function $\textsc{Jump}_k$ with an asymptotic runtime comparable to that of the (1+1) EA using the optimal fitness-dependent mutation rate in (Böttcher et al., 2010). This mechanism might take too long to use larger mutation rates, which are beneficial for escaping local optima, and thus may slow down the optimisation process (B. Doerr and Rajabi, 2023).

Another area of interest is population size adaptation. Jansen et al. (2005) introduced the *multiplicative success-based rule* for adapting offspring population size, which also build on the idea of the $1/5^{\text{th}}$-success rule. In this approach, the offspring size λ is multiplied by F if there is no fitness improvement compared to the parent population, and multiplied by s otherwise, where $F \geq 1$ and $s \in (0, 1)$. The authors rigorously derived optimal offspring population sizes for the (1+λ) EA on benchmark functions, such as OneMax and LeadingOnes, suggesting that using this adaptive approach could yield comparable performance to employing the optimal offspring population size informed by their insights. Their following empirical study demonstrated that the adaptive (1+λ) EA with $F = 1/s = 2$ can be promising on some benchmark functions. Specifically, the number of function evaluations required to find the optimum was slightly higher, but still comparable to the best choices for λ observed in their earlier empirical studies. Later, the (1+λ) EA with this multiplicative success-based rule has been called the $(1 + \{F\lambda, \lambda/s\})$ EA (B. Doerr and C. Doerr, 2020; Hevia Fajardo, 2019). In (Hevia Fajardo and Sudholt, 2022), the authors proved that this

algorithm has comparable performance with static (1+λ) EA using any mutation rate when optimising any everywhere hard function.

Despite adapting the population size in elitist EAs, adapting the population size in non-elitist EAs is even more crucial. As mentioned, non-elitist EAs should balance the reproductive rate and the mutation rate. Therefore, parent and offspring population sizes should be set carefully in comma selection EAs, such as the $(1, \lambda)$ EA. A sharp threshold exists at $\lambda = \log_{\frac{e}{e-1}}(n)$ between exponential and polynomial runtimes on OneMax when using the standard mutation rate, i.e., $\chi/n = 1/n$ (Rowe and Sudholt, 2014). Hevia Fajardo and Sudholt (2021a) introduced a multiplicative success-based rule into a non-elitist EA to adapting population size, $(1, \lambda)$ EA, which they named $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA. This algorithm has three adaptive parameters: update strength $F \geq 1$, success rate $s > 0$, and the maximal population size $\lambda_{\max}$. In a generation where no improvement in fitness is found, λ is increased by a factor of $F^{1/s}$ and in a successful generation, λ is divided by a factor F. If one out of $s + 1$ generations is successful, the value of λ is maintained. They demonstrated in (Hevia Fajardo and Sudholt, 2021b) that this algorithm, incorporating the *reset mechanism* where λ is reset to $\lambda = 1$ whenever it exceeds a predefined maximum of $\lambda_{\max}$, can achieve the global optimum within $O(n)$ expected generations and $O(n \log n)$ runtime on a bi-modal function Cliff (Hevia Fajardo and Sudholt, 2021b). This results in an acceleration of order $\Omega(n^{2.9767})$ compared to the expected optimisation time when using the most suitable fixed value for λ. The reason for using this reset mechanism is to allow the offspring to "jump down" from local optima of the Cliff function with a high probability. This adaptation mechanism has also been proven beneficial for dynamic optimisation. More recently, Kaufmann et al. (2023) demonstrated that the $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA without the reset mechanism achieves the optimum of any dynamic monotone function, such as the dynamic linear function and the DBV function, in $O(n \log n)$.

Moreover, studies exist on adapting population size in GAs which involve crossover, such

as the $(1 + (\lambda, \lambda))$ GA (B. Doerr et al., 2015). In each generation, this algorithm employs two phases: In the first phase, the parent is mutated λ times with a mutation rate of χ/n. Next, the second phase uses a biased uniform crossover λ times between the parent and the fittest mutated offspring. This biased uniform crossover selects each bit independently at random from the fittest mutated offspring with crossover rate $c \in (0, 1)$ and from the parent otherwise. Finally, it performs an elitist selection considering the parent and the λ offspring obtained from the second phase. B. Doerr et al. (2015) suggested that using a relatively high mutation rate, i.e., $\chi/n = \lambda/n$, and a crossover rate $c = 1/\lambda$ can balance exploration and exploitation; however, the setting of population size becomes critical. To control the population size, they proposed a fitness-dependent adaptive mechanism and proved that updating the population size by $\lambda = \left\lceil \sqrt{n/(n - \mathrm{OM}(x))} \right\rceil$ where x is the current parent individual, with mutation rate $\chi/n = \lambda/n$ and crossover rate $c = 1/\lambda$, yields an expected optimisation time of $\Theta(n)$ on OneMax. This is asymptotically faster than any static parameter values. Similarly to the fitness-dependent adaptive mutation rate mechanism, this approach also requires knowledge of the objective function. Then, B. Doerr and C. Doerr (2018) applied the $1/5^{\text{th}}$-success rule to the $(1 + (\lambda, \lambda))$ GA, so-called self-adjusting $(1 + (\lambda, \lambda))$ GA, and proved that it requires also only $O(n)$ expected function evaluations on OneMax. This reduces the optimisation time of the algorithm by more than a constant factor compared to optimal static parameter settings. When optimising bi-modal functions, such as $\textsc{Jump}_k$, Hevia Fajardo and Sudholt (2023) demonstrated that the adaptive $(1 + (\lambda, \lambda))$ GA rapidly increases λ to $\lambda_{\max}$ when encountering a local optimum, even though the "optimal" population size to escape local optima varies depending on the jump size k. To make the algorithm more efficient in escaping local optima, the authors applied the reset mechanism to this adaptive $(1 + (\lambda, \lambda))$ GA, allowing the parameter to cycle through $[\lambda_{\max}]$ to attempt different population sizes. Their analysis revealed that the expected runtime of the self-adjusting $(1 + (\lambda, \lambda))$ GA with the reset mechanism is as low as that of the (1+1) EA with the optimal mutation rate.

2.3.2 Self-adaptation in EAs

Intuitively, an offspring produced by specific operators with suitable parameter settings can possess a higher potential to surpass its parent. Therefore, we can encode parameters as well as solutions in the genotype of each individual, allowing each individual to represent not only the solution but also its unique parameters. We can then design specific variation operators and selection mechanisms for these individuals to evolve their parameters and solutions simultaneously. This mechanism is known as self-adaptation, which was originally developed to control parameters in evolution strategies (ESs) for solving continuous optimisation problems in the 1980s (Schwefel, 1981), was later introduced to discrete optimisation. We refer readers to the thesis by Meyer-Nieberg (2007) for further information on self-adaptation in ES. In this section, we concentrate on EAs for optimising pseudo-Boolean functions. Within the scope of this research, we emphasise the theoretical study of self-adaptation, while also slightly extending to empirical study. Table 2.2 provides a summary of various studies on self-adaptive parameter control mechanisms in EC, inclusive of the research presented in this thesis.

Early studies on EAs with self-adaptation solving discrete optimisation problems concentrated on simple benchmark problems and employed empirical analysis. In 1992, Bäck (1992) first transferred the idea of self-adaptation from ESs to GAs. The mutation rate, represented as real numbers, was transformed into binary code and integrated with the solution in the bitstring of each individual, or genotype. During mutation, the binary-coded mutation rate of the genotype of an individual is first decoded to a real number. Next, a new binary-coded mutation rate is generated by bit-wisely mutating the original binary-coded mutation rate with a probability equivalent to the decoded real number. Finally, the new solution is created by bitwise mutation of the solution of the individual, using the mutation rate decoded from the newly generated binary-coded mutation rate. Thus, the offspring is created

Table 2.2: Research on self-adaptive parameter control mechanisms in EC

Reference	Algorithm	Self-adapted parameter(s)	Encoded method	Selection mechanism	Problem class
(Bäck, 1992)	GA	Mutation rate	Binary-encoded	Proportional selection	Continuous functions
(Smith and Fogarty, 1996)	GA	Mutation rate	Gray-coded Binary-encoded	Proportional selection	NK-Landscape
(Bäck and Schütz, 1996)	EA	Mutation rate	Real-valued	Proportional selection	Dynamic function
(Galaviz and Xuri, 1996)	GA	Mutation rate Crossover rate	Binary-encoded	Proportional selection	Continuous functions
(Bäck et al., 2000)	GA	Mutation rate Crossover rate	Real-valued	Elitist selection	Continuous functions
(Smith, 2001)	GA	Mutation rate	Finite set	Elitist selection	OneMax Dynamic function
(Eiben et al., 2006)	GA	Mutation rate Population size Selection parameter	Real-valued	Elitist selection	Continuous functions
(Serpell and Smith, 2010)	GA	Mutation rate Choice of mutation operator	Finite set	Elitist selection	TSP
(Dang and Lehre, 2016b)	EA	Mutation rate	Finite set (2 rates)	2-tournament selection	PeakedLO
(Case and Lehre, 2020)	EA	Mutation rate	Real-valued	(μ, λ)-selection	LeadingOnes$_k$
(B. Doerr et al., 2021)	EA	Mutation rate	Real-valued	$(1, \lambda)$-selection	OneMax
Chapter 4	EA	Mutation rate	Finite set (2 rates)	2-tournament selection	Noisy functions
Chapter 5	EA	Mutation rate	Real-valued	(μ, λ)-selection	Dynamic function
Chapter 6	EA	Mutation rate	Real-valued	(μ, λ)-selection	PeakedLO$_{m,k}$
Chapter 7	EA	Mutation rate	Real-valued	(μ, λ)-selection	NK-Landscape Max-k-Sat Noisy functions

with the new solution and the new mutation rate. The selection process is solely based on the fitness values of the individuals. The experiments conducted by Bäck (1992) illustrated that self-adaptation is a possible method to control the mutation rate in the non-elitist GA. However, the experiments did not include a comparison with other algorithms. Instead of encoding the mutation rate directly into a binary code, Smith and Fogarty (1996) argued that the use of Gray-coded mutation rates can provide a smoother change of mutation rates. Moreover, they demonstrated that GAs utilising the self-adaptive method can outperform those with fixed mutation rates when solving NK-Landscape problems, achieving higher

fitness within the evaluation budget. As introduced in Section 2.2.4, the landscape of NK-Landscape problems contains numerous local optima (Kauffman and Weinberger, 1989). Notably, the advantage of the self-adaptive algorithm in achieving higher fitness becomes more pronounced for harder (more local optima) problem instances of NK-Landscape. Bäck and Schütz (1996) further improved this self-adaptation method by using a real-valued mutation rate to address the imprecision of the binary-coded mutation rate. Consequently, the genotype of an individual became a combination of a bitstring and a real number, rather than just a bitstring, an encoding method similar to the one presented in Section 2.2.2. During mutation, a new mutation rate is generated by sampling from a distribution that is based on the previous value and a learning rate. The learning rate is a hyper-parameter in this self-adaptive EA that controls the speed of the adaptation process. They used a learning rate which was independent from the problem instance size n. An empirical study was conducted on the EA with this self-adaptive parameter control by minimising a dynamic function that periodically switches between two pseudo-Boolean functions, OneMax and ZeroMax. Here, $\textsc{ZeroMax}(x) := n - \textsc{OM}(x)$, given a bitstring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$. The results showed that the mutation rates can respond to changing of objective functions. Specifically, the mutation rates decrease if the fitness value decreases, and increase otherwise. Similarly, Smith (2001) made a similar observation where the mutation rate is self-adapted by selecting uniformly at random from a finite set of mutation rates. These empirical observations suggest that self-adaptation is capable of adapting the mutation rate in dynamic environments. In the context of more practical problems, Serpell and Smith (2010) investigated various self-adapting mutation rate mechanisms in the TSP. They demonstrated that a GA applying self-adaptation of the mutation rate, using a set of discrete mutation rates, yielded shorter mean paths compared to the best fixed-parameter GA.

In addition to self-adapting mutation rates, which is the primary focus of the aforementioned research, other parameters, such as the crossover rate, can also be adapted. Galaviz

and Xuri (1996) empirically investigated the performance of GAs with self-adapting mutation and crossover rates on several numerical functions and compared them to static GA. Mutation rates and crossover rates were encoded as binary codes, which were then incorporated into genotypes of individuals. They employed the same self-adapting mutation rate strategy as proposed by Bäck (1992). To self-adapt the crossover rate, they applied the crossover operator to two selected individuals with a probability equal to the average of their inherited crossover rates. They evaluated the algorithms on several binary-coded, two-dimensional continuous optimisation benchmark problems. Their findings revealed that GAs adapting both mutation and crossover rates can achieve fitness values comparable to static GAs with the most optimal settings, within the given budget of fitness evaluations. A subsequent study by Bäck et al. (2000) proposed a "parameterless" GA, which featured self-adapting mutation and crossover rates, as well as an adapting population size via the $1/5^{\text{th}}$-success-rule. From their experiments on binary-coded ten-dimensional continuous optimisation benchmark problems, they discovered that this "parameterless" GA could outperform other GAs with only one dynamic parameter. Eiben et al. (2006) explored self-adaptation of global parameters in GAs, specifically the population size and the selection parameter. They encoded values into the genotype of each individual, and the global parameters were determined by summing the local votes of all individuals. On all benchmarking functions in their experiments, the mean achieved fitness values for the GA with self-adapting selection parameter were statistically higher than those of other self-adaptive GAs and fine-tuned GAs.

Empirical studies cannot guarantee the correctness of analysis for randomised heuristic algorithms, as only a limited number of problem sizes can be explored in experiments. Theoretical analysis, such as runtime analysis introduced in Section 2.2.3, can compensate for this limitation. Moreover, the proofs underlying runtime analysis often yield a deeper understanding of the evolutionary process than experiments alone. However, it is usually challenging to achieve rigorous results even for simple algorithms on straightforward functions.

In analysing self-adaptive algorithms, the difficulty increases, as it involves not only the changing of solutions but also the dynamics of parameters. Therefore, compared to empirical studies, theoretical research on self-adaptation in discrete optimisation is less explored. Only a few theoretical studies on self-adaptation have emerged in the past decade. Dang and Lehre (2016b) first proved that the 2-tournament EA with self-adapting two mutation rates can perform effectively on the PeakedLO function, defined as

$$\textsc{PeakedLO}(x) := \begin{cases} m & \text{if } x = 0^n, \\ \mathrm{LO}(x) & \text{otherwise,} \end{cases} \tag{2.21}$$

where a bitstring $x \in \{0,1\}^n$ and $m \in \mathbb{R}$ for all $n \in \mathbb{N}$. Their algorithm utilises a fitness-only sorting partial order (Definition 2.2.4(a)) to sort the parent population before tournament selection and adjusts the mutation rate for each offspring individual by switching to an alternative rate with probability p_c. This probability serves as a self-adaptive strategy parameter for the algorithm. They employed the level-based theorem (Corus et al., 2018) to estimate the expected number of evaluations required for the 2-tournament EA with self-adaptive mutation rates to escape local optima and achieve the global optimum, assuming the initial population starts from the local optimum 0^n. As a comparison, they also provided the runtime of the 2-tournament EA using either a fixed mutation rate or a uniformly chosen mutation rate from two given rates for this problem. The study demonstrated that self-adaptation, with a sufficiently low self-adaptive strategy parameter p_c, can robustly control mutation rates in non-elitist EAs. Moreover, this automated control can lead to exponential speedups compared to EAs employing fixed mutation rates or uniform mixing of mutation rates. Although this rigorous analysis provided the first evidence of the benefits of self-adapting mutation rate mechanisms, it only involved two mutation rates.

Subsequent works on self-adapting mutation rates from a given interval were conducted by Case and Lehre (2020) and B. Doerr et al. (2021). Case and Lehre (2020) demonstrated that (μ, λ) self-adaptive EA (defined in Section 2.2.1) can be effective on the unknown structure

version of the LeadingOnes function:

$$\textsc{LeadingOnes}_k(x) := \sum_{i=1}^{k} \prod_{j=1}^{i} x_j \tag{2.22}$$

where a bitstring $x \in \{0,1\}^n$ and $k \in [n]$ for all $n \in \mathbb{N}$. Here, the term *structure* in this context refers to the number of relevant bit-positions, which are determined by the parameter k. Static mutation-only EAs, such as the (1+1) EA, require knowledge of k to set the "right" mutation rate for optimal performance, which corresponds to an asymptotic mutation rate of $\Theta(1/k)$ (Cathabard et al., 2011). In the context of unknown structure, the algorithm has black-box access to the function, meaning it is aware of n but not k. This algorithm employs (μ, λ) selection to choose individuals from the parent population, which is sorted according to a fitness-first sorting partial order (Definition 2.2.4(b)). Next, the mutation rates of selected individuals are self-adapted by multiplying with A with a probability of p_{inc} or multiplying with b otherwise, where self-adaptive parameters $A > 1$ and $p_{\text{inc}}, b \in (0,1)$. Finally, individuals in the offspring population are generated using a bitwise mutation operator with the updated mutation rate. Theoretical results have shown that the runtime of (μ, λ) self-adaptive EA is asymptotically optimal among all unary unbiased black-box algorithms (Cathabard et al., 2011). Their analysis partitioned the search space of the solution and mutation rate into two-dimensional levels. Following this, the level-based theorem (Corus et al., 2018) (presented in Theorem 2.2.1) was employed to estimate the runtime.

Another study on self-adapting mutation rates within an interval is by B. Doerr et al. (2021), who rigorously analysed a self-adaptation mechanism for the $(1, \lambda)$ EA. In this mechanism, the mutation rate χ/n is adjusted by either multiplying or dividing by a constant $F > 1$ (selected uniformly at random) before mutating the solution. They demonstrated that the algorithm optimises the OneMax function in an expected runtime $O(n \log(n))$, which is the best possible result for mutation-only EAs (Lehre and Witt, 2012). In the opinion of the author of this thesis, it is more appropriate to classify the algorithm as adaptive rather than

self-adaptive, despite the authors referring to it as self-adaptive in B. Doerr et al. (2021). This is because each generation contains only one parent, leading to a single parameter setting. Thus, the process of producing the next parent individual can be described without the context of self-adaptation: First, generate λ offspring using two mutation rates, $F\chi/n$ and $\chi/(Fn)$, selected uniformly at random to mutate the current parent individual. Then, select the fittest individual as the next parent and set the mutation rate to the one used for that individual. There is no need to "store" the mutation rate in the genotype of each individual. This approach is similar to the self-adjusting (adaptive) (1+λ) EA introduced in (B. Doerr et al., 2019).

Overall, although there are theoretical studies on self-adaptation demonstrating efficiency and effectiveness on various benchmarking functions and scenarios, empirical research also suggests that self-adaptation may be beneficial for dynamic optimisation and multi-modal functions. However, the theoretical investigation of these specific scenarios remains limited, such as uncertain environments.

2.3.3 EAs in Uncertain Environments

In many cases, obtaining the exact objective value of a search point is challenging, or the fitness function may change over time. As a result, a variety of *uncertainties* must be considered in real-world optimisation (Jin and Branke, 2005). While most studies on how EAs respond to uncertainties are empirical (Cruz et al., 2011; Jin and Branke, 2005; T. T. Nguyen et al., 2012), there is also a wealth of rigours analyses of EAs under noise. In uncertain environments, theoretical analyses of EAs have explored three types of uncertainty: noise, dynamic, and partial evaluation. This section primarily focuses on related work in noisy and dynamic optimisation. For partial evaluation, readers are referred to (Dang and Lehre, 2016a; B. Doerr et al., 2012). Through theoretical analysis, researchers aim to understand how uncer-

tainty affects the behaviour of EAs, such as runtime, and identify effective strategies and parameter settings for addressing uncertainty. Section 2.2.5 provided preliminaries related to noise and dynamics. Furthermore, Sections 2.3.3.1 and 2.3.3.2, will present summaries of notable theoretical works in the fields of noisy and dynamic optimisation, respectively.

2.3.3.1 Noisy Optimisation

In optimising noisy fitness functions, the algorithm cannot rely on obtaining exact objective values, as they may be affected by noise. Due to the difficulty of analysing EAs in the presence of noise, most existing theoretical studies have focused on two simple functions: OneMax and LeadingOnes under in noisy environments. As introduced in Section 2.2.5, the noise level is describes the degree of noise affecting the evaluation. Initial research demonstrated that several simple EAs can be robust to moderate noise levels but can become inefficient when faced with high levels of noise. For instance, the runtimes of (1+1) EA on LeadingOnes are polynomial if noise levels (descriptions of noise levels shown in Section 2.2.5) are $q \in O(\log(n)/n^2)$ and $p \in O(\log(n)/n^3)$ under one-bit noise and bit-wise noise, respectively (Sudholt, 2021). In contrast, runtimes become exponential if noise levels are $q \in \Omega(1/n)$ and $p \in \Omega(1/n^2)$ (Sudholt, 2021).

A common method to cope with high levels of uncertainty is using a resampling strategy which averages the value of many uncertain evaluations (Qian et al., 2019; Qian et al., 2018). The (1+1) EA using a resampling strategy solves OneMax and LeadingOnes in high-level one-bit, bit-wise and Gaussian noise models in expected polynomial time (Qian et al., 2019; Qian et al., 2018). For instance, the runtime of the (1+1) EA using a resampling strategy remains polynomial on LeadingOnes under the one-bit noise model with any noise level $q \in [0, 1]$. More specifically, this algorithm guarantees a runtime of $O(mn^8)$, if $m = 4n^4 \log(n)/15$, where sample size m represents the parameter defining the number of evaluations for each

individual (Qian et al., 2019). However, the drawback of using a resampling strategy is that it may "waste" numerous evaluations if the noise level is small or even non-existent. For example, the runtime of the original (1+1) EA is $O(n^2)$ on LeadingOnes in the one-bit noise model with $q \in O(1/n^2)$ (Sudholt, 2021). Therefore, the algorithm employing a resampling strategy requires prior knowledge of the noise level to set appropriate parameters for optimal performance. Furthermore, the resampling strategy cannot discern the fitter of two search points if the expected noisy fitness values are identical, regardless of the sample size settings. For instance, Qian et al. (2021) demonstrated that the (1+1) EA fails in the symmetric noise model, where the fitness value of a search point x is $C - \mathrm{OM}(x)$ with probability $q = 1/2$ and $\mathrm{OM}(x)$ otherwise, with $C \in \mathbb{R}$ (a particular case of Definition 2.2.15). In this case, the expected noisy fitness value of any search point is always C.

Employing a population is another common approach to address noise. With respect to the aforementioned symmetric noise model, where the resampling strategy encounters difficulties, Qian et al. (2021) demonstrated that the (μ+1) EA and the (1+λ) EA can solve the OneMax problem in the symmetric noise model within an expected polynomial time. Also, using a population, the algorithm can tolerate higher levels of noise compared with single-individual EAs. For instance, Gießen and Kötzing (2016) demonstrated that the runtime of the (1+λ) EA on LeadingOnes under one-bit noise with $[0, 1/2]$ is $O(n^2)$, provided that the population size is $\lambda \geq 3.42 \log(n)$ and $\lambda \in O(n)$. Utilising a non-elitist population has also proven to be efficient under noise. Dang and Lehre (2015) demonstrated that 2-tournament EAs with a sufficiently large population size and a conservative mutation rate yields an expected runtime of $O(n \log(n) \log(\log(n)))$ on OneMax under any one-bit noise level. Additionally, they proved that the non-elitist EA can handle extremely high levels of Gaussian noise (Dang and Lehre, 2015). As mentioned in Section 2.2.1, the parameter setting, particularly the mutation rate, is crucial for non-elitist EAs. Their analysis solely considers a specific, sufficiently small mutation rate to cope with noise. However, the results

of the relationship for non-elitist EAs between the mutation rate, noise level, population size, and runtime are currently incomplete. Gaining insight into how noise affects the algorithms is valuable. More importantly, understanding this relationship is essential for setting appropriate parameters to achieve the highest possible performance.

Concerning the focus of this thesis, the robustness of non-elitist EAs with self-adaptation remains uncertain in noisy environments. Although Dang and Lehre (2015) did not provide precise guidance on setting the mutation rate under noise, they recommended using a small mutation rate. Based on this, we believe controlling the mutation rate is necessary in noisy environments. Exploring self-adapting mutation rates appears to be a promising avenue.

Several algorithms, somewhat broader in scope than EAs, can also handle noise. These include estimation of distribution algorithms (EDAs) (Friedrich et al., 2016), ant colony optimisation (ACO) (Friedrich et al., 2016), and the voting algorithm (Aishwaryaprajna and Rowe, 2023; Rowe and Aishwaryaprajna, 2019). Although these results are not included in the discussion of this thesis, we refer to Tables 2.3-2.10 for more details. These tables summarise recent theoretical studies of randomised heuristic algorithms in noisy settings (including those obtained in this thesis). Tables 2.3 and 2.4 show results for the one-bit noise model (q) on OneMax and LeadingOnes, respectively. Table 2.5 and 2.6 show results for the bit-wise noise model (p) on OneMax and LeadingOnes, respectively. Table 2.7 and 2.8 show results for the Gaussian noise model (σ) on OneMax and LeadingOnes, respectively. Tables 2.9 and 2.10 show results for the symmetric noise model (C, q) on OneMax and LeadingOnes, respectively. Note that the previous studies in (Qian et al., 2021; Qian et al., 2018) do not provide exact runtimes. In these cases, the runtime results are deduced from proofs.

Table 2.3: Theoretical results of randomised heuristic algorithms on OneMax in the one-bit noise model (q)

Algorithm	**Parameter settings**	**Noise level q**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$O(1/n)$	$\Theta(n \log(n))$ (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$O(\log(n)/n)$	poly(n) (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$\omega(\log(n)/n)$	$2^{\omega(\log(n))}$ (Gießen and Kötzing, 2016)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = 4n^3$	$[0, 1]$	poly(n) (Qian et al., 2019)
(μ+1) EA	$\chi/n = 1/n$; $\mu > 12\log(15n)/q$	$(0, 1]$	$O\left(\mu n \log(n)\right)$ (Gießen and Kötzing, 2016)
(1+λ) EA	$\chi/n = 1/n$; $\lambda \geq \max\{12/q, 24\} n \log n$	$(0, 1]$	$O\left(\left(n^2 \log(n) + n^2\lambda\right)/q\right)$ (Gießen and Kötzing, 2016)
ACO-fp	$\rho = \mathrm{o}\left(1/\left(n^3 \log n\right)\right)$	$[0, 1]$	$O\left(n^2 \log(n)/\rho\right)$ (Friedrich et al., 2016)
2-tournament EA	$\chi/n = a/n$; $\lambda = b\log(n)$	$[0, 1]$	$O\left(n \log(n) \log\left(\log(n)\right)\right)$
	for some constants $a, b > 0$	$[0, 1]$	(Dang and Lehre, 2015)
	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\log(n/\chi))$	$[0, 1]$	$O\left(\lambda n \log(1/\chi) + n \log(n)/\chi\right)$
	$\theta = 1/2 - (q/2)(1 - q/2) - q/2n_0$,		(Theorem 3.3.1)
	for any constant $n_0 \geq 3$		

Table 2.4: Theoretical results of randomised heuristic algorithms on LeadingOnes in the one-bit noise model (q)

Algorithm	**Parameter settings**	**Noise level q**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$[0, 1/2]$	$\Theta\left(n^2\right) e^{\Theta\left(\min\left\{qn^2, n\right\}\right)}$ (Sudholt, 2021)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = 4n^4 \log(n)/15$	$[0, 1]$	$O(mn^8)$ (Qian et al., 2019)
(1+λ) EA	$\chi/n = 1/n$; $\lambda \geq 3.42 \log(n)$, $\lambda \in O(n)$	$[0, 1/2]$	$O\left(n^2 e^{O(qn/\lambda)}\right)$ (Sudholt, 2021)
UMDA	$\mu \geq c \log(n)$; $\lambda > 4e(1+\delta)\mu$ for constants c, δ;	$[0, 1)$	$O(n\lambda \log(\lambda) + n^2)$ (Lehre and P. T. H. Nguyen, 2021)
2-tournament EA	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\log(n/\chi))$	$[0, 1)$	$O\left(n\lambda \log(n/\chi) + n^2/\chi\right)$
	$\theta = 1/2 - q(1-q/2)$		(Theorem 3.3.2)

Table 2.5: Theoretical results of randomised heuristic algorithms on OneMax in the bit-wise noise model (p)

Algorithm	**Parameter settings**	**Noise level p**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$O(1/n^2)$	$\Theta(n \log(n))$ (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$O(\log(n)/n^2)$	$\text{poly}(n)$ (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$\omega(\log(n)/n^2)$	$2^{\omega(\log(n))}$ (Gießen and Kötzing, 2016)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = n^{3+2c}/4$	$p = 1/2 - 1/n^c$ for $0 < c = \Theta(1)$	$\text{poly}(n)$ (Qian et al., 2019)
2-tournament EA	$\chi/n \in (0, \ln(1+2\theta)/n)$ $\theta = 9(1/2-p)/\left(64\sqrt{2pn}+16\right)$; $\lambda \in \Omega\left(\frac{1+pn}{(1-2p)^2}\log\left(\frac{n}{(1-2p)\chi}\right)\right)$	$(0, 1/2)$	$O\left(\frac{n(1+pn)}{(1-2p)^2}\left(\lambda \log\left(\frac{1}{\chi}\right)+\frac{\log(n)}{\chi}\right)\right)$ (Theorem 3.3.3)

Table 2.6: Theoretical results of randomised heuristic algorithms on LeadingOnes in the bit-wise noise model (p)

Algorithm	**Parameter settings**	**Noise level p**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$[0, 1/(2n)]$	$\Theta\left(n^2\right) e^{\Theta\left(\min\left\{pn^3, n\right\}\right)}$ (Sudholt, 2021)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = 36n^{2c+4}$	$p = c\log(n)/n$ for $0 < c = \Theta(1)$	$12m \cdot n^{30c+1}$ (Qian et al., 2019)
	$\chi/n = 1/n$; $m \in O(\text{poly}(n))$	$\omega(\log(n)/n)$	$e^{\Omega(n)}$ (Qian et al., 2019)
2-tournament EA	$\chi/n \in (0, \ln(1+2\theta)/n)$ $\theta = (1/2 - p)\, e^{-3np}$; $\lambda \in \Omega\left(\frac{e^{6np}}{(1-3p)^2} \log\left(\frac{n}{\chi}\right)\right)$	$[0, 1/3)$	$O\left(\frac{ne^{6np}}{(1-3p)^2}\left(\lambda \log\left(\frac{n}{\chi}\right) + \frac{n}{\chi}\right)\right)$ (Theorem 3.3.4)

Table 2.7: Theoretical results of randomised heuristic algorithms on OneMax in the Gaussian noise model (σ^2)

Algorithm	**Parameter settings**	**Noise level σ**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$\sigma^2 \leq 1/(4\log(n))$	$O(n\log(n))$ (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$\sigma^2 \geq 1$	$e^{\Omega(n)}$ (Qian et al., 2018)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = \lceil \frac{n\sigma^2}{\log(n)} \rceil$	$1 \leq \sigma^2 \in \text{poly}(n)$	poly(n) (Qian et al., 2018)
$(\mu+1)$ EA	$\chi/n = 1/n$; $\mu \in \text{poly}(n)$	$\sigma \geq n^3$	$2^{\omega(\log(n))}$ (Friedrich et al., 2015)
cGA	$K = \omega(\sigma^2\sqrt{n}\log(n))$	$\sigma^2 > 0$	$O(K\sigma^2\sqrt{n}\log(Kn))$ (Friedrich et al., 2016)
ACO-fp	$\rho = \text{o}(1/(n(n+\sigma\log n)^2\log n))$	$\sigma^2 \geq 0$	$O(n^2\log(n)/\rho)$ (Friedrich et al., 2016)
Voting Algorithm	-	$\sigma^2 \geq 3n/8$	$O(\sigma^2\log(n))$ (Rowe and Aishwaryaprajna, 2019)
2-tournament EA	$\chi/n = a/(n\sigma)$; $\lambda = b\sigma^2\log(n)$	$\sigma^2 \geq 0$	$O(\sigma^6 n\log(n)\log(\log(n)))$
	for some constants $a, b > 0$	$[0,1]$	(Dang and Lehre, 2015)
	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\sigma^2\log(n/\chi))$	$\sigma^2 > 0$	$O\big(\sigma^2\lambda n\log(1/\chi) + \sigma^2 n\log(n)/\chi\big)$
	$\theta = 1/(6+48\sigma/\pi)$		(Theorem 3.3.5)

Table 2.8: Theoretical results of randomised heuristic algorithms on LeadingOnes in the Gaussian noise model (σ^2)

Algorithm	**Parameter settings**	**Noise level σ**	**Expected runtime**
(1+1) EA	$\chi/n = 1/n$	$\sigma^2 \leq 1/(12en^2)$	$O(n^2)$ (Gießen and Kötzing, 2016)
	$\chi/n = 1/n$	$\sigma^2 \geq n^2$	$\Omega(e^n)$ (Qian et al., 2018)
(1+1) EA (resampling)	$\chi/n = 1/n$; $m = \lceil 12en^2\sigma^2 \rceil$	$n^2 \leq \sigma^2 \in \text{poly}(n)$	$O(mn^2)$ (Qian et al., 2018)
2-tournament EA	$\chi/n = a/(n\sigma)$; $\lambda = b\sigma^2 \log(n)$	$\sigma^2 \geq 0$	$O(\sigma^7 n \log(n) + \sigma^6 n^2)$
	for some constants $a, b > 0$	$[0, 1]$	(Dang and Lehre, 2015)
	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\sigma^2 \log(n/\chi))$	$\sigma^2 > 0$	$O(\sigma^2 \lambda n \log(n/\chi) + \sigma^2 n^2/\chi)$
	$\theta = 1/(6 + 48\sigma/\pi)$		(Theorem 3.3.5)

Table 2.9: Theoretical results of randomised heuristic algorithms on OneMax in the symmetric noise model (C, q)

Algorithm	**Parameter settings**	**Noise parameters**	**Expected runtime**
(1+1) EA (resampling)	$\chi/n = 1/n$; Any $m \geq 1$	Any $C \in \mathbb{R}$; $q = 1/2$	$e^{\Omega(n)}$ (Qian et al., 2021)
$(\mu + 1)$ EA	$\chi/n = 1/n$; $\mu = 3\log(n)$	$C = 2n$; $q = 1/2$	$O\left(n\log^3(n)\right)$ (Qian et al., 2021)
$(1 + \lambda)$ EA	$\chi/n = 1/n$; $\lambda = 8\log(n)$	$C = 0$; $q = 1/2$	$O\left(n\log^2(n)\right)$ (Qian et al., 2021)
	$\chi/n = 1/n$; $\lambda \leq \log(n)/10$	$C = 0$; $q = 1/2$	$e^{\Omega(n)}$ (Qian et al., 2021)
2-tournament EA	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\log(n/\chi))$ $\theta = 1/2 - q$	$C \in \mathbb{R}$; $q \in [0, 1/2)$	$O\left(\lambda n\log(1/\chi) + n\log(n)/\chi\right)$ (Theorem 3.3.6)
	$\chi/n > \ln(1+2\theta+o(1))/n$; $\lambda \in \text{poly}(n)$ $\theta = 1/2 - q$	$C \in \mathbb{R}$; $q \in [0, 1/2)$	$e^{\Omega(n)}$ (Theorem 3.3.8)
(μ, λ) EA	$\lambda \in \Omega(\log(n))$; $(1-q)\zeta\lambda > \mu \in \Omega(\log(n))$ $\chi/n \in \left(0, \ln\left(\frac{(1-q)\lambda}{(1+\delta)\mu}\right)/n\right)$ and $\chi \in \Omega(1)$ for any constant $\zeta, \delta \in (0,1)$	$C \leq 0$; $q \in [0, 1)$	$O\left(\lambda n + n\log(n)\right)$ (Theorem 3.3.7)
	$\lambda, \mu \in \log(n)$; $(1-q)\lambda > \mu$ $\chi/n > \ln\left(\frac{(1-q)\lambda}{\mu}\right)/n$	$C \leq 0$; $q \in [0, 1]$	$e^{\Omega(n)}$ (Theorem 3.3.9)

Table 2.10: Theoretical results of randomised heuristic algorithms on LeadingOnes in the symmetric noise model (C, q)

Algorithm	**Parameter settings**	**Noise parameters**	**Expected runtime**
2-tournament EA	$\chi/n \in (0, \ln(1+2\theta)/n)$; $\lambda \in \Omega(\log(n/\chi))$	$C \in \mathbb{R}$; $q \in [0, 1/2)$	$O\left(n\lambda \log\left(n/\chi\right) + n^2/\chi\right)$
	$\theta = 1/2 - q$		(Theorem 3.3.6)
	$\chi/n > \ln(1+2\theta+o(1))/n$; $\lambda \in \text{poly}(n)$	$C \in \mathbb{R}$; $q \in [0, 1/2)$	$e^{\Omega(n)}$ (Theorem 3.3.8)
	$\theta = 1/2 - q$		
(μ, λ) EA	$\lambda \in \Omega(\log(n))$; $(1-q)\zeta\lambda > \mu \in \Omega(\log(n))$	$C \leq 0$; $q \in [0, 1)$	$O\left(n\lambda \log\left(n\right) + n^2\right)$
	$\chi/n \in \left(0, \ln\left(\frac{(1-q)\lambda}{(1+\delta)\mu}\right)/n\right)$ and $\chi \in \Omega(1)$		(Theorem 3.3.7)
	for any constant $\zeta, \delta \in (0, 1)$;		
	$\lambda, \mu \in \log(n)$; $(1-q)\lambda > \mu$	$C \leq 0$; $q \in [0, 1]$	$e^{\Omega(n)}$ (Theorem 3.3.9)
	$\chi/n > \ln\left(\frac{(1-q)\lambda}{\mu}\right)/n$		

2.3.3.2 Dynamic Optimisation

In dynamic optimisation problems, the objective function changes over time. Many rigorous analyses of EAs and other randomised search heuristics in dynamic environments have been published in the previous two decades. Table 2.11 summarises existing theoretical works on dynamic problems in EC. These studies can be categorised into three types, each fulfilling the criteria outlined in Definition 2.2.16. The first type of research aims to evaluate the performance of algorithms optimising a dynamic function with randomly changing optima. The criteria can be the number of evaluations when the algorithm first hits the current optima. The second type of research is to analyse the runtime of algorithms on a dynamic function with a global and unique optimum. The third type of study examines the ability to track dynamic optima. This type of problem has a sequence (path) of optima and the optimum changes over time. The algorithms need to follow the path and find and hold the current optimal solutions before the next change; otherwise, they will get lost soon.

At the early stage of the theoretical analysis of EAs applied to dynamic optimisation, Droste (2002, 2003) examined a simple EA, the (1+1) EA, on the dynamic version of ONEMAX. In this function, each iteration generates a new optimum by flipping the last optimal bitstring bit-wisely with probability q. They proved that the (1+1) EA "catches" an optimal solution in polynomial time if and only if $q = O(\log(n)/n^2)$. Subsequently, Kötzing et al. (2015) broadened the analysis from bitstrings to larger alphabets. It was demonstrated that the number of values per dimension does not impact the performance of the (1+1) EA. Rohlfshagen et al. (2009) illustrated some counter-intuitive scenarios that provide insights into how the dynamics of a function can influence the runtime of this simple algorithm. Specifically, they introduced the function MAGNITUDE, where the relocation time of the global optimum for the (1+1) EA is less than $n^2 \log n$ with high probability when the magnitude of change is large, indicating efficiency. However, when the magnitude of change is

small, the expected time to relocate the global optimum becomes $e^{\Omega(n)}$, pointing to significant inefficiency. Similarly, the expected runtime of the (1+1) EA on the BALANCE function is $O(n^2)$, signifying efficiency, for high change frequencies, and is $n^{\Omega(\sqrt{n})}$, implying severe inefficiency, for low change frequencies. These findings contribute to a deeper comprehension of dynamic optimisation.

Using a population can enhance the capability of EAs in dynamic optimisation. In a pioneering work, Jansen and Schellbach (2005) conducted an analysis on a population-based algorithm, the (1+λ) EA, though it was applied to a straightforward ONEMAX-variant problem in a two-dimensional lattice. More recently, Dang et al. (2017) introduced a class of dynamic optimisation problems to investigate the role of populations in dynamic environments. They proved that the (1+1) EA and the RLS lose the optimal solution region with constant probability at any generation, whereas the non-elitist population-based EAs remain within the optimal region for a long time with probability $1 - e^{-\Omega(n^{\varepsilon})}$, where $c, \varepsilon > 0$ are constants (Dang et al., 2017).

In certain dynamic scenarios, only using a population might be insufficient, and the inclusion of other mechanisms such as diversity mechanisms and parallelisation could be beneficial. For instance, with the BALANCE function, Oliveto and Zarges (2013) rigorously demonstrated that the original (μ+1) EA fails, yielding an exponential runtime, if the frequency of change is low enough. However, a (μ+1) EA that employs a fitness-diversity mechanism can guarantee a polynomial expected runtime irrespectively of the frequency. In another example, Kötzing and Molter (2012) proposed the MAZE function where the optimum switches from 1^n to 0^n with n steps of change. In this scenario, each transition modifies a single bit from the preceding "optimal" bitstring, and the interval between two successive changes is $(kn^3 \log(n))$-generations, given a constant $k > 0$. The (1+1) EA fails to keep track of and reach the final optima. On the other hand, the parallel (1+1) EA can successfully trace the MAZE function (Lissovoi and Witt, 2017). Beyond EAs, other randomised

heuristic algorithms also hold promise for solving dynamic optimisation problems. To illustrate, Kötzing and Molter (2012) established that a simplified version of the Max-Min Ant System (MMAS) ant colony optimisation (ACO) algorithm can effectively track and reach the final optima of the Maze function.

As discussed in Section 2.3.2, the runtime of EAs on objective functions with different structures is significantly affected by the mutation rate used. However, there is currently no existing study that investigates the sequence of objective functions in dynamic optimisation with varied structures. Additionally, as discussed in Section 2.3.2, two empirical studies (Bäck and Schütz, 1996; Smith, 2001) have demonstrated that self-adaptation can respond to changes in fitness functions, such as switching between OneMax and ZeroMax. Therefore, using a self-adaptive parameter control mechanism to adapt the mutation rate under this kind of dynamic is promising. However, the full extent of the advantages of self-adaptation in dynamic optimisation problems is still not fully understood.

2.3.4 EAs on Multi-modal Landscapes

Fitness landscapes offer an abstract way to express the relationship between search points and their corresponding fitness on fitness functions (Malan, 2021). The concept of a fitness landscape was first introduced by (Wright, 1932). Wright depicted this idea through an abstract, two-dimensional contour plot of fitness values, providing an intuitive illustration of the evolutionary processes occurring in a high-dimensional space. Moreover, the application of fitness landscapes has extended beyond the realm of biological evolution, encompassing areas such as computational evolution, including EAs. A fitness landscape is defined as comprising three elements (Stadler, 2002): (1) a set S_z of solutions (search points) for the problem instance z; (2) a definition of distance (neighbourhood) on S_z, and (3) a fitness function $f : S_z \to \mathbb{R}$. Replacing f with a more general objective function, these three

Table 2.11: Summary of theoretical studies of randomised search heuristics on dynamic optimisation

Type of Dynamics	**Problem**	**Algorithm**	**Study**
Optimising dynamic function with randomly changed optima	Dynamic OneMax (Droste, 2002, 2003)	(1+1) EA	(Droste, 2002, 2003)
	Generalised Dynamic OneMax (Kötzing et al., 2015)	(1+1) EA	(Kötzing et al., 2015)
	OneMax-variant in 2D lattice (Jansen and Schellbach, 2005; Weicker, 2005)	(1+λ) EA	(Jansen and Schellbach, 2005)
Optimising dynamic function with a global optimum	Magnitude (Rohlfshagen et al., 2009)	(1+1) EA	(Rohlfshagen et al., 2009)
	Balance (Rohlfshagen et al., 2009)	(1+1) EA	(Rohlfshagen et al., 2009)
		(μ+1) EA	(Oliveto and Zarges, 2013)
		(μ+1) EAs with diversities	(Oliveto and Zarges, 2013)
		(2+1) RLS with diversity	(Oliveto and Zarges, 2013)
	Noisy linear function (Lengler and Schaller, 2018)	(1+1) EA	(Lengler and Schaller, 2018)
	Dynamic BinVal (DBV) (Lengler and Meier, 2020, 2022)	(μ+1) EA	(Lengler and Riedi, 2022)
		2-tournament EA	(Lehre and Qin, 2022a)
Tracking dynamic optima	Maze (Kötzing and Molter, 2012)	(1+1) EA	(Kötzing and Molter, 2012)
		MMAS	(Kötzing and Molter, 2012)
		(2+1) EA	(Lissovoi, 2016)
		(1+λ) EA	(Lissovoi and Witt, 2017)
		Parallel (1+1) EA	(Lissovoi and Witt, 2017)
	Finite-alphabet Maze (Lissovoi, 2016)	(μ+1) EA with diversity	(Lissovoi, 2016)
		MMAS*	(Lissovoi, 2016)
	Dynamic shortest path (Lissovoi and Witt, 2015)	λ-MMAS	(Lissovoi and Witt, 2015)
	(κ, ρ)-stable dynamic function, e.g., Moving Hamming Ball (MHB) (Dang et al., 2017)	(1+1) EA	(Dang et al., 2017)
		Non-elitist EAs	(Dang et al., 2017)
	Dynamic matching substring (DSM) (Definition 5.2.1)	Static mutation-based EAs	(Theorem 5.5.1)
		(μ, λ) self-adaptive EA	(Theorem 5.4.1)

basic elements can be used to describe landscapes in a wide range of contexts such as combinatorial optimisation. In the realm of fitness landscapes for optimising pseudo-Boolean functions using mutation-only EA, Hamming distance is typically utilised. This is because the application of Hamming distance as a metric in the fitness landscape provides a direct mechanism to traverse the search space. In mutation-only EAs, a bit-wise mutation equates to a transition to a neighbouring point in the Hamming shell.

The class of multi-modal functions comprises landscapes with locally optimal search point, where a local optima search point, denoted as $\bar{x}$, satisfies the condition $f(\bar{x}) \geq f(x)$ for all $x \in \mathrm{N}_k(\bar{x})$. Here, $\mathrm{N}_k(\bar{x})$ denotes the set of neighbouring search points. Nonetheless, $\bar{x}$ is still inferior to the optimal solution x^*, i.e., $f(\bar{x}) < f(x^*)$. To escape local optima, one possible method involves using a bitwise mutation operator to directly "jump" from local optima to search points with higher fitness. This is achieved by flipping multiple bits simultaneously. As an illustration, by setting the mutation rate $\chi/n = k/n$ in the (1+1) EA, an optimal expected runtime of $O\left(\frac{n^k}{k^k}\left(\frac{n}{n-k}\right)^{n-k}\right)$ can be achieved on the $\textsc{Jump}_k$ function, which is bimodal as outlined in Eq. (2.11) (B. Doerr et al., 2017). To determine this optimal mutation rate, the mechanism of stagnation detection (Rajabi and Witt, 2022), discussed in Section 2.3.1.2, has been proven to be helpful. However, longer optimisation time may be necessitated if the jump distance k is extensive. For example, the runtime of the (1+1) EA algorithm could increase exponentially if $k = \Theta(n)$ on the $\textsc{Jump}_k$ function.

The application of a crossover operator can potentially shorten the expected time required to escape local optima, a proposition initially established by Jansen and Wegener (2002). Following this work, more detailed analysis reveals that the $(\mu + 1)$ GA with crossover only requires time $O(n^3)$ if the jump size $k = 4$ on $\textsc{Jump}_k$ (Dang et al., 2018).

Contrarily, the (1+1) EA with the optimal mutation rate $\chi/n = k/n$ guarantees a runtime of $O(n^4)$. Regarding another GA, mentioned in Section 2.3.1.2, the $(1 + (\lambda, \lambda))$ GA guaran-

tees a $O(n^{2.5})$ runtime in this specific case (Antipov et al., 2022). Nevertheless, parameter tuning is necessary. Hevia Fajardo and Sudholt (2023) demonstrated that employing an adaptive population size in the $(1 + (\lambda, \lambda))$ GA algorithm, in conjunction with a reset mechanism, can effectively assist in configuring the parameter settings on optimising JUMP_k, as related to the discussion in Section 2.3.1.2. This self-adjusting $(1 + (\lambda, \lambda))$ GA guarantees a runtime as low as the (1+1) EA with the optimal mutation rate.

Recently, it has been proven that non-elitism can facilitate the escape from local optima. Non-elitist EAs can "jump" a large Hamming distance. But they can potentially also maintain less fit individuals in the population, allowing the population to cross a *fitness valley*, which is a set of search points surrounded by higher fitness value search points. They might keep some currently low but potentially high fitness individuals in the population and optimise them "smoothly". A tunable problem class SPARSELOCALOPT was proposed to describe a kind of fitness landscapes with *sparse deceptive regions* (sparse local optima) and *dense fitness valleys* (Dang et al., 2021b). Informally, every search point in a dense set has many neighbours in that set, and every search point in a sparse set has few members in any direction. The formal definition of the SPARSELOCALOPT problem class is given in Definitions 1-4 in (Dang et al., 2021b). Dang et al. (2021b) show that EAs with a non-linear selection and a sufficiently high mutation rate, i.e., close to the error threshold, can cope with sparse local optima. Non-linear selection is a type of non-elitist selection, in which the probability of each individual to be selected is based on its rank in the population, e.g., tournament selection. Typically, the fitter individual has a higher probability to be selected, but the worse individual still has some chance to be chosen. From their analysis, non-linear selection and sufficiently high mutation rates can limit the percentage of "sparse" local optimal individuals in the population by choosing a sufficiently high mutation rate. The reason is that the sparse local optimal individuals can have a higher chance to be selected but can only survive a small percentage of such individuals after mutation, while the dense fitness valley individuals may have less

chance of being selected but can have higher chance of surviving mutation. However, the performance of the non-elitist EA depends critically on choosing the "right" mutation rate, which should be sufficiently high but below the error threshold. Moreover, finding such a mutation rate might be difficult or infeasible for some problem instances with not too sparse local optima.

Chapter Three

Fixed Parameter Settings in Uncertain Environments

Authors: Per Kristian Lehre and Xiaoyu Qin

This chapter is based on the following publications:

More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments (Lehre and Qin, 2021, 2022a) which are published in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'21) and Algorithmica (2022).

3.1 Introduction

Real-world applications often involve "uncertain" objectives, i.e., where optimisation algorithms observe objective values as a random variables with positive variance. In the past decade, several rigorous analysis results for evolutionary algorithms (EAs) on discrete problems show that EAs can cope with low-level uncertainties, i.e. when the variance of the uncertain objective value is small. Previous work showed that a large population combined with a non-elitist selection mechanism is a promising approach to handle high levels of uncertainty. However, the population size and the mutation rate can dramatically impact the performance of non-elitist EAs, and the optimal choices of these parameters depend on the level of uncertainty in the objective function. The performance and the required parameter settings for non-elitist EAs in some common objective-uncertainty scenarios are still unknown.

In this chapter, we theoretically analyse the runtime of non-elitist EAs with 2-tournament and (μ, λ) selection in several uncertain settings. For the 2-tournament EA, we first use the level-based theorem (Corus et al., 2018) to derive a general theorem in uncertain environments. Then we apply this general theorem to obtain upper bounds of runtimes on OneMax and LeadingOnes in prior and posterior noise models, i.e., one-bit, bit-wise, Gaussian and symmetric noise models. In noisy settings, our analyses are more extensive and precise than previous analyses of non-elitist EAs (Dang and Lehre, 2015). We provide more precise guidance on how to choose the mutation rate and the population size as a function of the level of uncertainty. We also use the negative drift theorem for populations (Lehre, 2010) to show that a too high mutation rate relative to the noise level leads to an exponentially low probability to find the optimum within exponential time. For the (μ, λ) EA, we analyse the runtime under symmetric noise for the first time. Similarly, we provide guidance on how to choose the mutation rate, the selective pressure and the population size as a function of the

noise level. We also show that too low selective pressure, i.e., low reproductive rate λ/μ, and too high mutation rate according to the noise level lead to inefficient optimisation. Overall, in several noisy settings, we prove that non-elitist EAs outperform the current state of the art results (see Tables 2.3-2.10). Finally, we prove for the first time that non-elitist EAs can optimise the DBV problem in expected polynomial time.

The chapter is structured as follows: Section 3.2 provides a general theorem for analysing non-elitist EAs (Algorithm 3) with 2-tournament selection (Algorithm 5) in uncertain environments. In Section 3.3, we consider four noise models. We prove that the expected runtime of the 2-tournament EA on OneMax and LeadingOnes under one-bit, bit-wise and Gaussian noise are polynomial for appropriate parameter settings in Sections 3.3.1, 3.3.2 and 3.3.3, respectively. In Section 3.3.4, we show that non-elitist EAs with 2-tournament and (μ, λ) selection (Algorithm 6) can find the optimum of OneMax and LeadingOnes under symmetric noise in expected polynomial time if using appropriate parameter settings, otherwise runtimes are exponential. Section 3.4 then shows the runtime analysis for the 2-tournament EA on the DBV function. Finally, Section 3.5 concludes the chapter.

3.2 2-tournament EA in Uncertain Environments

In this section, we introduce a general result (Theorem 3.2.1) which is an upper bound of the expected runtime of the 2-tournament EA in uncertain environments. The key step is to estimate the probability of the "real" fittest individual being selected from x_1 and x_2 in line 3 of Algorithm 5. In an uncertainty-free case, the fittest individual is selected with probability 1. In uncertain environments, condition (C2) of Theorem 3.2.1 is satisfied if the probability that the truly fitter individual is selected is greater than 1/2. We call this probability minus 1/2 *fitness bias*. This property of an uncertain problem decides how small the mutation

rate should be set, how large the population size should be and how fast the algorithm can achieve the optimum. We summarise fitness biases in some noisy scenarios in Lemma 3.2.1. Note that the concept of fitness bias only describes a property of an uncertainty model for 2-tournament selection.

The general theorem for the 2-tournament EA is derived from the level-based theorem (Theorem 2.2.1). Condition (C1) is to estimate a lower bound of the probability of "level upgrading", i.e., producing an individual in level $j+1$ after mutation of an individual in level j. Condition (C2) shows the fitness bias in uncertain environments. Condition (C3) states the required population size. Finally, we can get an upper bound for the runtime.

Theorem 3.2.1. *Let $(A_0, A_1 \dots A_m)$ be a fitness partition of a finite state space $\mathcal{X}$. Let $T := \min\{2t\lambda \mid |P_t \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of the 2-tournament EA with noisy function $f^n(x)$ and mutation rate χ/n. If there exist h_0, h_1, ..., h_{m-1} and $\theta \in (0, 1/2]$, and where $\chi \in (0, \ln(1+2\theta\zeta))$ for an arbitrary constant $\zeta \in (0,1)$, such that, for an arbitrary constant $\xi \in (0, 1/16)$,*

(C1) for all $j \in [0..m-1]$,

$$\Pr(y \in A_{\geq j+1} \mid z \in A_j) \geq h_j,$$

(C2) for all $j \in [0..m-2]$, and all search points $x_1 \in A_{\geq j+1}$ and $x_2 \in A_{\leq j}$, it follows

$$\Pr\left(f^n(x_1) > f^n(x_2)\right) + \frac{1}{2}\Pr\left(f^n(x_1) = f^n(x_2)\right) \geq \frac{1}{2} + \theta,$$

(C3) and the population size $\lambda \in \mathbb{N}$ satisfies

$$\lambda > \frac{4\,(1+o(1))}{\theta^2\xi(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4 \min\{h_j\}}\right),$$

then

$$E[T] < \frac{16\,(1+o(1))}{\theta^2\xi(1-\zeta)^2} \sum_{j=0}^{m-1} \left(\lambda \ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2 h_j}\right).$$

Proof. We use the level-based theorem (Theorem 2.2.1) to prove Theorem 3.2.1. Firstly, we derive some inequalities which are used later. From $\theta \in (0, 1/2]$, $\zeta \in (0, 1)$ and $0 < \chi < \ln(1 + 2\theta\zeta)$ which are assumptions of Theorem 3.2.1, we obtain

$$e^{\chi} < 1 + 2\theta\zeta \tag{3.1}$$

$$(1 + 2\theta) - e^{\chi} > 2\theta(1 - \zeta). \tag{3.2}$$

Then we define ε and γ_0 which are used later. Let constant $\varepsilon := \left(1 + \sqrt{1 - 4\sqrt{\xi}}\right)/2$, and we know that $\varepsilon \in (1/2, 1)$ by $\xi \in (0, 1/16)$. We define $\gamma_0 := \frac{(1+2\theta)-\exp(\chi)}{2\theta}(1-\varepsilon)$. By Eq. (3.2), we know that

$$\gamma_0 = \frac{(1 + 2\theta) - e^{\chi}}{2\theta}(1 - \varepsilon) > \frac{2\theta(1 - \zeta)(1 - \varepsilon)}{2\theta} = (1 - \zeta)(1 - \varepsilon). \tag{3.3}$$

We first show that condition (G2) of Theorem 2.2.1 holds. We define the "current level" to be the highest level $j \in [0..m - 1]$ such that there are at least $\gamma_0\lambda$ individuals in level j or higher, and there are fewer than $\gamma_0\lambda$ individuals in level $j + 1$ or higher. Following condition (G2) of Theorem 2.2.1, we assume that the current level is $j \leq m - 2$, which means that there are at least $\gamma_0\lambda$ individuals of the population P_t in $A_{\geq j}$, and at least $\gamma\lambda$ but less than $\gamma_0\lambda$ individuals in $A_{\geq j+1}$. Let x_1 and x_2 be the individuals selected from the population P_t in lines 1 and 2 of 2-tournament selection (Algorithm 5), z be the solution after comparison in line 3 of 2-tournament selection (Algorithm 5), and y be the solution after mutating corresponding to line 4 of Algorithm 3.

Now we estimate a lower bound on the probability that the offspring y is still in $A_{\geq j+1}$. By the law of total probability, $\Pr(y \in A_{\geq j+1}) \geq \Pr(z \in A_{\geq j+1}) \cdot \Pr(y \in A_{\geq j+1} | z \in A_{\geq j+1})$. The probability of selecting an individual z which is in $A_{\geq j+1}$ via binary tournament is composed of two cases. The first case is both x_1 and x_2 which are selected in lines 1 and 2 of Algorithm 5 are in $A_{\geq j+1}$ whose probability is at least γ^2. The second case is that x_1 or x_2 is evaluated to be in $A_{\geq j+1}$, whereas the other is evaluated to be in $A_{\leq j}$. In this case, noise

leads to incorrect comparison in line 3 of Algorithm 5 with some probability. Let S be the event of a successful comparison, i.e. the better individual of x_1 and x_2 is exactly selected from line 3 of Algorithm 5. Hence, the second case occurs with probability $2(1-\gamma)\gamma \Pr(S)$. Then, $\Pr(z \in A_{\geq j+1}) \geq \gamma^2 + 2(1-\gamma)\gamma \Pr(S)$.

To estimate a lower bound for $\Pr(S)$, we assume without loss of generality $x_1 \in A_{\geq j+1}$ and $x_2 \in A_{\leq j}$. Then, by condition (C2),

$$\begin{aligned}\Pr(S) &= \Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) \\ &\geq \frac{1}{2} + \theta.\end{aligned}$$

To estimate a lower bound for $\Pr(y \in A_{\geq j+1} \mid z \in A_{\geq j+1})$, we only consider the case that the mutation operator does not flip any bits, then by Lemma A.2.5 and Lemma A.2.4,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_{\geq j+1}) &\geq \left(1 - \frac{\chi}{n}\right)^n \\ &\geq e^{-\chi}\left(1 - \frac{\chi^2}{n}\right) \\ &\geq e^{-\chi}\left(1 - \frac{2\theta}{n}\right)\end{aligned}$$

for all $n > 1$.

Overall, we can get a lower bound for $\Pr(y \in A_{\geq j+1})$ by plugging in $\Pr(z \in A_{\geq j+1})$, $\Pr(y \in A_{\geq j+1} | z \in A_{\geq j+1})$ and $\Pr(S)$,

$$\begin{aligned}\Pr(y \in A_{\geq j+1}) &> \left(\gamma^2 + 2(1-\gamma)\gamma \Pr(S)\right) e^{-\chi}\left(1 - \frac{2\theta}{n}\right) \\ &\geq \left(\gamma^2 + 2(1-\gamma)\gamma\left(\frac{1}{2} + \theta\right)\right) e^{-\chi}\left(1 - \frac{2\theta}{n}\right) \\ &\geq \gamma\left(1 + 2\theta - 2\theta\gamma_0\right) e^{-\chi}\left(1 - \frac{2\theta}{n}\right)\end{aligned}$$

by the definition of $\gamma_0 = \frac{(1+2\theta)-\exp(\chi)}{2\theta}(1-\varepsilon)$,

$$\begin{aligned}
&= \gamma\left(1+2\theta-(1+2\theta-e^{\chi})+(1+2\theta-e^{\chi})\,\varepsilon\right)e^{-\chi}\left(1-\frac{2\theta}{n}\right)\\
&= \gamma\left(1+(1+2\theta-e^{\chi})\,\varepsilon e^{-\chi}\right)\left(1-\frac{2\theta}{n}\right)
\end{aligned}$$

letting $\delta := \left(1+(1+2\theta-e^{\chi})\,\varepsilon e^{-\chi}\right)(1-2\theta/n)-1$,

$$= \gamma(1+\delta). \tag{3.4}$$

Now we prove that $\delta > 0$, where

$$\delta = \left(1+(1+2\theta-e^{\chi})\,\varepsilon e^{-\chi}\right)\left(1-\frac{2\theta}{n}\right)-1$$

by Eq. (3.1),

$$\begin{aligned}
&> \left(1+2\theta e^{-\chi}\varepsilon(1-\zeta)\right)\left(1-\frac{2\theta}{n}\right)-1\\
&= 2\theta e^{-\chi}\varepsilon(1-\zeta)-\frac{2\theta}{n}\left(1+2\theta e^{-\chi}\varepsilon\right)+\frac{4\theta^2 e^{-\chi}\varepsilon\zeta}{n}\\
&> 2\theta e^{-\chi}\varepsilon(1-\zeta)-\frac{6\theta}{n}\\
&= \theta\left(2e^{-\chi}\varepsilon(1-\zeta)-\frac{6}{n}\right)
\end{aligned}$$

by Eq. (3.1), we have $e^{\chi} < 1+2\theta\zeta < 1+2\theta < 2$,

$$\begin{aligned}
&> \theta\left(\varepsilon(1-\zeta)-\frac{6}{n}\right)\\
&= \theta\varepsilon(1-\zeta)\left(1-o(1)\right).
\end{aligned} \tag{3.5}$$

Thus, we get $\delta > 0$ so condition (G2) of Theorem 2.2.1 holds from Eq. (3.4).

To verify condition (G1), we estimate the probability of sampling an individual beyond the current level of the population. We assume there are at least $\gamma_0\lambda$ individuals in $A_{\geq j}$

where $j \in [0..m-1]$. We only consider the case that the selected individuals are both in A_j in lines 1 and 2 of Algorithm 5, and the individual increases its level after mutation,

$$\begin{aligned}\Pr(y \in A_{\geq j+1}) &\geq \gamma_0^2 \Pr(y \in A_{\geq j+1} \mid z \in A_{\geq j}) \\ &\geq \gamma_0^2 h_j =: z_j.\end{aligned}$$

Condition (G3) requires the population size to satisfy

$$\frac{4}{\gamma_0 \delta^2} \ln\left(\frac{128(m+1)}{\min\{z_j\}\delta^2}\right) = \frac{4}{\gamma_0 \delta^2} \ln\left(\frac{128(m+1)}{\gamma_0^2 \min\{h_j\}\delta^2}\right)$$

by Eq. (3.3) and (3.5),

$$\begin{aligned}&< \frac{4\,(1+o(1))}{(1-\zeta)(1-\varepsilon)\,(\theta\varepsilon(1-\zeta))^2} \cdot \ln\left(\frac{128(m+1)\,(1+o(1))}{(1-\zeta)^2(1-\varepsilon)^2\,(\theta\varepsilon(1-\zeta))^2 \min\{h_j\}}\right) \\ &= \frac{4\,(1+o(1))}{\theta^2\varepsilon^2(1-\varepsilon)(1-\zeta)^3} \ln\left(\frac{128(m+1)}{\theta^2\varepsilon^2(1-\varepsilon)^2(1-\zeta)^4 \min\{h_j\}}\right) \\ &< \frac{4\,(1+o(1))}{\theta^2\varepsilon^2(1-\varepsilon)^2(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\varepsilon^2(1-\varepsilon)^2(1-\zeta)^4 \min\{h_j\}}\right)\end{aligned}$$

because $\varepsilon^2(1-\varepsilon)^2 = \xi$ by the definition of $\varepsilon = \left(1+\sqrt{1-4\sqrt{\xi}}\right)/2$,

$$< \frac{4\,(1+o(1))}{\theta^2\xi(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4 \min\{h_j\}}\right) < \lambda.$$

Therefore, condition (C3) of Theorem 3.2.1 guarantees that the population size satisfies condition (G3) of Theorem 2.2.1.

Finally, all conditions of Theorem 2.2.1 hold and the expected time (the reevaluation strategy is taken into account) to reach the optimum is no more than

$$\begin{aligned}E[T] &\leq 2 \cdot \frac{8}{\delta^2} \sum_{j=0}^{m-1} \left(\lambda \ln\left(\frac{6\delta\lambda}{4 + z_j\delta\lambda}\right) + \frac{1}{z_j}\right) \\ &< \frac{16}{\delta^2} \sum_{j=0}^{m-1} \left(\lambda \ln\left(\frac{6}{z_j}\right) + \frac{1}{z_j}\right) \\ &\leq \frac{16}{\delta^2} \sum_{j=0}^{m-1} \left(\lambda \ln\left(\frac{6}{\gamma_0^2 h_j}\right) + \frac{1}{\gamma_0^2 h_j}\right)\end{aligned}$$

by Eq. (3.3) and (3.5),

$$
\begin{aligned}
&< \frac{16\,(1+o(1))}{\theta^2\varepsilon^2(1-\zeta)^2}\sum_{j=0}^{m-1}\left(\lambda\ln\left(\frac{6}{(1-\varepsilon)^2(1-\zeta)^2h_j}\right)+\frac{1/h_j}{(1-\varepsilon)^2(1-\zeta)^2}\right)\\
&< \frac{16\,(1+o(1))}{\theta^2\varepsilon^2(1-\varepsilon)^2(1-\zeta)^2}\sum_{j=0}^{m-1}\left(\lambda\ln\left(\frac{6}{\varepsilon^2(1-\varepsilon)^2(1-\zeta)^2h_j}\right)+\frac{1/h_j}{\varepsilon^2(1-\varepsilon)^2(1-\zeta)^2}\right)\\
&= \frac{16\,(1+o(1))}{\theta^2\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\left(\lambda\ln\left(\frac{6}{\xi(1-\zeta)^2h_j}\right)+\frac{1}{\xi(1-\zeta)^2h_j}\right).
\end{aligned}
$$

□

In Lemma 3.2.1, we show fitness biases of OneMax and LeadingOnes functions in the one-bit, the bit-wise, the Gaussian and the symmetric noise models.

Lemma 3.2.1. *Let $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$ be a partition of $\{0,1\}^n$. Let x_1 and x_2 be two individuals in $A_{\geq j+1}$ and $A_{\leq j}$ respectively, where $j \in [0..n-2]$, then $\theta_1 \leq \Pr\left(f^n(x_1) > f^n(x_2)\right) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)\big) - 1/2 \leq \theta_2$ where*

(a) $\theta_1 = 1/2 - q/2(1-q/2) - q/(2n_0)$ for $q \in [0,1)$ and $n_0 \in [3,\infty)$ on OneMax in the one-bit noise model (q),

(b) $\theta_1 = 1/2 - q(1-q/2)$ for $q \in [0,1)$ on LeadingOnes in the one-bit noise model (q),

(c) $\theta_1 = \frac{9(1/2-p)}{64\sqrt{2pn+16}}$ for $p \in (0,1/2)$ on OneMax in the bit-wise noise model (p),

(d) $\theta_1 = (1/2 - 3p/2)\,e^{-3np}$ for $p \in [0,1/3)$ on LeadingOnes in the bit-wise noise model (p), and

(e) $\theta_1 = 1/(6 + 48\sigma/\pi)$ for $\sigma > 0$ on OneMax and LeadingOnes in the Gaussian noise model (σ^2).

(f) $\theta_1 = \theta_2 = 1/2 - q$ for any $C \in \mathbb{R}$ and $q \in [0,1/2)$ on OneMax and LeadingOnes in the symmetric noise model (C,q).

Proof. Let E be the event that $f^n(x_1) > f^n(x_2)$ or individual x_1 is selected uniformly from $\{x_1, x_2\}$ if $f^n(x_1) = f^n(x_2)$, then $\Pr(E) = \Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2))$. Now we derive the fitness bias in different cases.

(a) To estimate a lower bound for $\Pr(E)$ on the OneMax problem in the one-bit noise model (q), we pessimistically assume that $x_1 \in A_{j+1}$ and $x_2 \in A_j$. Then we say that x_1 "wins" if the event E happens, and we distinguish between four cases:

- Let E_{00} be the event that there is no noise, and x_1 wins, then $\Pr(E_{00}) = (1-q)^2$.
- Let E_{01} be the event that there is no noise in x_1 and noise in x_2, and x_1 wins, then $\Pr(E_{01}) \geq (1-q)q\left(\left(1-\frac{j}{n}\right)\frac{1}{2} + \frac{j}{n}\right) = q(1-q)\left(\frac{j}{2n} + \frac{1}{2}\right)$.
- Let E_{10} be the event that there is noise in x_1 and no noise in x_2, and x_1 wins, then $\Pr(E_{10}) \geq q(1-q)\left(\frac{j+1}{n}\cdot\frac{1}{2} + \left(1-\frac{j+1}{n}\right)\right) = q(1-q)\left(-\frac{j}{2n} - \frac{1}{2n} + 1\right)$.
- Let E_{11} be the event that there is noise in x_1 and x_2, and x_1 wins. There are two situations leading x_1 to win:

  1. The noise flips one of the $j+1$ 1-bits of x_1 and one of the j 1-bit of x_2.
  2. The noise flips one of $n-(j+1)$ 0-bits of x_1.

  Thus,

$$\begin{aligned}\Pr(E_{11}) &\geq q^2\left(\frac{j+1}{n}\cdot\frac{j}{n} + \left(1-\frac{j+1}{n}\right)\right)\\ &= \left(\frac{(j+1)(j-n)}{n^2} + 1\right)q^2\end{aligned}$$

  since $(j+1)(j-n)$ achieves the minimum when $j = (n-1)/2$,

$$\geq \left(\frac{3}{4} - \frac{1}{2n}\right)q^2.$$

By combining all four cases above, we obtain

$$\begin{aligned}
\Pr(E) &\geq \Pr(E_{00}) + \Pr(E_{01}) + \Pr(E_{10}) + \Pr(E_{11}) \\
&\geq (1-q)^2 + q(1-q)\left(\frac{j}{2n} + \frac{1}{2}\right) + q(1-q)\left(-\frac{j}{2n} - \frac{1}{2n} + 1\right) + \left(\frac{3}{4} - \frac{1}{2n}\right)q^2 \\
&= 1 - \frac{q}{2} + \frac{q^2}{4} - \frac{q}{2n} \\
&= \frac{1}{2} + \frac{1}{2} - \frac{q}{2}\left(1 - \frac{q}{2}\right) - \frac{q}{2n} \\
&\geq \frac{1}{2} + \frac{1}{2} - \frac{q}{2}\left(1 - \frac{q}{2}\right) - \frac{q}{2n_0} \\
&= \frac{1}{2} + \theta
\end{aligned}$$

(b) To estimate a lower bound for $\Pr(E)$ on the LeadingOnes problem in the one-bit noise model (q), we pessimistically assume that $x_1 \in A_{j+1}$ and $x_2 \in A_j$. We also pessimistically assume that the suffix of x_1, i.e. the bits after the $(j+2)$-th position, are all 0-bits, and the suffix of x_2, i.e. the bits after the $(j+1)$-th position, are all 1-bits, which is the worst case because if the noise flips the $(j+1)$-th bit in x_2, then x_2 will have the maximal noisy fitness n. We say that x_1 "wins" if the event E happens, then we distinguish between four cases to estimate $\Pr(E)$:

- Let E_{00} be the event that there is no noise, and x_1 wins, then $\Pr(E_{00}) = (1-q)^2 = q^2 - 2q + 1$.
- Let E_{01} be the event that there is no noise in x_1 and noise in x_2, and x_1 wins. By the assumption of x_2, x_1 only fails if noise flips the only 0-bit in x_2. Thus, $\Pr(E_{01}) \geq (1-q) \cdot q \cdot (1-1/n) = -(1-1/n)\,q^2 + (1-1/n)\,q$.
- Let E_{10} be the event that there is noise in x_1 and no noise in x_2, and x_1 wins. Unless any of the first $j+1$ 1-bits of x_1 is flipped, x_1 wins. Therefore, $\Pr(E_{10}) \geq q \cdot (1-q) \cdot (1-(j+1)/n) = -(1-(j+1)/n) \cdot q^2 + (1-(j+1)/n)\,q$.

- Let E_{11} be the event that there is noise in x_1 and x_2, and x_1 wins. Because $j = n-2$ is a special case, we first estimate the probability $\Pr(E_{11})$ when $j \leq n-3$. There are four situations leading x_1 to win:

  1. The noise does not flip the first $j+1$ 1-bits of x_1, and does not flip the 0-bit of x_2.
  2. The noise flips the i-th 1-bits of x_1 where $i \leq j+1$, and flips one of the first $i-1$ 1-bits of x_2,
  3. The noise flips the same bit-position in the first j 1-bits of x_1 and x_2 (tie and half chance to win).
  4. The noise flips the $(j+1)$-th 1-bit of x_1, and does not flip the first j 1-bits of x_2 (tie and half chance to win).

  Thus,

$$\begin{aligned}\Pr(E_{11}) &\geq q^2\left(\left(1-\frac{j+1}{n}\right)\left(1-\frac{1}{n}\right)+\sum_{i=2}^{j+1}\left(\frac{i-1}{n}\cdot\frac{1}{n}\right)+\frac{j}{2n^2}+\frac{1}{2n}\left(1-\frac{j+1}{n}\right)\right)\\ &= \left(j^2/2-(n-3/2)\,j+(n-1)(2n-1)/2\right)\frac{q^2}{n^2}\end{aligned}$$

  since $j^2/2-(n-3/2)j+(n-1)(2n-1)/2$ is monotone decreasing as j increases when $j \leq n-3/2$, $\Pr(E_{11})$ achieves the minimum if $j = n-3$,

$$\begin{aligned}&\geq \frac{1}{2}\left(1+\frac{1}{n^2}\right)q^2\\ &> \frac{q^2}{2}.\end{aligned}$$

  Then we estimate $\Pr(E_{11})$ in the special case $j = n-2$. Since both x_1 and x_2 have one 0-bit to the optimum, i.e. x_1 has only one 0-bit in the last position and x_2 has only one 0-bit in the penultimate position, there are five situations to leading x_1 to win:

  1. The noise flips the i-th 1-bits of x_1 where $i \leq n-1$, and flips one of the first $i-1$ 1-bits of x_2.

2. The noise flips the same bit-position in the first $n-2$ 1-bits of x_1 and x_2 (tie and half chance to win).
3. The noise flips the last 0-bits of x_1, and does not flip the 0-bit of x_2.
4. The noise flips both the 0-bits of x_1 and x_2 (tie and half chance to win).
5. The noise flips the $(n-1)$-th 1-bit of x_1, and flips the last 0-bits of x_2 (tie and half chance to win).

Thus,

$$\Pr(E_{11}) \geq q^2 \left(\sum_{i=2}^{n-1} \left(\frac{i-1}{n} \cdot \frac{1}{n} \right) + \frac{n-2}{2n^2} + \frac{1}{n} \left(1 - \frac{1}{n} \right) + \frac{1}{2n^2} + \frac{1}{2n^2} \right) = \frac{q^2}{2}.$$

Therefore, we obtain $\Pr(E_{11}) \geq q^2/2$ for all $j \leq n-2$.

By combining all four cases above and $j \leq n-2$, we obtain

$$\begin{aligned}
\Pr(E) &\geq \Pr(E_{00}) + \Pr(E_{01}) + \Pr(E_{10}) + \Pr(E_{11}) \\
&\geq 1 - \frac{j+2}{n} q + \left(\frac{j+2}{n} - 1 \right) q^2 + \frac{q^2}{2} \\
&\geq 1 - \frac{j+2}{n} (1-q) q - q^2 + \frac{q^2}{2} \\
&\geq 1 - (1-q) q - q^2 + \frac{q^2}{2} \\
&= 1 - q + \frac{q^2}{2} \\
&= \frac{1}{2} + \frac{1}{2} - q(1 - \frac{q}{2}) \\
&= \frac{1}{2} + \theta.
\end{aligned}$$

(c) To estimate a lower bound for $\Pr(E)$ on the OneMax problem in the bit-wise noise model (p), we pessimistically assume that $x_1 \in A_{j+1}$ and $x_2 \in A_j$, such that $f(x_1) = f(x_2)+1$. There exists at least one bit-position i, such that x_1 has a 1-bit in position i and x_2 has a 0-bit in position i. The remaining bits of x_1 and x_2 have the same number of 1-bits. Therefore,

the bits after noise of x_1 and x_2 in position i decide the outcome of the fitness comparison. Let x_1' and x_2' be two substrings obtained by excluding position i from x_1 and x_2 respectively. Since each bit is flipped independently, we know that $f^n(x_1'), f^n(x_2') \sim \text{Bin}(n-1-j,p) + \text{Bin}(j,(1-p))$ which are Poisson-binomially distributed random variables with variance $\sigma^2 = (1-p)p(n-1)$. Then we apply Lemma A.2.10 with $\sigma = \sqrt{(1-p)p(n-1)} \leq \sqrt{pn}$ and $d = 2$ to obtain a lower bound for

$$\Pr(f^n(x_1') = f^n(x_2')) \geq \frac{(1-1/2^2)^2}{2 \cdot 2\sqrt{2pn}+1} \geq \frac{9}{64\sqrt{2pn}+16}. \tag{3.6}$$

By symmetry, we know that $\Pr(f^n(x_1') > f^n(x_2')) = \Pr(f^n(x_1') < f^n(x_2'))$. Let $a = \Pr(f^n(x_1') = f^n(x_2'))$ and $b = Pr(f^n(x_1') > f^n(x_2'))$, then we obtain $a = 1-2b$. Thus,

$$\begin{aligned}
\Pr(E) &= b + a\left((1-p)^2 + 2 \cdot \frac{1}{2} \cdot p(1-p)\right) \\
&= \frac{1}{2}(1-a) + a \cdot (1-p) \\
&= \frac{1}{2} + \left(\frac{1}{2} - p\right) a
\end{aligned}$$

by Eq. (3.6),

$$\begin{aligned}
&\geq \frac{1}{2} + \frac{9\,(1/2-p)}{64\sqrt{2pn}+16} \\
&= \frac{1}{2} + \theta.
\end{aligned}$$

(d) To estimate a lower bound for $\Pr(E)$ on the LeadingOnes problem in the bit-wise noise model (p), we pessimistically assume that $x_1 \in A_{j+1}$ and $x_2 \in A_j$. We also pessimistically assume that the suffix of x_1, i.e. the bits after the $(j+2)$-th position, are all 0-bits, and the suffix of x_2, i.e. the bits after the $(j+1)$-th position, are all 1-bits, which is the worst case since the noise flipping $(j+1)$-th bit of x_2 to achieve the optimum which leads to incorrect comparison while the noise only can at most increase 1 fitness for x_1. We distinguish between three cases to estimate $\Pr(E)$:

- Let E_0 be the event that $f^n(x_1) \geq j+1$ and $f^n(x_2) \leq j$, then $\Pr(E_0) = (1-p)^{j+1}(1-p) + (1-p)^{j+1}p\left(1-(1-p)^j\right) = \left(1-p(1-p)^j\right)(1-p)^{j+1}$.

- Let E_{1i} be the event that $f^n(x_1) = i$ and $f^n(x_2) \leq i-1$ for any $i \in [1,j]$, then $\Pr(E_1) = \sum_{i=1}^{j}\Pr(E_{1i}) = \sum_{i=1}^{j}\left(p(1-p)^i \cdot \left(1-(1-p)^i\right)\right) = p\left(\sum_{i=1}^{j}(1-p)^i - \sum_{i=1}^{j}(1-p)^{2i}\right)$, by the sum of a geometric series,

$$\begin{aligned}\Pr(E_1) &= p\left(\frac{1-(1-p)^{j+1}}{1-(1-p)} - \frac{1-(1-p)^{2(j+1)}}{1-(1-p)^2}\right)\\ &= \frac{1}{2} - \frac{p}{2(2-p)} - (1-p)^{j+1} + \frac{1}{2-p}(1-p)^{2(j+1)}.\end{aligned}$$

- Let E_{2i} be the event that $f^n(x_1) = i$ and $f^n(x_2) = i$ for any $i \in [0,j]$ then x_1 is selected uniformly, then

$$\begin{aligned}\Pr(E_2) &= \frac{1}{2}\sum_{i=0}^{j}\Pr(E_{2i})\\ &= \frac{1}{2}\left(\sum_{i=0}^{j-1}p^2(1-p)^{2i} + p(1-p)(1-p)^{2j}\right)\end{aligned}$$

since $p < 1/3$,

$$\geq \frac{1}{2}\sum_{i=0}^{j}p^2(1-p)^{2i}$$

by the sum of a geometric series,

$$\begin{aligned}&= \frac{1}{2}p^2\left(\frac{1-(1-p)^{2(j+1)}}{1-(1-p)^2}\right)\\ &= \frac{p}{2(2-p)} - \frac{p}{2(2-p)}(1-p)^{2(j+1)}.\end{aligned}$$

By combining all three cases above, we obtain

$$\begin{aligned}
\Pr(E) &\geq \Pr(E_0) + \Pr(E_1) + \Pr(E_2) \\
&= \left(1 - p(1-p)^j\right)(1-p)^{j+1} + \frac{1}{2} - \frac{p}{2(2-p)} - (1-p)^{j+1} \\
&\quad + \frac{1}{2-p}(1-p)^{2(j+1)} + \frac{p}{2(2-p)} - \frac{p}{2(2-p)}(1-p)^{2(j+1)} \\
&= \frac{1}{2} + \frac{(1-p)^{2(j+1)}}{2-p} - \frac{p}{2(2-p)}(1-p)^{2(j+1)} - p(1-p)^{2j+1} \\
&= \frac{1}{2} + \frac{1}{2}(1-p)^{2j+2} - p(1-p)^{2j+1} = \frac{1}{2} + \left(\frac{1}{2} - \frac{3}{2}p\right)(1-p)^{2j+1}
\end{aligned}$$

by $\left((1-x)^{1/x-1}\right)^y \geq e^{-y}$ (see Lemma A.2.6),

$$\begin{aligned}
&\geq \frac{1}{2} + \left(\frac{1}{2} - \frac{3}{2}p\right) e^{-(2j+1)\frac{p}{1-p}} \\
&> \frac{1}{2} + \left(\frac{1}{2} - \frac{3}{2}p\right) e^{-2(j+2)\frac{p}{1-p}}
\end{aligned}$$

by $0 \leq j \leq n-2$ and $p \in [0, 1/3)$,

$$\begin{aligned}
&\geq \frac{1}{2} + \left(\frac{1}{2} - \frac{3}{2}p\right) e^{-3np} \\
&\geq \frac{1}{2} + \theta.
\end{aligned}$$

(e) To estimate a lower bound for $\Pr(E)$ on the OneMax and LeadingOnes problems in the Gaussian noise model (σ^2), we pessimistically assume that $x_1 \in A_{j+1}$ and $x_2 \in A_j$. Let $X \sim \mathcal{N}(0, 2\sigma^2)$ be a random variable, then

$$\Pr(E) \geq \Pr(f^n(x_1) - f^n(x_2) > 0) = \Pr(X > -1) = \Pr(X < 1)$$

by Lemma A.2.9 with $x = 1$ and standard deviation is $\sqrt{2}\sigma$,

$$
\begin{aligned}
&> 1 - 1/\sqrt{\pi/(\sqrt{2}\sqrt{2}\sigma) + 4} \\
&= \left(1 - \left(1/\sqrt{\pi/(2\sigma) + 4}\right)^2\right) / \left(1 + 1/\sqrt{\pi/(2\sigma) + 4}\right) \\
&> \left(\frac{\pi/(2\sigma) + 3}{\pi/(2\sigma) + 4}\right) / \frac{3}{2} \\
&= \frac{1 + 6\sigma/\pi}{1 + 8\sigma/\pi} \cdot \frac{4}{3} \cdot \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{6 + 48\sigma/\pi} \\
&= \frac{1}{2} + \theta.
\end{aligned}
$$

(f) To estimate a lower bound for $\Pr(E)$ on OneMax and LeadingOnes in the symmetric noise model (C, q), we assume that $x_1 \in A_a$ and $x_2 \in A_b$ where $a > b$. Then we say that x_1 "wins" if event E happens, and we distinguish between three cases:

- If $a+b > C$, then x_1 wins if and only if there is noise in x_2, i.e, $\Pr(E) = (1-q)^2+(1-q)q$.
- If $a + b = C$, then x_1 wins if and only if there is no noise in both x_1 and x_2, or there is noise in either x_1 and x_2 (same fitness values, so with half chance), i.e., $\Pr(E) = (1-q)^2 + (1-q)q/2 + q(1-q)/2$.
- If $a + b < C$, then x_1 wins if and only if there is no noise in x_2, i.e., $\Pr(E) = (1-q)^2 + q(1-q)$.

Therefore, we obtain

$$\Pr(E) = (1-q)^2 + (1-q)q = \frac{1}{2} + \frac{1}{2} - q = \frac{1}{2} + \theta.$$

□

3.3 Noisy Optimisation

This section provides runtime bounds for non-elitist EAs with 2-tournament and (μ, λ) selection on two classical functions in four noise models. We give the upper bound of runtime and the appropriate parameter settings, e.g., mutation rate, which leads to efficient optimisation for each noise model. In particular, we show for which parameter settings the non-elitist EA is inefficient in the symmetric noise model.

3.3.1 One-bit Noise Model

Theorem 3.3.1 implies that one-bit noise does not impact the asymptotical runtime of the 2-tournament EA on OneMax if we choose a constant mutation parameter χ which satisfies the assumption. However, we have fewer choices of the mutation rate as the level of is noise growing. In contrast, the (1+1) EA becomes inefficient if the noise level is a constant (see Tables 2.3 and 2.4). Compared to other EAs, e.g., ACO-fp, UMDA and (1+1) EA (resampling), the 2-tournament EA can outperform the current state of the art results in these two settings (see Tables 2.3 and 2.4).

Theorem 3.3.1. *For any constant $q \in [0,1]$, any constant $n_0 \in [3,\infty)$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 1/2-(q/2)(1-q/2)-q/(2n_0)$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > c\log(n/\chi)$ for a sufficiently large constant c achieves the optimum on OneMax in the one-bit noise model (q) in expected time $O(\lambda n \log(1/\chi) + n\log(n)/\chi)$.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.1. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. By constants $q \in [0,1]$ and $n_0 \in [3,\infty)$, we obtain $1/12 < \theta \leq 1/2$ which satisfies the assumption in Theorem 3.2.1.

By case (a) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) > 1/2+\theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that no 1-bits is flipped and one of 0-bits is flipped after mutation, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_j) &> \left(1-\frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\ &\geq e^{-\chi}\left(1-o(1)\right)(n-j)\chi/n =: h_j \in \Omega\left((n-j)\chi/n\right).\end{aligned}$$

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a constant, then

$$\lambda > \frac{4\left(1+o(1)\right)}{\theta^2\xi(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4 \min\{h_j\}}\right) = O(\log(n/\chi)).$$

Condition (C3) is satisfied by $\lambda \geq c\log(n/\chi)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time to reach the optimum is no more than

$$\begin{aligned}E[T] &\leq \frac{16\left(1+o(1)\right)}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right) \\ &= O\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{n}{(n-j)\chi}\right) + \sum_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right) \\ &= O\left(\lambda\ln\left(\prod_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right) + n\sum_{j=0}^{m-1}\frac{1}{(n-j)\chi}\right) \\ &= O\left(\lambda\ln\left(\frac{n^n}{n!\chi^n}\right) + n\log(n)/\chi\right)\end{aligned}$$

using the lower bound $n! > (n/e)^n$,

$$= O\left(\lambda n\log(1/\chi) + n\log(n)/\chi\right).$$

□

Theorem 3.3.2. *For any constant $q \in [0,1)$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 1/2 - q(1-q/2)$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > c\log(n/\chi)$ for a sufficiently large constant c achieves the optimum on* LeadingOnes *in the one-bit noise model (q) in expected time $O(n\lambda\log(n/\chi) + n^2/\chi)$.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.2. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. By $q \in [0,1)$, we know that $0 < \theta \leq \frac{1}{2}$ which satisfies the assumption in Theorem 3.2.1.

By case (b) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) > 1/2 + \theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that the first 0-bit is flipped and other bits are not flipped. By Lemma A.2.8 it follows,

$$\begin{aligned} \Pr(y \in A_{\geq j+1} \mid z \in A_j) &\geq \left(1 - \frac{\chi}{n}\right)^{n-1} \frac{\chi}{n} \geq e^{-\chi}(1 - o(1))\frac{\chi}{n} \\ &=: h_j = \Omega(\chi/n). \end{aligned} \tag{3.7}$$

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a constant, then

$$\lambda > \frac{4(1+o(1))}{\theta^2\xi(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4 \min\{h_j\}}\right) = O(\log(n/\chi))$$

Condition (C3) is satisfied by $\lambda \geq c \log(n/\chi)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time to reach the optimum is no more than

$$\begin{aligned} E[T] &< \frac{16\,(1+o(1))}{\theta^2 \xi (1-\zeta)^2} \left(\lambda \sum_{j=0}^{m-1} \ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2} \sum_{j=0}^{m-1} \frac{1}{h_j} \right) \\ &= O\left(\lambda \sum_{j=0}^{m-1} \ln(n/\chi) + \sum_{j=0}^{m-1} n/\chi \right) \\ &= O\left(n\lambda \log(n/\chi) + n^2/\chi \right). \end{aligned}$$

□

3.3.2 Bit-wise Noise Model

The best-known result on OneMax in the bit-wise noise model is that the (1+1) EA using a resampling strategy can achieve the optimum in expected polynomial time even if the noise level is extremely high, i.e. $p = 1/2 - 1/n^b$ for any constant $b > 0$ (see Table 2.5). By Theorem 3.3.3, we can compute that for extremely high-levels of bit-wise noise, the 2-tournament EA with mutation rate $\chi/n = \theta\zeta/n$ which is less than $\ln(1+2\theta\zeta)/n$ by Lemma A.2.1, i.e., $\chi = d/n^{b+1/2}$ for some constant $d > 0$, and a sufficiently large population size $\lambda \in \Omega\left(n^{2b+1}\log(n)\right)$ has polynomial expected runtime $O\left(n^{2b+2}\lambda\log(n)\right)$ on OneMax. In contrast, the (1+1) EA cannot find the optimum in expected polynomial time for noise levels $q \in \omega\left(\log(n)/n^2\right)$ (see Table 2.5).

Theorem 3.3.3. *For any $p \in (0, 1/2)$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 9(1/2 - p)/\left(64\sqrt{2pn} + 16\right)$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > \frac{c(1+pn)}{(1-2p)^2} \log\left(\frac{n}{(1-2p)\chi}\right)$ for a sufficiently large constant c achieves the optimum on OneMax in the bit-wise noise model (p) in expected time $O\left(\frac{n(1+pn)}{(1-2p)^2}\left(\lambda \log\left(\frac{1}{\chi}\right) + \frac{\log(n)}{\chi}\right)\right)$.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.3. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. Since $p \in (0, 1/2)$, we know that $0 < \theta < 9/32$ which satisfies the assumption in Theorem 3.2.1.

By case (c) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) > 1/2+\theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that no 1-bits are flipped and one of the 0-bits is flipped after mutation and by Lemma A.2.8,

$$\begin{aligned}
\Pr(y \in A_{\geq j+1} \mid z \in A_j) &> \left(1-\frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\
&\geq \left(1-\frac{\chi}{n}\right)^n \frac{\chi}{n}(n-j) \\
&\geq e^{-\chi}\left(1-\frac{\chi^2}{n}\right)\frac{\chi}{n}(n-j) \\
&\geq e^{-\chi}\left(1-o(1)\right)(n-j)\chi/n =: h_j = \Omega\left((n-j)\chi/n\right)
\end{aligned}$$

since $e^{-\chi} \in \Omega(1)$ for any $\chi \in O(1)$.

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a

constant, then

$$\begin{aligned}\lambda &> \frac{4\,(1+o(1))}{\theta^2\xi(1-\zeta)^4}\ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4\min\{h_j\}}\right)\\ &= O\left(\frac{1}{\theta^2}\ln\left(\frac{n^2}{\chi\theta^2}\right)\right)\\ &= O\left(\frac{\log\left(n/(\chi\theta)\right)}{\theta^2}\right)\\ &= O\left(\frac{1+pn}{(1-2p)^2}\log\left(\frac{n}{(1-2p)\chi}\right)\right).\end{aligned}$$

Condition (C3) is satisfied by $\lambda \geq c\frac{1+pn}{(1-2p)^2}\log\left(\frac{n}{(1-2p)\chi}\right)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time to reach the optimum is no more than

$$\begin{aligned}E[T] &\leq \frac{16\,(1+o(1))}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2h_j}\right)+\frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right)\\ &= O\left(\frac{1}{\theta^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{n}{(n-j)\chi}\right)+\sum_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right)\right)\\ &= O\left(\frac{1}{\theta^2}\left(\lambda\ln\left(\frac{n^n}{\chi^n\cdot n!}\right)+\frac{n}{\chi}\log(n)\right)\right)\end{aligned}$$

using the lower bound $n! > (n/e)^n$,

$$\begin{aligned}&= O\left(\frac{1}{\theta^2}\left(n\lambda\log\left(\frac{1}{\chi}\right)+\frac{n}{\chi}\log(n)\right)\right)\\ &= O\left(\frac{n(1+pn)}{(1-2p)^2}\left(\lambda\log\left(\frac{1}{\chi}\right)+\frac{\log(n)}{\chi}\right)\right).\end{aligned}$$

□

For the LeadingOnes problem, we consider the case of the high bit-wise noise $p = b\log(n)/n$ for any constant $b > 0$. By Theorem 3.3.4, the 2-tournament EA with mutation rate $\chi/n = \theta\zeta/n$ which satisfies the condition, i.e., $\chi = d/n^{3b}$ for some constant $d > 0$,

and a sufficiently large population size $\lambda \in \Omega\left(n^{6b}\log(n)\right)$ achieves the optimum on LeadingOnes in expected time $O\left(n^{6b+1}\lambda\log(n) + n^{9b+2}\right)$. In contrast, the expected runtime of the (1+1) EA with a resampling strategy is $12mn^{30b+1}$ under high bit-wise noise (see Table 2.6).

Theorem 3.3.4. *For any $p \in [0, 1/3)$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := (1/2 - 3p/2)\, e^{-3np}$, the 2-tournament EA with mutation rate χ/n and population size $\lambda \geq c\frac{e^{6np}}{(1-3p)^2}\log\left(\frac{n}{\chi}\right)$ for a sufficiently large constant c achieves the optimum on LeadingOnes in the bit-wise noise model (p) in expected time $O\left(\frac{ne^{6np}}{(1-3p)^2}\left(\lambda\log\left(\frac{n}{\chi}\right) + \frac{n}{\chi}\right)\right)$.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.4. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. Since $p \in [0, 1/3)$, we know that $0 < \theta \leq 1/2$ satisfies the assumption in Theorem 3.2.1.

By case (d) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) > 1/2+\theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that the first 0-bit is flipped and other bits are not flipped, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_j) &\geq \left(1 - \frac{\chi}{n}\right)^{n-1}\frac{\chi}{n} \geq e^{-\chi}\left(1 - o(1)\right)\frac{\chi}{n} =: h_j \\ &= \Omega\left(\chi/n\right).\end{aligned}$$

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a

constant, then

$$\begin{aligned}\lambda &> \frac{4\,(1+o(1))}{\theta^2\xi(1-\zeta)^4}\ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4\min\{h_j\}}\right)\\ &= O\left(\log\left(n/\chi\right)/\theta^2\right)\\ &= O\left(\frac{e^{6np}}{(1-3p)^2}\log\left(\frac{n}{\chi}\right)\right).\end{aligned}$$

Condition (C3) is satisfied by $\lambda \geq c\frac{e^{6np}}{(1-3p)^2}\log\left(\frac{n}{\chi}\right)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time to reach the optimum is no more than

$$\begin{aligned}E[T] &< \frac{16\,(1+o(1))}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2h_j}\right)+\frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right)\\ &= O\left(\frac{1}{\theta^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{n}{\chi}\right)+\sum_{j=0}^{m-1}\frac{n}{\chi}\right)\right)\\ &= O\left(\frac{1}{\theta^2}\left(n\lambda\log\left(\frac{n}{\chi}\right)+\frac{n^2}{\chi}\right)\right)\\ &= O\left(\frac{1}{\theta^2}\left(n\lambda\log\left(n/\chi\right)+\frac{n^2}{\chi}\right)\right)\\ &= O\left(\frac{ne^{6np}}{(1-3p)^2}\left(\lambda\log\left(\frac{n}{\chi}\right)+\frac{n}{\chi}\right)\right).\end{aligned}$$

□

3.3.3 Gaussian Noise Model

Theorem 3.3.5 implies that the 2-tournament EA with mutation rate $\chi/n = \theta\zeta/n$, i.e., $\chi = d/\sigma$ for some constant $d > 0$, and a sufficiently large population size $\lambda \in \Omega\left(\sigma^2\log(\sigma n)\right)$ can optimise OneMax and LeadingOnes in expected polynomial time, i.e., $O\big((\sigma^4\log^2(n)+\sigma^3 n\log(n)\big)$ and $O\left(\sigma^4\log^2(n)+\sigma^4n^2\right)$ respectively, even if $\sigma^2 \in \text{poly}(n)$. Similarly to optimisation in the bit-wise noise model, the mutation rate should be fairly conservative, and the population size should be large enough if the noise level is extremely high, e.g., $\sigma = n^b$

for any constant $b > 0$. However, the (1+1) EA using a resampling strategy and EDAs can outperform the 2-tournament EA in these scenarios. It may be possible to increase the tournament size to achieve a better result.

Theorem 3.3.5. *For any $\sigma > 0$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 1/(6+48\sigma/\pi)$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > c\sigma^2 \log(n/\chi)$ for a sufficiently large constant c achieves the optimum on* OneMax *and* LeadingOnes *in the Gaussian noise model (σ^2) in expected time $O(\sigma^2\lambda n \log(1/\chi) + \sigma^2 n \log(n)/\chi)$ and $O(\sigma^2\lambda n \log(n/\chi) + \sigma^2 n^2/\chi)$, respectively.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.5. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. By $\sigma \in \text{poly}(n)$, we obtain $0 < \theta < \frac{1}{6}$ which satisfies the assumption in Theorem 3.2.1.

By case (e) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) > 1/2+\theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that no 1-bits is flipped and one of the 0-bits is flipped after mutation for OneMax, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_j) &> \left(1-\frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\ &\geq e^{-\chi}(1-o(1))(n-j)\chi/n =: h_j\end{aligned}$$

For LeadingOnes, we only consider the case that the first 0-bit is flipped and other bits

are not flipped, then by Lemma A.2.8,

$$\Pr(y \in A_{\geq j+1} \mid z \in A_j) \geq \left(1 - \frac{\chi}{n}\right)^{n-1} \frac{\chi}{n} \geq e^{-\chi} \left(1 - o(1)\right) \frac{\chi}{n} =: h_j.$$

Then, we get $h_j \in \Omega\left((n-j)\chi/n\right)$ and $h_j \in \Omega\left(\chi/n\right)$ for OneMax and LeadingOnes respectively.

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a constant and we know that $\min\{h_j\} \in \Omega\left(\chi/n\right)$ for both problems, then

$$\lambda > \frac{4\left(1 + o(1)\right)}{\theta^2 \xi (1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2 \xi (1-\zeta)^4 \min\{h_j\}}\right) = O\left(\sigma^2 \log(n/\chi)\right).$$

Condition (C3) is satisfied by $\lambda \geq c\sigma^2 \log(n/\chi)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time on OneMax to reach the optimum is no more than

$$\begin{aligned}
E[T] &\leq \frac{16\left(1 + o(1)\right)}{\theta^2 \xi (1-\zeta)^2} \left(\lambda \sum_{j=0}^{m-1} \ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2} \sum_{j=0}^{m-1} \frac{1}{h_j}\right) \\
&= O\left(\sigma^2 \left(\lambda \sum_{j=0}^{m-1} \ln\left(\frac{n}{(n-j)\chi}\right) + \sum_{j=0}^{m-1} \frac{n}{(n-j)\chi}\right)\right) \\
&= O\left(\sigma^2 \left(\lambda \ln\left(\frac{n^n}{\chi^n n!}\right) + n\log(n)/\chi\right)\right)
\end{aligned}$$

using the lower bound $n! > (n/e)^n$,

$$= O\left(\sigma^2 \left(\lambda n \log(1/\chi) + n\log(n)/\chi\right)\right),$$

and on LeadingOnes,

$$\begin{aligned}
E[T] &= O\left(\sigma^2 \left(\lambda \sum_{j=0}^{m-1} \ln\left(\sigma n\right) + \sum_{j=0}^{m-1} \sigma n\right)\right) \\
&= O\left(\sigma^2 \lambda n \log(n/\chi) + \sigma^2 n^2/\chi\right).
\end{aligned}$$

□

3.3.4 Symmetric Noise Model

Resampling is a common method to cope with uncertainties (Qian et al., 2019; Qian et al., 2018). It dramatically improves the robustness of the (1+1) EA on OneMax and LeadingOnes in the one-bit, the bit-wise and the Gaussian noise (see Tables 2.3-2.8). However, from Table 2.9, we know that the symmetric noise model (C, q) for any $C \in \mathbb{R}$ and $q = 1/2$ makes the resampling strategy inefficient, but using an elitist population can help. This section shows that non-elitist EAs also find the optimum in expected polynomial time if using an appropriate parameter setting. We also demonstrate a mutation rate *error threshold* as a function of the noise level in the symmetric noise model. Optimisation is efficient if the mutation rate is above the error threshold; otherwise inefficient.

3.3.4.1 Efficient Optimisation

Theorem 3.3.6 states that the 2-tournament EA with any mutation rate can optimise on noisy OneMax and LeadingOnes functions, for the noise level $q < 1/2$. Theorem 3.3.7 states that the (μ, λ) EA can optimise in $O(n\lambda)$ time under all level noise if we set a sufficiently large population size, i.e., $\lambda \in \Omega(\log(n))$, a sufficiently large selective pressure as measured by the reproductive rate, i.e., $\lambda/\mu > 1/((1-q)\zeta)$, and a low mutation rate $\chi/n \in \left(0, \ln\left(\frac{(1-q)\lambda}{(1+\delta)\mu}\right)/n\right)$ for any constant $\zeta, \delta \in (0,1)$ and $\chi \in \Omega(1)$. This is due to insufficient selective pressure in 2-tournament selection in the high-level symmetric noise model. If we increase the tournament size, i.e., select $k > 2$ candidate individuals from P_t in Algorithm 5, it could be possible to optimise efficiently in such high level noisy functions, because the selective pressure of the non-elitist EA with k tournament selection is approximately k (Lehre and Yao, 2012).

Theorem 3.3.6. *For any constant $q \in [0, 1/2)$, and $C \in \mathbb{R}$ and any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 1/2 - q$, the 2-tournament EA with mutation rate χ/n and population size $\lambda >$*

$c\log(n)$ for a sufficiently large constant c achieves the optimum on OneMax *and* LeadingOnes *in the symmetric noise model (C, q) in expected time $O(\lambda n \log(1/\chi) + n\log(n)/\chi)$ and $O(n\lambda \log(n/\chi) + n^2/\chi)$ respectively.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.3.6. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$. By $q \in [0, 1/2)$, we obtain $0 < \theta \leq \frac{1}{2}$ which satisfies the assumption in Theorem 3.2.1.

By case (f) of Lemma 3.2.1, we get $\Pr(f^n(x_1) > f^n(x_2)) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2)) = 1/2+\theta$, then condition (C2) of Theorem 3.2.1 holds.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that no 1-bits is flipped and one of the 0-bits is flipped after mutation for OneMax, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_j) &> \left(1-\frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\ &\geq e^{-\chi}(1-o(1))(n-j)\chi/n =: h_j \in \Omega((n-j)\chi/n).\end{aligned}$$

For LeadingOnes, we only consider the case that the first 0-bit is flipped and other bits are not flipped, then by Lemma A.2.8,

$$\Pr(y \in A_{\geq j+1} \mid z \in A_j) \geq \left(1-\frac{\chi}{n}\right)^{n-1} \frac{\chi}{n} \geq e^{-\chi}(1-o(1))\frac{\chi}{n} =: h_j = \Omega(\chi/n).$$

Then, we get $h_j \in \Omega((n-j)\chi/n)$ and $h_j \in \Omega(\chi/n)$ for OneMax and LeadingOnes respectively.

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a constant and we know that $\min\{h_j\} \in \Omega(\chi/n)$ for both problems, then

$$\lambda > \frac{4(1+o(1))}{\theta^2\xi(1-\zeta)^4} \ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4 \min\{h_j\}}\right) = O(\log(n/\chi)).$$

Condition (C3) is satisfied by $\lambda \geq c\log(n/\chi)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time on OneMax to reach the optimum is no more than

$$\begin{aligned}
E[T] &\leq \frac{16(1+o(1))}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right) \\
&= O\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{n}{(n-j)\chi}\right) + \sum_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right) \\
&= O\left(\lambda\ln\left(\prod_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right) + n\sum_{j=0}^{m-1}\frac{1}{(n-j)\chi}\right) \\
&= O\left(\lambda\ln\left(\frac{n^n}{n!\chi^n}\right) + n\log(n)/\chi\right)
\end{aligned}$$

using the lower bound $n! > (n/e)^n$,

$$= O\left(\lambda n\log(1/\chi) + n\log(n)/\chi\right).$$

and on LeadingOnes,

$$\begin{aligned}
E[T] &\leq \frac{16(1+o(1))}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right) \\
&= O\left(\lambda\sum_{j=0}^{m-1}\ln(n/\chi) + \sum_{j=0}^{m-1} n/\chi\right) \\
&= O\left(n\lambda\log(n/\chi) + n^2/\chi\right).
\end{aligned}$$

□

Theorem 3.3.7. *For any constant $q \in [0, 1)$, and $C \leq 0$, any constant $\delta \in (0, 1)$ and any $\chi \in \left(0, \ln\left(\frac{(1-q)\lambda}{(1+\delta)\mu}\right)\right)$ and $\chi \in \Omega(1)$, the (μ, λ) EA with mutation rate χ/n, population*

size $\lambda > c\log(n)$ *for a sufficiently large constant* c *and* $(1-q)\lambda/(1+\delta) > \mu \in \Omega(\log(n))$ *achieves the optimum on* OneMax *and* LeadingOnes *in the symmetric noise model* (C, q) *in expected time* $O(\lambda n + n\log(n))$ *and* $O(n\lambda\log(n) + n^2)$ *respectively.*

Proof. We use the level-based theorem (Theorem 2.2.1) to prove Theorem 3.3.7. We use the partition $A_j := \{x \in \{0,1\}^n | f(x) = j\}$ for all $j \in [0..n]$ and we define $\gamma_0 := \mu/\lambda$.

We first show that condition (G2) of Theorem 2.2.1 holds. If the current level is $j \leq n-2$, then there are at least $\gamma\lambda$ individuals in level $j+1$. Let $\varepsilon := \delta^2/(1+\delta)$ be a constant. We assume that there are at least $\gamma(1-q)(1-\varepsilon)\lambda$ individuals which are in level $j+1$ and there is no noise when evaluating these individuals. We will verify this assumption later. Let z be the selected individual and y be the individual after mutating z, then

$$\begin{aligned}\Pr(y \in A_{\geq j+1}) &\geq \Pr(z \in A_{\geq j+1}) \cdot \Pr(y \in A_{\geq j+1} | z \in A_{\geq j+1}) \\ &\geq \frac{\gamma\lambda(1-\varepsilon)(1-q)}{\mu}\left(1-\frac{\chi}{n}\right)^n\end{aligned}$$

by Lemma A.2.5,

$$\geq \frac{\gamma\lambda(1-\varepsilon)(1-q)}{\mu}e^{-\chi}\left(1-\frac{\chi^2}{n}\right)$$

by $\chi < \ln\left(\frac{(1-q)\lambda}{(1+\delta)\mu}\right)$,

$$> \gamma(1-\varepsilon)(1+\delta)(1-o(1))$$

by $\varepsilon = \delta^2/(1+\delta)$,

$$\begin{aligned}&= \gamma(1+\delta-\delta^2)(1-o(1)) \\ &= \gamma\left(1+\left(\delta-\delta^2\right)(1-o(1))\right)\end{aligned}$$

To verify condition (G1), we need to estimate the probability of sampling individuals beyond the current level of the population if there are at least $\gamma_0\lambda$ individuals in level

$j \in [0..n-1]$. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. We only consider the case that no 1-bit is flipped and one 0-bit is flipped after mutation for OneMax, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1}) &= \Pr(z \in A_{\geq j}) \cdot \Pr(y \in A_{\geq j+1} \mid z \in A_j) \\ &> \left(1 - \frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\ &\geq e^{-\chi}\left(1 - o(1)\right)(n-j)\chi/n =: z_j \in \Omega\left((n-j)\chi/n\right).\end{aligned}$$

For LeadingOnes, we only consider the case that the first 0-bit is flipped and no other bit is flipped, then by Lemma A.2.8,

$$\begin{aligned}\Pr(y \in A_{\geq j+1}) &= \Pr(z \in A_{\geq j}) \cdot \Pr(y \in A_{\geq j+1} \mid z \in A_j) \\ &\geq \left(1 - \frac{\chi}{n}\right)^{n-1} \frac{\chi}{n} \\ &\geq e^{-\chi}\left(1 - o(1)\right)\frac{\chi}{n} =: z_j = \Omega\left(1/n\right).\end{aligned}$$

Then, we get $z_j \in \Omega\left((n-j)/n\right)$ and $z_j \in \Omega\left(1/n\right)$ for OneMax and LeadingOnes respectively.

Then we compute the population size required by condition (G3). We know that $\min\{z_j\} \in \Omega\left(\chi/n\right)$ for both problems, then

$$\lambda > \frac{4\lambda}{\left((\delta - \delta^2)\left(1 - o(1)\right)\right)^2 \mu} \ln\left(\frac{128(n+1)}{\left((\delta - \delta^2)\left(1 - o(1)\right)\right)^2 \min\{z_j\}}\right) = O\left(\log(n)\right).$$

Condition (G3) is satisfied by $\lambda \geq c\log(n)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 2.2.1 hold and the expected time on OneMax to reach

the optimum is no more than

$$t_0(n) \leq \frac{8}{(\delta-\delta^2)(1-o(1))^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6\left((\delta-\delta^2)(1-o(1))\right)^2\lambda}{4+\left((\delta-\delta^2)(1-o(1))\right)^2\lambda z_j}\right)+\sum_{j=0}^{m-1}\frac{1}{z_j}\right)$$
$$= O\left(\lambda n + n\log(n)\right),$$

and on LeadingOnes,

$$= O\left(n\lambda\log\left(n\right)+n^2\right).$$

Now we verify the assumption that there are at least $\gamma(1-q)(1-\varepsilon)$ individuals in level $j+1$ and the noise does not affect the ranking if the current level is $j+1$ for any $j \in [n-2]$. We refer to a sequence of $2t_0(n)/\lambda$ generations as a *phase*, and call a phase *good* if for $2t_0(n)/\lambda$ consecutive generations the assumption holds. Let $Z \sim \mathrm{Bin}\left(\gamma\lambda,(1-q)\right)$ be a random variable, which represent the number of individuals not affected by noise in any generation $t \in \mathbb{N}$. By a Chernoff bound, the probability that the assumption holds in a generation is $\Pr\left(Z \leq \gamma\lambda(1-\varepsilon)(1-q)\right) \leq e^{-\Omega(\lambda)}$. By a union bound, a phase is good with probability $1-\left(2t_0(n)/\lambda\right)e^{-\Omega(\lambda)} = \Omega(1)$. By Markov's inequality, the probability of reaching a global optimum in a good phase is at least $1-\Pr\left(T \geq 2t_0(n)\right) \geq 1-\frac{t_0(n)}{2t_0(n)} \geq 1-\frac{1}{2} = \frac{1}{2}$. Hence, the expected number of phases required, each costing $2t_0(n)$ evolutions, is $O(1)$, and the theorem follows.

□

3.3.4.2 Inefficient Optimisation

Non-elitist EAs fail when the mutation rate becomes too high Lehre, 2010. In this section, we investigate what mutation rate is too high for non-elitist EAs in uncertain settings. We use the negative drift theorem for populations to derive what mutation rate make non-elitist

EAs inefficient on OneMax and LeadingOnes (shown in Theorems 3.3.8 and 3.3.9). For 2-tournament selection, there exists a mutation rate error threshold $\ln(2(1-q))/n$. Similarly, we can find a mutation rate error threshold in (μ, λ) selection, which is $\ln((1-q)\lambda/\mu)/n$. Without uncertainty, it is well-known that error thresholds of mutation rate is $\ln(2)/n$ and $\ln(\lambda/\mu)/n$ for the 2-tournament EA and the (μ, λ) EA, respectively (Lehre, 2010; Lehre and Yao, 2012). As we can see from the proofs of Theorems 3.3.8 and 3.3.9, the presence of uncertainties can reduce the maximal reproductive rate of algorithms. Consequentially, the error threshold of the mutation rate would be reduced. We should reduce the mutation rate or increase the selection pressure, e.g., reduce μ in the (μ, λ) EA, as the uncertainty level is increased to ensure efficient optimisation.

Theorem 3.3.8. *For any $C \in \mathbb{R}$ and any constant $q \in [0, 1/2)$, the probability that the 2-tournament EA with any population size $\lambda = \text{poly}(n)$, mutation rate $\chi/n > \ln(2(1-q) + o(1))/n$, optimises OneMax or LeadingOnes in the symmetric noise model (C, q) within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Proof. We estimate the maximal reproductive rate under noise. In each generation of the 2-tournament EA, two individuals x_1 and x_2 are selected uniformly at random from the population by lines 1 and 2 of Algorithm 5, respectively. Then the fittest individual of x_1 and x_2 is chosen by fitness comparison (line 3). However, the presence of noise can lead to a failed comparison, i.e., the worse individual is selected in line 3 of Algorithm 5. we assume without loss of generality $f(x_1) > f(x_2)$. Let E be the event that $f^n(x_1) > f^n(x_2)$ or individual x_1 is selected uniformly from $\{x_1, x_2\}$ if $f^n(x_1) = f^n(x_2)$, then $\Pr(E) = \Pr\left(f^n(x_1) > f^n(x_2)\right) + \frac{1}{2}\Pr(f^n(x_1) = f^n(x_2))$. Then the maximal reproductive rate is the

reproductive rate of the best individual, which is

$$\begin{aligned}\alpha_0 \leq E[R_t(i)] &= \lambda\left(\frac{1}{\lambda^2} + \frac{2\Pr(E)}{\lambda}\left(1-\frac{1}{\lambda}\right)\right)\\ &= \frac{1}{\lambda} + 2\Pr(E)\left(1-\frac{1}{\lambda}\right)\\ &< 2\Pr(E) + \frac{1}{\lambda}\\ &= 2\Pr(E) + o(1)\end{aligned}$$

by Lemma 3.2.1 (f), $\Pr(E) = 1-q$,

$$= 2(1-q) + o(1).$$

Then Theorem 3.3.8 is proved by applying Theorem 2.2.1. □

Theorem 3.3.9. *For any constant $q \in [0,1]$, the probability that the (μ,λ) EA with any population size $\lambda = \mathrm{poly}(n)$, mutation rate $\chi/n > (\ln(1-q)\lambda/\mu)/n$, optimises* OneMax *or* LeadingOnes *in the symmetric noise model (C,q) within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Proof. We first compute the maximal reproductive rate. Let x be the fittest individual in P_t evaluated by $f(x)$. Only if there is no noise in x, x has a chance to be selected with probability $1/\mu$. Then we can compute the maximal reproductive rate $\alpha_0 \leq \frac{\lambda}{\mu}(1-q) + 0 = \frac{\lambda}{\mu}(1-q)$. Finally, Theorem 3.3.9 is proved by applying Theorem 2.2.1. □

3.4 Dynamic Optimisation

Now we consider dynamic optimisation. First, we apply the general theorem for the 2-tournament EA (Theorem 3.2.1) on the DBV problem and derive the runtime and the

parameters required. The proof idea is similar to noisy optimisation in which the critical step is estimating a lower bound of the fitness bias.

Lengler and Schaller (2018) proved that the (1+1) EA can achieve the optimum in $O\big(n\log(n)\big)$ with standard mutation rate $\chi/n = 1/n$ on the noisy linear function which is a general case of the DBV problem. However, there only exists a partial result for population-based EAs, i.e., runtime when the population is initiated close to the optimum (Lengler and Meier, 2020). Theorem 3.4.1 gives for the first time the runtime from any start point on DBV for a population-based EA. It implies that if choosing a low mutation rate, e.g., $\chi/n = \zeta/(2n^2)$ and a population size $\lambda > cn^2\log(n)$ for a sufficiently large constant c, the 2-tournament EA can optimise the DBV problem in $O\left(n^3\lambda\log(n)\right)$ time. The analysis could be further improved by estimating the maximal Hamming-distance in all pairs of individuals more precisely.

Theorem 3.4.1. *For any $\chi \in (0, \ln(1+2\theta))$, where $\theta := 1/(2n)$, the 2-tournament EA with mutation rate χ/n and population size $\lambda > cn^2\log(n)$ for a large enough constant c achieves the optimum on DBV in expected time $O\left(n^3\lambda\log(1/\chi) + n^3\log(n)/\chi\right)$.*

Proof. We apply Theorem 3.2.1 to prove Theorem 3.4.1. If $\chi \in (0, \ln(1+2\theta))$, there exists a constant $\zeta \in (0,1)$ such that $\chi \in (0, \ln(1+2\theta\zeta))$, which satisfies the condition in Theorem 3.2.1. We first partition the search space into levels. We use the partition $A_j := \{x \in \{0,1\}^n | \mathrm{OM}(x) = j\}$ for $j \in [0..n]$. It is easy to see that θ satisfies the assumption in Theorem 3.2.1.

We first show that condition (C2) of Theorem 3.2.1 holds. Let x_1 and x_2 be two individuals in $A_{\geq j+1}$ and $A_{\leq j}$ respectively, where $j \in [0..n-2]$. Let E be the event that $f^t(x_1) > f^t(x_2)$ or individual x_1 is selected uniformly from $\{x_1, x_2\}$ if $f^t(x_1) = f^t(x_2)$. The probability of this event is $\Pr(f^t(x_1) > f^t(x_2)) + \frac{1}{2}\Pr(f^t(x_1) = f^t(x_2)) = \Pr(E)$.

To estimate a lower bound for $\Pr(E)$ on DBV, we pessimistically assume that $x_1 \in A_{j+1}$

and $x_2 \in A_{\leq j}$, such that $\mathrm{OM}(x_1) = \mathrm{OM}(x_2) + h$ where $h \in [1, j]$. We assume $\mathrm{H}(x_1, x_2) \leq l + l + h = s$, where $s \leq n$ and there exist l bit-positions that x_1 has a 1-bit and x_2 has a 0-bit, and there exist another l bit-positions that x_1 is with 0-bit and x_2 has a 1-bit, such that x_1 and x_2 have the same bit in the rest of $n - 2l - h$ positions. For the DBV problem, the coefficients vary exponentially, thus the largest coefficient is the deciding factor for the fitness value. We first compare the fitness in the $n - 2l - h$ positions of x_1 and x_2, which are the same in the same generation. The next largest coefficient decides the final fitness value. Then we say that x_1 "wins" if the event E happens. If the next largest coefficient is in the $l + h$ positions, x_1 wins, else in another l positions, x_2 wins. Therefore,

$$\begin{aligned}
\Pr(E) &\geq \frac{l+h}{2l+h} \\
&= \frac{l + h/2 + h/2}{2\,(l + h/2)} \\
&= \frac{1}{2} + \frac{h}{2(2l+h)} \\
&\geq \frac{1}{2} + \frac{1}{2(2l+h)} \\
&= \frac{1}{2} + \frac{1}{2s}
\end{aligned}$$

since the Hamming-distance s between any pair of individuals is at most n, then

$$\geq \frac{1}{2} + \theta.$$

Condition (C2) of Theorem 3.2.1 holds. Since $s \leq n$, we get the fitness bias $\theta \geq 1/(2n)$.

To verify condition (C1), we need to estimate the probability of sampling individuals beyond the current level of the population. We assume that there is an individual $z \in A_j$ where $j \in [0..n-1]$, and let y be obtained from z by the mutation operator with mutation rate χ/n. For a lower bound, it suffices to only consider the case that none of the 1-bits are

flipped and one of 0-bits is flipped after mutation. Then, by Lemma A.2.8 it follows,

$$\begin{aligned}\Pr(y \in A_{\geq j+1} \mid z \in A_j) &> \left(1-\frac{\chi}{n}\right)^j \frac{\chi}{n}(n-j) \\ &\geq e^{-\chi}\left(1-o(1)\right)(n-j)\chi/n =: h_j \\ &= \Omega\left(\frac{(n-j)\chi}{n}\right).\end{aligned}$$

Then we compute the population size required by condition (C3). Let $\xi \in (0, 1/16)$ be a constant, then

$$\lambda > \frac{4\left(1+o(1)\right)}{\theta^2\xi(1-\zeta)^4}\ln\left(\frac{128(m+1)}{\theta^2\xi(1-\zeta)^4\min\{h_j\}}\right) = O\left(n^2\log(n/\chi)\right).$$

Condition (C3) is satisfied by $\lambda > cn^2\log(n/\chi)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 3.2.1 hold and the expected time $E[T]$ to reach the optimum is no more than

$$\begin{aligned}E[T] &\leq \frac{16\left(1+o(1)\right)}{\theta^2\xi(1-\zeta)^2}\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{6}{\xi(1-\zeta)^2 h_j}\right) + \frac{1}{\xi(1-\zeta)^2}\sum_{j=0}^{m-1}\frac{1}{h_j}\right) \\ &= O\left(s^2\left(\lambda\sum_{j=0}^{m-1}\ln\left(\frac{n}{(n-j)\chi}\right) + \sum_{j=0}^{m-1}\frac{n}{(n-j)\chi}\right)\right) \\ &= O\left(s^2\left(\lambda\ln\left(\frac{n^n}{\chi^n n!}\right) + n\log(n)/\chi\right)\right)\end{aligned}$$

using the bounds $n! > (n/e)^n$, and $s \leq n$,

$$\begin{aligned}&= O\left(s^2 n\lambda\log(1/\chi) + s^2 n\log(n)/\chi\right) \\ &= O\left(n^3\lambda\log(1/\chi) + n^3\log(n)/\chi\right).\end{aligned}$$

□

3.5 Conclusion

This chapter has derived runtime results for non-elitists EAs with 2-tournament and (μ, λ) selection on two well-known benchmark functions, i.e., OneMax and LeadingOnes in uncertain environments. For the one-bit, the bit-wise and the Gaussian noise models, we improved and extended results of the non-elitist EA with 2-tournament from Dang and Lehre (2015). We introduced the notion of fitness bias which indicates the probability that the truly fitter individual is selected. We summarised fitness biases for 2-tournament selection in some noisy scenarios in Lemma 3.2.1. Then we got more precise upper bounds for the expected runtimes and also provide more precise guidance on how to choose the mutation rate and the population size as a function of the level of noise. From Tables 2.3-2.10, we concluded that by using an appropriate mutation parameter, i.e., $\chi \in [\theta, \ln(1+2\theta))$ where θ is a function of the level of noise, and a sufficiently large population size, the 2-tournament EA optimises OneMax and LeadingOnes in less time in expectation under one-bit and extremely high-level bit-wise noise, than the (1+1) EA using a resampling strategy (Qian et al., 2019; Qian et al., 2018). In some settings, such as in the Gaussian noise model, we obtained a lower upper bound of runtimes than the ACO-fp (Friedrich et al., 2016) and a comparable upper bound with EDAs (Friedrich et al., 2016; Lehre and P. T. H. Nguyen, 2021; Rowe and Aishwaryaprajna, 2019).

We then, for the first time, studied the performance of non-elitist EAs with two selection mechanisms, i.e., 2-tournament and (μ, λ), on OneMax and LeadingOnes in the symmetric noise model. We also provided for the first time mutation rate error thresholds under the symmetric noise model, which are $\ln(2(1-q))/n$ and $\ln((1-q)\lambda/\mu)/n$ for the 2-tournament and the (μ, λ) selection, respectively. The noise essentially affects the maximal reproductive rate and the error threshold of a non-elitist population. Furthermore, in these scenarios, non-elitist EAs can outperform the known best results, i.e., for the (1+λ) EA

and the (μ+1) EA. Finally, we proved for the first time that with appropriate parameter settings, non-elitist EAs can optimise the DBV problem in expected polynomial time.

In overall, we provide advice on how to choose the mutation rate, the selective pressure and the population size (see Theorems 3.3.1-3.3.9), for non-elitist EAs on some uncertain scenarios.

Chapter Four

Self-adaptation in Noisy Environments

Authors: Per Kristian Lehre and Xiaoyu Qin

This chapter is based on the following publication:

Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms (Lehre and Qin, 2023b) which is published by the 17-th ACM/SIGEVO Workshop on Foundations of Genetic Algorithms (FOGA'23).

4.1 Introduction

Real-world optimisation often involves uncertainty, such as noise. The exact fitness value of a search point may not be determined due to noise. As outlined in Chapter 3, we have established that non-elitist EAs can effectively tolerate high uncertainties. This tolerance is achieved when conditions of a sufficiently high selective pressure (reproductive rate) and a sufficiently high population size, and a sufficiently low mutation rate, all relative to the level of uncertainty. It is challenging to find the appropriate parameter setting if the occurrence of noise is unpredictable (or the noise level is unknown). Self-adaptation as a parameter control mechanism can help to configure the mutation rate in the noise-unknown environments. However, the rigorous analysis of self-adaptation to noise is missing. Runtime analysis of non-elitist population-based EAs can be challenging. Clearly, including self-adaptation and noise makes the analysis even harder.

The main contribution of this chapter is the first theoretical analysis of self-adaptive EAs in noisy environments. The rigorous runtime analysis on the LeadingOnes problem shows that the 2-tournament EA with self-adaptation from high/low mutation rates (SA-2mr) can guarantee the lowest expected runtime among the fixed high/low mutation rates and the uniformly chosen mutation rate from high/low rates (uniformly mixing mutation rate, UM-2mr), regardless of the presence of symmetric noise. The results are summarised in Table 4.1. In addition, we extend to more types of noise, one-bit and bit-wise noise, and a more natural self-adaptation mechanism that adapts the mutation rate from a given interval $(0, 1/2]$ (SA) in the empirical study. The experimental results show that self-adaptive EAs can adapt to noise levels and outperform static EAs.

The chapter is organised as follows: Section 4.2 introduces algorithms. Sections 4.4-4.5 show limitations of using high/low and uniformly mixing mutation rates under symmetric noise. Section 4.6 analyses the runtime of the 2-tournament EA with self-adapting mutation

Table 4.1: Theoretical results of EAs on LeadingOnes under symmetric noise (C, q) ($C \in \mathbb{R}$, constant $0 < \chi_{\text{high}} < \ln(2)$, $\chi_{\text{low}} = a/n$, $\lambda = c\log(n)$ where $a, c > 0$ are constants, $p_c \in o(1) \cap \Omega(1/n)$ in 2-tour' EA with SA-2mr)

Algorithm	**Noise-free**	**Under Noise**
(1+1) EA	$O(n^2)$ (Droste et al., 2002)	$e^{\Omega(n)}$ † (Theorem 4.3.1)
2-tour' EA with $\frac{\chi_{\text{high}}}{n}$	$O(n^2)$ (Corus et al., 2018)	$e^{\Omega(n)}$ ‡ (Theorem 3.3.8)
2-tour' EA with $\frac{\chi_{\text{low}}}{n}$	$\Omega(n^2 \log(n))$ (Corollary 4.4.1)	$O(n^3)$ § (Theorem 3.3.6)
2-tour' EA with UM-2mr	$O(n^2)$ (Theorem 4.5.1)	$e^{\Omega(n)}$ ‡ (Theorem 4.5.2)
2-tour' EA with SA-2mr	$O(n^2)$ (Theorem 4.6.1)	$O(n^3)$ § (Theorem 4.6.2)

rates with/without noise. Section 4.7 shows empirical results. The paper concludes in Section 4.8.

The main contribution of this chapter is the first theoretical analysis of self-adaptive EAs in the noisy environments. The rigorous runtime analysis on the LeadingOnes problem shows that the 2-tournament EA with self-adapting from high/low mutation rates (SA-2mr) can guarantee the lowest runtime among the fixed high/low mutation rates and the uniformly chosen mutation rate from high/low rates (uniformly mixing mutation rate, UM-2mr), regardless of the presence of symmetric noise. The results are summarised in Table 4.1. In addition, we extend to more types of noise, one-bit and bit-wise noise, and a more natural self-adaptation mechanism that adapts the mutation rate from a given interval $(0, 1/2]$ (SA) in the empirical study. The experimental results show that self-adaptive EAs can adapt to noise levels and outperform static EAs.

†For any constant noise level $q \in (0.127107, 1/2)$.

‡For some constant noise level $q \in (0, 1/2)$.

§For all constant noise level $q \in (0, 1/2)$.

Algorithm 13 2-tournament EA with self-adaptation

Require: Pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$, where $n \in \mathbb{N}$

Require: Population size $\lambda \in \mathbb{N}$.

Require: Sorting partial order $\succeq_{P,f}$.

Require: Self-adapting mutation rate strategy $D_{\text{mut}} : (0, 1/2] \to \Omega \to (0, 1/2]$, where Ω is some sample space.

Require: Initial self-adaptive population $P_0 \in \mathcal{Y}^\lambda$.

1: **for** $\tau = 0, 1, 2, ...$ until termination condition met **do**
2: **for** $i = 1$ to λ **do**
3: $(x_1, \chi_1/n) \leftarrow P_\tau(i_1)$ where $i_1 \backsim \text{Uniform}([\lambda])$.
4: $(x_2, \chi_2/n) \leftarrow P_\tau(i_2)$ where $i_2 \backsim \text{Uniform}([\lambda])$.
5: **if** $(x_1, \chi_1/n) \succeq_{P_\tau,f} (x_2, \chi_2/n)$ **then**
6: $(z, \chi/n) \leftarrow (x_1, \chi_1/n)$,
7: **else**
8: $(z, \chi/n) \leftarrow (x_2, \chi_2/n)$.
9: Sample $\chi'/n \sim D_{\text{mut}}(\chi/n)$.
10: $P_{\tau+1}(i) \leftarrow (y, \chi'/n)$ where y created by mutating z with mutation rate χ'/n.

4.2 Algorithms

In this chapter, we consider three different mutation rate strategies: static, uniformly mixing (UM-2mr) and self-adaptive (SA-2mr and SA). The static 2-tournament EA is delineated as Algorithm 3 utilising Algorithm 6. In generation τ, we obtain a new individual by selection and mutation. For the 2-tournament with UM-2mr (Dang and Lehre, 2016b), a mutation rate χ/n uniformly sampled from two rates is used in each mutation in stead of using only one rate.

In pursuit of a fair comparison, this chapter takes into consideration self-adaptive EAs

Algorithm 14 Self-adapting two mutation rates (SA-2mr) (Dang and Lehre, 2016b)

Require: Two rates $\chi_{\text{high}} > \chi_{\text{low}} > 0$.

Require: Switch probability $p_{\text{c}} \in (0, 1)$.

Require: Mutation rate χ/n.

1: Set χ'/n:=$\begin{cases} \chi_{\text{high}}/n \text{ if } \chi = \chi_{\text{low}},\ \chi_{\text{low}}/n \text{ if } \chi = \chi_{\text{high}} & \text{with probability } p_{\text{c}}, \\ \chi/n & \text{otherwise.} \end{cases}$

2: **return** χ'/n.

with 2-tournament selection. The self-adaptive EA framework is already illustrated in Algorithm 7, which requires population sorting prior to selection. As highlighted in Section 2.2.5, the reevaluation strategy is commonly used for analysis of 2-tournament selection algorithm. This approach ensures independence for each comparison in tournament, which simplifies the analysis. Therefore, we reformulate the self-adaptive EAs using 2-tournament selection, as depicted in Algorithm 13. In this chapter, the comparison in a self-adaptive population is based on the fitness-first sorting partial order which prefers a high mutation rate, as defined in Definition 2.2.4 (b) and applied in (Case and Lehre, 2020). To self-adapt the mutation rate, we can utilise self-adapting mutation rate strategy in (Case and Lehre, 2020) (Algorithm 10), with set $b := 1/A$. We describe the 2-tournament EA with SA as Algorithm 13 using this strategy (Algorithm 10).

Although some studies exist on self-adapting mutation rate from the given interval $(0, 1/2]$ (Case and Lehre, 2020; B. Doerr et al., 2021), the involvement of noise can make runtime analysis more challenging. Therefore, we consider a simplified version in the runtime analysis, the 2-tournament EA with SA-2mr, which only self-adapts two mutation rates $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$. In contrast, we illustrate that the fixed high/low mutation rates and the uniformly mixing mutation rate cannot be fast or efficient in the noisy setting. For the sake of analysis, we re-define the self-adaptive population in this algorithm: $P_\tau \in \mathcal{Y}^\lambda$, where $\mathcal{Y} = \{0, 1\}^n \times \{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$. We apply a simple self-adapting mutation rate

strategy D_{mut}, in which the mutation rate switches to the other value with a probability p_{c}(self-adaptive parameter) (Algorithm 14). Thus, the 2-tournament EA with SA-2mr can be described as Algorithm 13 using Algorithm 14. Note that a similar two-rate self-adaptive EA was studied in (Dang and Lehre, 2016b). However, they employed a ranking rule based solely on fitness values, which can constrain the optimisation speed, as low mutation rate individuals may dominate the population preventing the necessary exploration for faster convergence.

4.3 Analysed Noise Models

In the theoretical study, we focus on the symmetric noise model, a well-established noise model extensively investigated in (Qian et al., 2021) and Chapter 3. This choice is motivated by existing runtime analyses of EAs that demonstrate the ineffectiveness of resampling strategies for successful optimisation in the symmetric noise model. In contrast, employing a population has been shown to enhance robustness in this context. Nevertheless, for non-elitist EAs, attaining successful noisy optimisation requires precise tuning of the mutation rate in relation to the noise level. Expanding upon previous research, our theoretical study investigates the potential for further enhancing robustness through the self-adapting mutation rates in the scenario where the presence of symmetric noise is unknown.

We revisit some earlier results related to the symmetric noise model. For static 2-tournament EAs, two theorems related to the LeadingOnes problem in the symmetric noise model have been established in Chapter 3. Theorem 3.3.8 identifies the mutation rate which leads to a inefficient optimisation for a given noise level, while Theorem 3.3.6 reveals the appropriate mutation rate for efficient optimisation. Additionally, for the sake of completeness in our research, we introduce Theorem 4.3.1, which demonstrates that the (1+1) EA

is inefficient under high-level symmetric noise, adapted from Theorem 20 in (Gießen and Kötzing, 2016)).

Theorem 4.3.1. *For any $C \in \mathbb{R}$ and any constant $q \in (0.127107, 1/2)$, the probability of the (1+1) EA optimising* LeadingOnes *in the symmetric noise model (C, q) in $e^{\Omega(n)}$ runtime is $e^{-\Omega(n)}$.*

Proof. To prove Theorem 4.3.1, we can modify the proof of Theorem 20 in (Gießen and Kötzing, 2016), which states the lower bound of the (1+1) EA on LeadingOnes under one-bit noise. We need to re-estimate the lower bound of event E_1 in the symmetric noisy model instead of in the one-bit noise model. By the assumption of E_1 that the offspring x' only flips exactly one 1 in the right half of the positions, we know $f(x) \geq f(x')$. We distinguish three cases to estimate $\Pr(E_1)$:

1. $C < f(x) + f(x')$: The probability of accepting the offspring x' is at least $(1-q)q$, where there is no noise in x and noise in x'.
2. $C > f(x) + f(x')$: The probability of accepting the offspring x' is at least $q(1-q)$, where there is noise in x and no noise in x'.
3. $C = f(x) + f(x')$: The probability of accepting the offspring x' is at least $2q(1-q)/2$, where there is noise in either x_1 or x_2 ($f^n(x) = f^n(x')$, so with probability $1/2$).

Therefore, the lower bound of the probability of event E_1 is $\Pr(E_1) \geq 49/(100e) \cdot q(1-q) > 1/50$ for any constant $q \in (0.127107, 1/2)$. The rest of the proof is the same as the proof of Theorem 20 in (Gießen and Kötzing, 2016). □

In our empirical study, we expand the analysis to include two widely studied models, one-bit noise and bit-wise noise, as introduced in Definitions 2.2.12 and 2.2.13, respectively.

4.4 High/Low Mutation Rates Lead to Failed/Slow Optimisation

From Chapter 3, we know that the non-elitist EAs should reduce the mutation rate to handle noise. However, too low mutation rates lead to a slow optimisation in the noise-free environment. In this section, we show that the static 2-tournament EAs using the high or low mutation rate cannot be efficient if the presence of noise is unknown. We say the high mutation rate is $\frac{\chi_{\text{high}}}{n}$ where the mutation parameter $\chi_{\text{high}} > 0$ is a constant, and the lower mutation rate is $\frac{\chi_{\text{low}}}{n}$ where $\chi_{\text{low}} = a/n$ for some constant $a > 0$.

It is well-known that the 2-tournament EA with mutation rate χ/n with $\chi < \ln(2)$ is a constant, and population size $\lambda = c\log(n)$ for a sufficiently large constant c, achieves the optimum of LeadingOnes without noise in expected runtime $O(n^2)$ (Corus et al., 2018). However, the algorithm can fail in noisy environments if using a constant mutation parameter, i.e., χ_{high}. From Theorem 3.3.8, we know that for any constant mutation parameter $\chi > 0$, we can find some constant noise level $q \in [0, 1/2)$ such that the 2-tournament EA using any population size $\lambda = \text{poly}(n)$ optimises LeadingOnes under symmetric noise within e^{cn} generations with probability $e^{-\Omega(n)}$ where $c > 0$ is a constant.

We can use a sufficiently low mutation rate against noise, e.g., $\frac{\chi_{\text{low}}}{n}$. From Theorem 3.3.6, we know that the expected runtime of the 2-tournament EA with mutation rate χ_{low}/n and population size $\lambda = c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes under symmetric noise for any constant noise level $q \in [0, 1/2)$ is $O(n^3)$. However, such a low mutation rate slows down the noise-free optimisation by a small but super-constant factor, i.e., $\Omega(n^2 \log(n))$ runtime instead of $O(n^2)$ guaranteed by using $\frac{\chi_{\text{high}}}{n}$, which is indicated by Corollary 4.4.1 via Theorem 2.2.3.

Corollary 4.4.1. *The expected runtime of the 2-tournament EA using mutation parameter*

satisfying $\chi \geq 2^{-n/3}n$ *and* $\chi \in O(1/n)$ *on* LeadingOnes *is* $\Omega(n^2 \log(n))$.

4.5 Uniformly Mixing Mutation Rates Do Not Help under Noise

In this section, we show runtime analysis results on the 2-tournament EA with uniformly mixing high/low mutation rates (UM-2mr) under noise. Theorems 4.5.1-4.5.2 present that using UM-2mr can optimise the noise-free LeadingOnes function in expected runtime $O(n^2)$, but can fail under symmetric noise with a high probability.

Theorem 4.5.1. *For any constants* $\chi_{\text{high}}, a > 0$ *and* $\chi_{\text{low}} = a/n$*, the expected runtime of the 2-tournament EA with UM-2mr from* $\{\chi_{\text{high}}/n,\ \chi_{\text{low}}/n\}$ *and population size* $\lambda > c\log(n)$ *for a sufficiently large constant* c *on optimising* LeadingOnes *is* $O\left(n\lambda \log\left(n\right) + n^2\right)$.

Proof. We apply the level-based theorem (Theorem 2.2.1) with respect to a partitioning of the search space $\{0,1\}^n$ into the following $n+1$ levels: $A_j := \{x \mid \text{LO}(x) = j\}$ for all $j \in [0..n]$.

To verify condition (G2) of Theorem 2.2.1, we assume that at least $\gamma\lambda$ individuals in level $A_{\geq j+1}$, where $\gamma \in (0, \gamma_0)$, $j \in [0..n-2]$. The lower bound of the probability of the offspring $y \in A_{\geq j+1}$ can be estimated by selecting an individual from $A_{\geq j+1}$ and flipping no bit.

$$
\begin{aligned}
&\Pr\left((y, \chi'/n) \in A_{\geq j+1}\right) \\
&= \Pr\left((z, \chi/n) \in A_{\geq j+1}\right) \Pr\left((y, \chi'/n) \in A_{\geq j+1} \mid (z, \chi/n) \in A_{\geq j+1}\right) \\
&\geq (\gamma^2 + 2\gamma(1-\gamma)) \left(\frac{1}{2}\left(1 - \frac{\chi_{\text{high}}}{n}\right)^n + \frac{1}{2}\left(1 - \frac{\chi_{\text{low}}}{n}\right)^n\right) \\
&\geq 2\gamma(1-\gamma) \left(\frac{1}{2}\left(1 - \frac{\chi_{\text{high}}}{n}\right)^n + \frac{1}{2}\left(1 - \frac{\chi_{\text{low}}}{n}\right)^n\right)
\end{aligned}
$$

by Lemma A.2.5,

$$= \gamma(1-\gamma)\left(e^{-\chi_{\text{high}}} + e^{-\chi_{\text{low}}}\right)(1-o(1))$$

since $\chi_{\text{low}} \in o(1)$, $e^{-\chi_{\text{low}}} = 1 - o(1)$, then

$$= \gamma(1-\gamma)\left(e^{-\chi_{\text{high}}} + 1\right)(1-o(1))$$

let constant $\delta := e^{-\chi_{\text{high}}} > 0$, then

$$= \gamma(1-\gamma)(1+\delta)(1-o(1))$$
$$\geq \gamma(1-\gamma_0)(1+\delta)(1-o(1))$$

let $\gamma_0 := \delta/2$ and let $\Delta := \frac{\delta(1-\delta)}{4} > 0$, then

$$= \gamma(1-\delta/2)(1+\delta)(1-o(1))$$
$$\geq \gamma(1+2\Delta)(1-o(1))$$
$$\geq \gamma(1+\Delta).$$

To verify condition (G1), we estimate the probability of sampling individuals beyond the current level of the population if there are at least $\gamma_0\lambda$ individuals in A_j. The lower bound of this probability can be estimated by selecting an individual in A_j (pessimistically assume that both x_1 and x_2 are from A_j), using $\frac{\chi_{\text{high}}}{n}$ mutation rate and only considering the case that the first 0-bit is flipped and no other bit is flipped, for $j \in [0..n-1]$,

$$\begin{aligned}\Pr((y,\chi'/n) \in A_{\geq j+1}) &\geq \frac{1}{2}\gamma_0^2\left(1-\frac{\chi_{\text{high}}}{n}\right)^{n-1}\frac{\chi_{\text{high}}}{n} \\ &\geq \gamma_0^2 e^{-\chi_{\text{high}}}\frac{\chi_{\text{high}}}{n}(1-o(1)) =: z_j = \Omega\left(\frac{1}{n}\right)\end{aligned}$$

Then we compute the population size required by condition (G3) $\lambda \geq \frac{4}{\gamma_0\Delta^2}\ln\left(\frac{128(n+1)}{\min\{z_j\}\Delta^2}\right) = O(\log(n))$. Condition (G3) is satisfied by $\lambda \geq c\log(n)$ for a sufficiently large constant c.

Overall, the expected runtime is no more than

$$\begin{aligned} E[T] &\leq \frac{8}{\Delta^2} \sum_{j=0}^{n-1} \left(\lambda \ln \left(\frac{6\Delta\lambda}{4 + z_j \Delta\lambda} \right) + \frac{1}{z_j} \right) \\ &\leq \frac{8}{\Delta^2} \left(\lambda(n-1) \ln \left(\frac{6}{\min\{z_j\}} \right) + \frac{n-1}{\min\{z_j\}} \right) \\ &= O(n\lambda \log(n) + n^2). \end{aligned}$$

□

Now we prove that using UM-2mr can fail under symmetric noise with a high probability.

Theorem 4.5.2. *For any constant $q \in (0, 1/2)$ and any constant $\delta \in (0, q)$, the probability that the 2-tournament EA with UM-2mr from $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ where constants $\chi_{\text{high}} \geq \ln\left(\frac{1-q}{q-\delta}\right) > \chi_{\text{low}} > 0$ with any population size $\lambda \in \text{poly}(n)$ optimises LeadingOnes in the symmetric noise model (C, q), where $C \in \mathbb{R}$, within time e^{cn} is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Proof. We use Theorem A.3.1 to prove Theorem 4.5.2. To estimate the upper bound of $E[R_t(i)]$, we compute the expected number of offspring of the fittest individual $\hat{x}$ in generation t. To select $\hat{x}$, there are two cases: (1) the algorithm selects $\hat{x}$ twice, or (2) selects $\hat{x}$ and one of the other individuals x and selects $\hat{x}$ even if the noise occurs. For case (2), the probability of a successful comparison S, i.e, $\hat{x}$ is exactly selected, is $\Pr(S) = \Pr(f^n(\hat{x}) > f^n(x)) + \frac{1}{2}\Pr(f^n(\hat{x}) = f^n(x)) = 1 - q$, where the last equation is from Lemma 3.2.1 (f). Then,

$$\begin{aligned} E[R_t(i)] &= \lambda \left(\left(\frac{1}{\lambda} \right)^2 + 2(1-q) \frac{1}{\lambda} \left(1 - \frac{1}{\lambda} \right) \right) \\ &= \frac{1}{\lambda} + 2(1-q) \left(1 - \frac{1}{\lambda} \right) < 2(1-q) =: \alpha_0. \end{aligned}$$

Then we estimate the upper bound of $\sum_{j=1}^{m} p_j e^{-\chi_j}$ in condition (3):

$$\sum_{j=1}^{m} p_j e^{-\chi_j} = \frac{1}{2}e^{-\chi_{\text{high}}} + \frac{1}{2}e^{-\chi_{\text{low}}}$$
$$\leq \frac{q-\delta}{2(1-q)} + \frac{1}{2} = \frac{1-\delta}{2(1-q)} = \frac{1-\delta}{\alpha_0}.$$

which by Theorem A.3.1 implies results on LeadingOnes. □

4.6 Self-adapting Mutation Rates Guarantee Efficiency Under Noise

We now analyse the self-adaptive EA using level-based theorems to show their efficiency in noisy and noise-free environments. Theorem 4.6.1 shows that the 2-tournament EA using SA-2mr achieves a comparable performance to using a high mutation rate $\frac{\chi_{\text{high}}}{n}$, i.e., $O(n^2)$ runtime, on the noise-free LeadingOnes function. The proof of Theorem 4.6.1 is conducted by the level-based theorem (Theorem 2.2.1). Theorem 4.6.2 shows that the self-adaptive EA also efficiently optimises under symmetric noise.

Theorem 4.6.1. *For any constant $\chi_{\text{high}} \in (0, \ln(2(1-\delta))]$ where $\delta \in (0, 1/2)$ is any constant, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, and any $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ with self-adaptation parameter p_c and population size $\lambda > c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes without noise is $O(n\lambda\log(n) + n^2)$.*

Proof. We apply the level-based theorem (Theorem 2.2.1) with respect to a partitioning of the state space $\mathcal{Y}$ into the following $n+1$ levels and $2n+1$ sub-levels: $A_j := \{(x, \frac{\chi_{\text{high}}}{n}), (x, \frac{\chi_{\text{low}}}{n}) \mid \text{LO}(x) = j\}$ for all $j \in [0..n]$. For each level, we divide A_j into two sub-levels $A_{(j,2)} :=$

$\{(x, \frac{\chi_{\text{high}}}{n}) \mid \text{LO}(x) = j\}$ and

$$A_{(j,1)} := \begin{cases} \{(x, \frac{\chi_{\text{low}}}{n}) \mid \text{LO}(x) = j\} & \text{if } j \leq n-1 \\ \{(x, \frac{\chi_{\text{high}}}{n}), (x, \frac{\chi_{\text{low}}}{n}) \mid \text{LO}(x) = j\} & \text{if } j = n. \end{cases}$$

To describe the order of two levels $A_{(j,i)}$ and $A_{(j',i')}$, we define $(j,i) > (j',i')$ if $(j > j') \vee (j = j' \wedge i > i')$. For convenience, we also define $(j,1)+1 = (j,2)$ and $(j,2)+1 = (j+1,1)$.

To verify condition (G2) of Theorem 2.2.1, we assume that at least $\gamma\lambda$ individuals are in level $A_{\geq(j,i)+1}$, where $\gamma \in (0, \gamma_0]$, $(j,i) \leq (n-1,1)$ and constant $\gamma_0 \in (0,1)$. We will define γ_0 later. The lower bound of the probability of the offspring $y \in A_{\geq(j,i)+1}$ can be estimated by selecting an individual from $A_{\geq(j,i)+1}$, keeping its mutation rate and flipping no bit.

$$\begin{aligned} &\Pr\left((y, \chi'/n) \in A_{\geq(j,i)+1}\right) \\ &= \Pr\left((z, \chi/n) \in A_{\geq(j,i)+1}\right) \\ &\qquad \cdot \Pr\left((y, \chi') \in A_{\geq(j,i)+1} \mid (z, \chi/n) \in A_{\geq(j,i)+1}\right) \\ &\geq (\gamma^2 + 2\gamma(1-\gamma))(1-p_c)\left(1 - \frac{\chi'}{n}\right)^n \\ &\geq 2\gamma(1-\gamma)(1-p_c)\left(1 - \frac{\chi_{\text{high}}}{n}\right)^n \end{aligned}$$

by Lemma A.2.11 and $p_c \in o(1)$,

$$\geq 2\gamma(1-\gamma_0)\frac{1}{2(1-\delta)}(1-o(1)) = \gamma\frac{1-\gamma_0}{1-\delta}(1-o(1))$$

let $\gamma_0 := \delta/2$ and let $\Delta := \frac{\delta(1-\delta)}{4} > 0$, then

$$= \gamma(1-\delta/2)(1+\delta)(1-o(1)) = \gamma(1+2\Delta)(1-o(1)) \geq \gamma(1+\Delta).$$

To verify condition (G1), we estimate the probability of sampling individuals beyond the current level of the population if there are at least $\gamma_0\lambda$ individuals in $A_{(j,i)}$ for $(j,i) \leq (n-1,1)$. The lower bound of this probability can be estimated by selecting an individual in $A_{(j,i)}$

(here we pessimistically only consider the situation that both x_1 and x_2 are in $A_{(j,i)}$) and only considering two cases:

- $i = 2$: the mutation rate is not changed, and the first 0-bit is flipped, but no other bit is flipped (the probability is $p_2 = (1-p_c)\frac{\chi_{\text{high}}}{n}(1-\chi_{\text{high}}/n)^{n-1} = \Omega(1/n)$),
- $i = 1$: the mutation rate is switched to $\frac{\chi_{\text{high}}}{n}$ and no bit is flipped (the probability is $p_1 = p_c(1-\chi_{\text{high}}/n)^n = \Omega(1/n)$).

Then, $\Pr((y,\chi'/n) \in A_{\geq j+1}) \geq \gamma_0^2 \min\{p_1,p_2\} =: z_{(j,i)} = \Omega\left(\frac{1}{n}\right)$.

Then we compute the population size required by condition (G3) $\lambda \geq \frac{4}{\gamma_0\Delta^2}\ln\left(\frac{128(n+2)}{\min\{z_{(j,i)}\}\Delta^2}\right) = O(\log(n))$. Condition (G3) is satisfied by $\lambda \geq c\log(n)$ for a sufficiently large constant c.

Overall, the expected runtime is no more than

$$\begin{aligned} E[T] &\leq \frac{8}{\Delta^2}\sum_{k=1}^{2n}\left(\lambda\ln\left(\frac{6\Delta\lambda}{4+\min\{z_{(j,i)}\}\Delta\lambda}\right)+\frac{1}{\min\{z_{(j,i)}\}}\right) \\ &\leq \frac{8}{\Delta^2}\left(2n\lambda\ln\left(\frac{6}{\min\{z_{(j,i)}\}}\right)+\frac{2n}{\min\{z_{(j,i)}\}}\right) \\ &= O(n\lambda\log(n)+n^2). \end{aligned}$$

□

Theorem 4.6.2. *For any constant $\chi_{\text{high}} > 0$, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, an arbitrary constant $q \in [0,1/2)$ and $p_c \in o(1)\cap\Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\frac{\chi_{\text{high}}}{n},\frac{\chi_{\text{low}}}{n}\}$ with self-adaptation parameter p_c and population size $\lambda > c\log(n)$ for a sufficiently large constant c on optimising LeadingOnes in the symmetric noise model (C,q), where $C \in \mathbb{R}$, is $O(n\lambda\log(n)+n^3)$.*

Theorem 4.6.2 is the most important result of this chapter. To prove it, we consider the two cases based on the noise level q. If the noise level is small enough compared to the high mutation rate $\frac{\chi_{\text{high}}}{n}$, we use a similar approach of Theorem 4.6.1 to complete the proof.

Otherwise, we use a different level partition and the new level-based theorem (Theorem 2.2.2) to prove it. Precisely, we define a value $\ell \in \mathbb{N}$ such that for any constant $\delta \in (0, (1/2-q)^3]$,

$$\left(1-\frac{\chi_{\text{high}}}{n}\right)^{\ell-1} > \frac{1+\delta}{2(1-q)} \geq \left(1-\frac{\chi_{\text{high}}}{n}\right)^{\ell}, \tag{4.1}$$

and distinguish between two cases: (A) $\ell \leq n-2$ and (B) $\ell \geq n-1$.

For case (A) $\ell \leq n-2$, we use the level partition defined in Definition 4.6.1. Figure 4.1 illustrates this level definition. The state space $\mathcal{Y}$ is divided into $n+1$ levels $A_{j\in[0..n]}$, with respect to $\text{LO}(x)$. Each of the first ℓ levels (the red/I region) is divided into two sub-levels, $A_{(j,1)}$ and $A_{(j,2)}$, representing the low and high mutation rates, respectively. Levels $A_{j\in[\ell+1..n-1]}$ (the green/III region) are defined as having only one sub-level $A_{(j,1)}$, which represents the low mutation rate. The final level (the optimal level, the white region), $A_n := A_{(n,1)}$, contains both high and low mutation rates. The sub-level $A_{(\ell,2)}$ (the cyan/II region) is extended to the rest of the state space , where $f(x) \geq \ell$ and with high mutation rate.

Definition 4.6.1. *For any $\ell \in [0, n-2]$, we define*

$$A_{(j,1)} := \begin{cases} \{(x, \frac{\chi_{\text{low}}}{n}) \mid LO(x) = j\} & \text{if } 0 \leq j \leq n-1, \\ \{(x, \frac{\chi_{\text{high}}}{n}), (x, \frac{\chi_{\text{low}}}{n}) \mid LO(x) = n\} & \text{if } j = n; \text{ and} \end{cases}$$

$$A_{(j,2)} := \begin{cases} \{(x, \frac{\chi_{\text{high}}}{n}) \mid LO(x) = j\} & \text{if } 0 \leq j \leq \ell-1 \\ \{(x, \frac{\chi_{\text{high}}}{n}) \mid LO(x) \geq \ell\} & \text{if } j = \ell. \end{cases}$$

To compare the levels $A_{(j,i)}$ and $A_{(j',i')}$, we define $(j,i) > (j',i')$ if either ($j = j'$ and $i > i'$) or ($j > j'$). To simplify the notation, we also define $(j,1)+1 = (j,2)$ and $(j,2)+1 = (j+1,1)$ for $j \leq \ell$, and $(j,1)+1 = (j+1,1)$ for $j \geq \ell+1$.

To apply Theorem 2.2.2, we must estimate the "upgrading" probability that is sampling an offspring in the higher level (condition (G1)), and the "growing" probability that sampling

an offspring in at least the same level (condition (G2)). The individuals in the green/III region might not have a sufficiently large probability of being selected if there are too many individuals with high fitness and high mutation rate (the cyan/II region). The high mutation rate can fail the optimisation under noise. Therefore, it is crucial to verify condition (G0), which ensures that there are not too many individuals in the cyan/II region. Finally, we gain an upper runtime bound by calculating the required population size (condition (G3)).

We first introduce some lemmas for the proof. The presence of noise essentially affects the selection (discussed in Chapter 3), so we compute the probability of selecting a high-level individual in noisy environments in Lemma 4.6.1. Lemmas 4.6.2-4.6.3 are used to verify conditions (G0) and (G2) of Theorem 2.2.2, respectively.

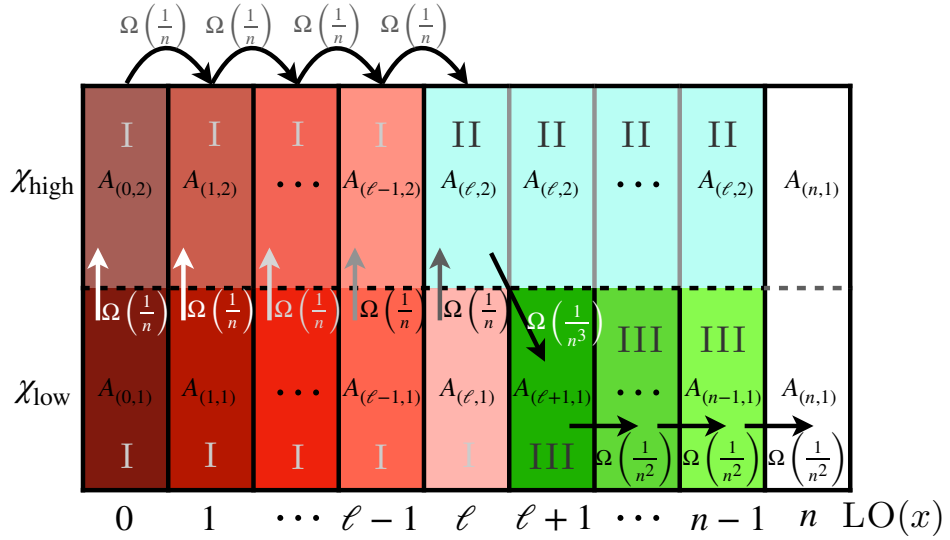

Figure 4.1: Illustration of the level partition defined in Definition 4.6.1. The notions on arrows indicate the "upgrading" probabilities for levels in the proof of Theorem 4.6.2.

Lemma 4.6.1. *Assume that we have a population $P_t \in \mathcal{Y}^\lambda$ where $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ and $\chi_{\text{high}} > \chi_{\text{low}} > 0$, that is sorted such that $P_t(1) \succeq \ldots \succeq P_t(\lambda)$ on the noise-free version of* LeadingOnes *function. For any $\gamma \in (0,1)$, any $C \in \mathbb{R}$ and any $q \in [0,1/2)$, if $(x_1, \chi_1/n)$ and $(x_2, \chi_2/n)$ are two individuals which are uniformly at random selected from P_t, and $(z, \chi'/n)$ is created by steps 5-8 in Algorithm 13 on the* LeadingOnes *function in the*

symmetric noise model (C, q)*, then the probability of* $(z, \chi'/n) \succeq P_t(\lfloor \gamma\lambda \rfloor)$ *where* $\lambda = |P_t|$ *is* $\gamma(2(1-q) - (1-2q)\gamma) \geq 2\gamma(1-q)(1-\gamma)$.

Proof. There are two events of selecting an "advanced individual", i.e., $(z, \chi'/n) \succeq P_t(\lfloor \gamma\lambda \rfloor)$:

(a) the algorithm selects two individuals $(x_1, \chi_1/n)$ and $(x_2, \chi_2/n)$ both from the top $\lfloor \gamma\lambda \rfloor$ individuals of sorted P_t. This happens with probability γ^2.

(b) either $(x_1, \chi_1/n)$ or $(x_2, \chi_2/n)$ from the top $\lfloor \gamma\lambda \rfloor$ individuals and selects such individual even if the noise occurs.

For event (2), we assume without loss of generality that $(x_1, \chi_1/n) \succeq (x_2, \chi_2/n)$. Then the probability of a successful comparison S, i.e, $(x_1, \chi_1/n)$ wins the tournament, is $\Pr(S) = \Pr\left(f^n(x_1) > f^n(x_2)\right) + \frac{1}{2}\Pr\left(f^n(x_1) = f^n(x_2)\right) = (1-q)$, where the last equation is from Lemma 3.2.1 (f). Therefore, we obtain $\Pr(S) = (1-q)^2 + (1-q)q = \frac{1}{2} + \frac{1}{2} - q$. Thus, the probability of selecting an individual in the top $\lfloor \gamma\lambda \rfloor$ individuals of sorted P_t is

$$\begin{aligned}\gamma^2 + 2\gamma(1-\gamma)\Pr(S) &= \gamma\left(2\left(1-q\right) - \left(1-2q\right)\gamma\right) \\ &\geq 2\gamma(1-q)(1-\gamma).\end{aligned}$$

□

The following lemma ensures that there are not too many individuals in the cyan/II region.

Lemma 4.6.2 (Condition (G0))**.** *Given any subset* $B \subset \mathcal{Y}$ *where* $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ *and* $\chi_{\text{high}} > \chi_{\text{low}} > 0$*, let* $Y_t := |P_t \cap B|$ *be the number of individuals in population* P_t *of the 2-tournament EA with SA-2mr from* $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ *and* $p_c \in o(1) \cap \Omega(1/n)$ *that belong to subset* B*. Consider the symmetric noise model* (C, q)*, where* $C \in \mathbb{R}$ *and constant* $q \in [0, 1/2)$*. If there exist three parameters* $\rho, \varepsilon, \sigma \in (0,1)$ *such that* $\Pr\left((y, \chi'/n) \in B \mid (z, \chi/n) \in B\right) \leq \rho$*, and* $\Pr\left((y, \chi'/n) \in B \mid (z, \chi/n) \notin B\right) \leq \sigma\psi - \varepsilon$ *for* $\psi \in [\psi_0, 1]$*, where* $\psi_0 = \frac{2(1-q)-(1-\sigma)/\rho}{1-2q}$ *and* $\psi_0 \in (0,1)$*, then*

$$\Pr\left((y, \chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right) \leq \psi(1-\varepsilon).$$

This lemma is very similar to Lemma 2 in (Dang and Lehre, 2016b).

Proof. Let $\psi := Y_t/\lambda$. For the upper bound, we assume that all search points in B have higher fitness and higher mutation rate than search points in $\mathcal{X}\backslash B$. Then,

$$\Pr\left((z,\chi/n) \in B \wedge (y,\chi'/n) \in B\right) = \Pr\left((z,\chi/n) \in B\right) \Pr\left((y,\chi'/n) \in B \mid (z,\chi/n) \in B\right)$$

by Lemma 4.6.1,

$$\leq \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi\right)\rho.$$

Let $g(\psi) = \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi\right)$ which is monotone increasing when $\psi \in (0,1)$ by Lemma A.2.11 (1), such that

$$\leq \left(\max(\psi_0,\psi)\right)\rho$$

by $\psi \geq \psi_0$ and the value of ψ_0,

$$\leq \psi\left(2\left(1-q\right) - \left(1-2q\right)\psi_0\right)\rho = \psi(1-\sigma).$$

Thus, the probability of producing an individuals in B is

$$\begin{aligned}\Pr\left((y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right) &= \Pr\left((z,\chi/n) \in B \wedge (y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right) \\ &\quad + \Pr\left((z,\chi/n) \notin B \wedge (y,\chi'/n) \in B \mid |P_t \cap B| \leq \psi\lambda\right) \\ &\leq \psi(1-\sigma) + (\sigma\psi - \varepsilon) \\ &\leq \psi - \varepsilon < \psi(1-\varepsilon) \leq \psi(1-\varepsilon).\end{aligned}$$

□

The following lemma gives the "expanding" probability of sampling an offspring in at least the same level.

Lemma 4.6.3 (Condition (G2))**.** *Assume that* $0 < \delta \leq (\frac{1}{2} - q)^3$ *and* $q \in [0, 1/2)$ *are any constants,* $\psi_0 := \frac{1-q}{1/2-q}\left(1 - \frac{1-(1/2)(1/2-q)^2}{1+\delta}\right)$ *and the state space* $\mathcal{Y} := \{0,1\}^n \times \{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ *is divided according to Definition 4.6.1. Consider* LeadingOnes *in the symmetric noise model* (C, q) *where* $C \in \mathbb{R}$*. There exist constants* $\Delta \in (0, (-q/5 + 1/10)/2]$ *and* $\gamma_0 \in (0, -q/90 + 1/180]$*, for any population* P_t *of the 2-tournament EA with SA-2mr from* $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ *and* $p_c \in o(1) \cap \Omega(1/n)$*, any* $\gamma \in (0, \gamma_0]$ *and* $(j, i) \geq (\ell, 2)$*, if* $|P_t \cap A_{(\ell,2)}| \leq \psi_0\lambda$*,* $|P_t \cap A_{\geq(j,i)}| \geq \gamma_0\lambda$ *and* $|P_t \cap A_{\geq(j,i)+1}| \geq \gamma\lambda$*, then* $\Pr\left((y, \chi'/n) \in A_{\geq(j,i)+1}\right) \geq \gamma(1 + \Delta)$.

Proof. We assume that $|P_t \cap A_{(\ell,2)}| \leq \psi_0\lambda$. By the definition of ψ_0, and $0 < \delta \leq (1/2 - q)^3$, we get upper and lower bounds of ψ_0 for later use. By replacing for the upper on δ in the definition of ψ_0,

$$\begin{aligned}\psi_0 &\leq \frac{2(1-q) - \frac{2\left(1-\frac{1}{2}\left(\frac{1}{2}-q\right)^2\right)(1-q)}{1+(1/2-q)^3}}{1-2q} \\ &= \frac{4 - 16q + 20q^2 - 8q^3}{9 - 6q + 12q^2 - 8q^3}. \end{aligned} \tag{4.2}$$

Then by replacing for the lower on δ in the definition of ψ_0,

$$\begin{aligned}\psi_0 &> \frac{2(1-q) - 2\left(1 - \frac{1}{2}\left(\frac{1}{2}-q\right)^2\right)(1-q)}{1-2q} \\ &= \left(\frac{1}{2} - q\right)\left(\frac{1}{2} - \frac{q}{2}\right). \end{aligned} \tag{4.3}$$

We now derive a lower bound p_0 on the probability that given an individual $(z, \chi/n)$ in level $A_{\geq(j,i)+1}$, the mutation operator produces an individual $(y, \chi'/n)$ in $A_{\geq(j,i)+1}$, for $(j, i) \geq (\ell, 2)$. The levels higher than $A_{(\ell,2)}$ are all with mutation rate $\frac{\chi_{\text{low}}}{n}$ except the optimal level $A_{(n,1)}$, thus

$$\begin{aligned}&\Pr\left((y, \chi'/n) \in A_{\geq(j,i)+1} \mid (z, \chi/n) \in A_{\geq(j,i)+1}\right) \\ &\geq \left(1 - \frac{\chi_{\text{low}}}{n}\right)^n (1 - p_c)\end{aligned}$$

by Lemma A.2.5,

$$\geq e^{-\chi_{\text{low}}}\left(1-\frac{{\chi_{\text{low}}}^2}{n}\right)(1-p_{\text{c}}) =: p_0 = (1-o(1)),$$

since $p_{\text{c}} \in o(1)$ and $\chi_{\text{low}} \in o(1)$.

Based on the level partition, the individuals in levels $A_{\geq(j,i)+1}$ are fitter than any other individual not in $A_{(\ell,2)}$, but could be less fit than individuals in $A_{(\ell,2)}$. Consequently, an individual in $A_{\geq(j,i)+1}$ can be generated under the condition that the following sequence of events occurs: event (a) no individual is selected from $A_{(\ell,2)}$, event (b) at least one individual is selected from $A_{\geq(j,i)+1}$ and it is exactly selected after comparison (Line 3 in Algorithm 5), and event (c) with probability at least p_0, the mutated individual z is in $A_{\geq j+1}$. The probability of event (a) occurring is given by $\Pr(S) = 1-q$, as demonstrated in Lemma 4.6.1. Thus, the joint probability of these events is at least

$$2\gamma(1-\psi_0-\gamma)(1-q)p_0 = 2\gamma(1-q)(1-\psi_0-\gamma)(1-o(1))$$

by Eq. (4.2),

$$\begin{aligned}
&\geq 2\gamma(1-q)\left(1-\frac{4-16q+20q^2-8q^3}{9-6q+12q^2-8q^3}-\gamma\right)(1-o(1)) \\
&= 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\gamma\right)(1-o(1))
\end{aligned}$$

by $\gamma_0 \in (0, -q/90 + 1/180]$, then for all $\gamma \in (0, \gamma_0)$,

$$\begin{aligned}
&\geq 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\gamma_0\right)(1-o(1)) \\
&= 2\gamma(1-q)\left(\frac{5+10q-8q^2}{9-6q+12q^2-8q^3}-\frac{1}{180}+\frac{q}{90}\right)(1-o(1)) \\
&= \gamma\left(1+\frac{81+1473q-4368q^2+2216q^3-48q^4+16q^5}{90(9-6q+12q^2-8q^3)}\right)(1-o(1))
\end{aligned}$$

by Lemma A.2.11 (2), we know $90(9-6q+12q^2-8q^3) < 810$, then

$$> \gamma\left(1+\frac{81+1473q-4368q^2+2216q^3-48q^4+16q^5}{810}\right)(1-o(1))$$
$$= \gamma\left(1+\left(\frac{1}{10}+\frac{491q}{270}-\frac{728q^2}{135}+\frac{1108q^3}{405}-\frac{8q^4}{135}+\frac{8q^5}{405}\right)\right)(1-o(1))$$

by $\Delta \in (0, (-q/5+1/10)/2]$ which is a constant and Lemma A.2.11 (3), we know $\frac{1}{10}+\frac{491q}{270}-\frac{728q^2}{135}+\frac{1108q^3}{405}-\frac{8q^4}{135}+\frac{8q^5}{405} \geq 2\Delta$, then

$$\geq \gamma(1+2\Delta)(1-o(1)) \geq \gamma(1+\Delta). \qquad \square$$

We now prove Theorem 4.6.2 using Lemmas 4.6.1 to 4.6.3.

Proof of Theorem 4.6.2. Recall Eq. 4.1, we first consider case (A) $\ell \leq n-2$ which refers to the level partition defined by Definition 4.6.1. We apply the new level-based theorem (Theorem 2.2.2) with respect to this level partitioning. Prior to proving the theorem, we introduce several constants that will be used in the subsequent calculations: δ is any constant that satisfies $0 < \delta \leq (1/2-q)^3$, $\rho := \frac{1+\delta}{2(1-q)}$, $\sigma := (1/2-q)^2/2$, $\psi_0 := \frac{2(1-q)-(1-\sigma)/\rho}{1-2q}$, $\zeta := \frac{1}{40}\left(\frac{1}{2}-q\right)^3$, $\varepsilon := (1/2-q)^3/10 > 0$, $\gamma_0 := \min\left\{\frac{\delta}{4(1+\delta)}, -q/90+1/180\right\}$ and $\Delta := \min\{(-q/5+1/10)/2, \delta/2\}$.

We now show that the condition (G0) of Theorem 2.2.2 is satisfied. We consider $A_{(\ell,2)}$ as the B subset in Theorem 2.2.2 for all $(j,i) \geq (\ell,2)$ (the cyan/II region illustrated in Figure 4.1). Assume $(z, \chi/n) \in A_{(\ell,2)}$, then, to produce $(y, \chi'/n) \in A_{(\ell,2)}$, it is necessary not to change the mutation rate and not flip the first ℓ bits. Using Eq. (4.1) and $p_c \in (0,1)$, the probability of this event is

$$\Pr\left((y,\chi'/n) \in A_{(\ell,2)} | (z,\chi/n) \in A_{(\ell,2)}\right) = (1-p_c)\left(1-\frac{\chi_{\text{high}}}{n}\right)^{\ell}$$
$$< \frac{1+\delta}{2(1-q)} = \rho.$$

When applying Lemma 4.6.2, we use the parameter ψ_0 which has been defined in terms of ρ and σ, as

$$\begin{aligned}\psi_0 &= \frac{2(1-q)-(1-\sigma)/\rho}{1-2q} \\ &= \frac{1-q}{1/2-q}\left(1-\frac{1-(1/2)\,(1/2-q)^2}{1+\delta}\right)\end{aligned}$$

by Eq. (4.3),

$$> \left(\frac{1}{2}-q\right)\left(\frac{1}{2}-\frac{q}{2}\right).$$

If $(z, \chi/n)$ is not in $A_{(\ell,2)}$, it is necessary to flip at least one specific bit-position, or change the mutation rate, an event which occurs with probability

$$\Pr\left((y,\chi'/n) \in A_{(\ell,2)} \mid (z,\chi/n) \notin A_{(\ell,2)}\right) < \max\left\{p_{\mathrm{c}}, \frac{\chi_{\mathrm{high}}}{n}\right\}$$

which is by constants $p_{\mathrm{c}}, \frac{\chi_{\mathrm{high}}}{n} \in o(1)$, $\psi_0, \sigma > 0$ and $\varepsilon = (1/2-q)^3/10 > 0$, then

$$< \psi_0\sigma - \varepsilon \leq \psi\sigma - \varepsilon.$$

$$\begin{aligned}\psi_0\sigma - \varepsilon &> \left(\frac{1}{2}-q\right)\left(\frac{1}{2}-\frac{q}{2}\right)\frac{(1/2-q)^2}{2} - \frac{(1/2-q)^3}{10} \\ &= \frac{1}{20}\left(\frac{1}{2}-q\right)^3(3-5q)\end{aligned}$$

since $q < 1/2$, we have $3-5q > 1/2$, then

$$> \frac{1}{40}\left(\frac{1}{2}-q\right)^3 =: \zeta > 0,$$

where ζ is a constant which is larger than $\max\left\{p_{\mathrm{c}}, \frac{\chi_{\mathrm{high}}}{n}\right\} = o(1)$. Lemma 4.6.2 now implies that condition (G0) satisfied with $\psi_0 = \frac{1-q}{1/2-q}\left(1-\frac{1-(1/2)(1/2-q)^2}{1+\delta}\right)$.

We then verify condition (G2) of Theorem 2.2.2. We first consider the case of $(j,i) \leq (\ell,1)$ (the red/I region illustrated in Figure 4.1). Recall the definition of γ_0 and define

$A_+ := A_{\geq(j,i)+1}$ Assume that $|P_t \cap A_+| = \gamma\lambda$ for $\gamma \in (0, \gamma_0]$. To produce an individual $(y, \chi'/n) \in A_+$, it suffices to select an individual $(z, \chi/n) \in A_+$, do not change the mutation rate and do not flip the first $j+1$ bits, then the lower bound of this probability is,

$$\begin{aligned}\Pr\left((y, \frac{\chi'}{n}) \in A_+\right) &= \Pr\left((z, \frac{\chi}{n}) \in A_+\right) \Pr\left((y, \frac{\chi'}{n}) \in A_+ | (z, \frac{\chi}{n}) \in A_+\right) \\ &= 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1 - \frac{\chi'}{n}\right)^j\end{aligned}$$

since $j \leq \ell$ in this case,

$$\begin{aligned}&\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1-\chi'/n\right)^\ell \\ &\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\left(1-\chi_{\text{high}}/n\right)^\ell\end{aligned}$$

by the definition of ℓ in Eq. (4.1),

$$\begin{aligned}&\geq 2\gamma(1-q)(1-\gamma)(1-p_c)\frac{1+\delta}{2(1-q)}\left(1 - \frac{\chi_{\text{high}}}{n}\right) \\ &= \gamma(1-\gamma)(1-p_c)(1+\delta)(1-o(1)) \\ &\geq \gamma(1-\gamma_0)(1-p_c)(1+\delta)(1-o(1)) \\ &\geq \gamma\left(1+3\delta/4\right)(1-p_c)(1-o(1))\end{aligned}$$

then since $\Delta = \delta/2$ and $p_c \in o(1)$,

$$\geq \gamma(1+\Delta).$$

We also know that condition (G2) is satisfied if $(j,i) \geq (\ell, 2)$ from Lemma 4.6.3 for constants γ_0 and Δ.

We now verify condition (G1) of Theorem 2.2.2. Assume that the size of $P_t \cap A_{\geq(j,i)}$ is at least $\gamma_0\lambda$, i.e., $\gamma_0\lambda \leq |P_t \cap A_{\geq(j,i)}|$. The lower bound of selecting an individual $(z, \chi/n) \in A_{\geq(j,i)}$ is $\Pr\left((z, \chi/n) \in A_{\geq(j,i)}\right) \geq \gamma_0^2 = \Omega(1)$. We distinguish levels into four groups:

- For levels $A_{(j,1)\leq(\ell,1)}$ (the red/I region with χ_{low} in Figure 4.1), to produce an individual $(y, \chi'/n) \in A_{\geq(j,1)+1}$ it suffices to select an individual $(z, \chi/n) \in A_{\geq(j,1)}$ and change its mutation rate from $\frac{\chi_{\text{low}}}{n}$ to $\frac{\chi_{\text{high}}}{n}$, then this probability is at least

$$\begin{aligned}\Pr\left((y, \chi'/n) \in A_{\geq(j,1)+1}\right) &\geq \Pr\left((z, \chi/n) \in A_{\geq(j,1)}\right) p_{\text{c}}(1-q) \\ &=: z_{(j,1)} \in \Omega\left(\frac{1}{n}\right).\end{aligned}$$

- For levels $A_{(j,2)\leq(\ell-1,2)}$ (the red/I region with $\frac{\chi_{\text{high}}}{n}$ in Figure 4.1), to produce an individual $(y, \chi'/n) \in A_{\geq(j,2)+1}$ it suffices to select an individual $(z, \chi/n) \in A_{\geq(j,2)}$, do not change the mutation rate and only flip the $(j+1)$-th bit, then the lower bound of this probability is

$$\begin{aligned}\Pr\left((y, \chi'/n) \in A_{\geq(j,2)+1}\right) &\geq \Pr\left((z, \chi/n) \in A_{\geq(j,2)}\right)(1-p_{\text{c}})(1-q)\left(1-\frac{\chi_{\text{high}}}{n}\right)^{n-1}\frac{\chi_{\text{high}}}{n} \\ &=: z_{(j,2)} \in \Omega\left(\frac{1}{n}\right).\end{aligned}$$

- For level $A_{(\ell,2)}$ (the cyan/II region in Figure 4.1), to produce an individual $(y, \chi'/n) \in A_{\geq(\ell+1,1)}$ it suffices to select an individual $(z, \chi/n) \in A_{\geq(\ell,2)}$, change its mutation rate from $\frac{\chi_{\text{high}}}{n}$ to $\frac{\chi_{\text{low}}}{n}$ and only flip the $\ell+1$-th bit, then the lower bound of this probability is

$$\begin{aligned}\Pr\left((y, \chi'/n) \in A_{\geq(\ell+1,1)}\right) &\geq \Pr\left((z, \chi/n) \in A_{\geq(\ell,2)}\right)(1-q)p_{\text{c}}\left(1-\frac{\chi_{\text{low}}}{n}\right)^{n-1}\frac{\chi_{\text{low}}}{n} \\ &=: z_{(\ell,2)} \in \Omega\left(\frac{1}{n^3}\right).\end{aligned}$$

- For levels $A_{(j,1)\geq(\ell+1,1)}$ (the green/III region in Figure 4.1), to produce an individual $(y, \chi'/n) \in A_{\geq(j+1,1)}$ it suffices to select an individual $(z, \chi/n) \in A_{\geq(j,1)}$, do not change the mutation rate and only flip the $(j+1)$-th bit, then the lower bound of this probability is

$$\begin{aligned}\Pr\left((y, \chi'/n) \in A_{\geq(j+1,1)}\right) &\geq \Pr\left((z, \chi/n) \in A_{\geq(j,1)}\right)(1-q)(1-p_{\text{c}})\left(1-\frac{\chi_{\text{low}}}{n}\right)^{n-1}\frac{\chi_{\text{low}}}{n} \\ &=: z_{(j,1)} \in \Omega\left(\frac{1}{n^2}\right).\end{aligned}$$

Then we compute the population size required by condition (G3). Since $\gamma_0, \Delta > 0$ are some constants and $m = 2\ell + (n-\ell) \leq 2n$, then $\lambda > \frac{12}{\gamma_0\Delta^2}\ln\left(\frac{300m}{\min\{z_{(j,i)}\}\Delta^2}\right) = O(\log(n))$. Condition (G3) is satisfied by $\lambda \geq c\log(n)$ for a sufficiently large constant c.

Finally, all conditions of Theorem 2.2.2 hold, and the expected runtime is no more than

$$\begin{aligned}
E[T] \leq{}& \frac{12\lambda}{\Delta} + \frac{96}{\Delta^2}\sum_{j=0}^{\ell}\left(\lambda \ln\left(\frac{6\Delta\lambda}{4+z_{(j,1)}\Delta\lambda}\right)+\frac{1}{z_{(j,1)}}\right)\\
&+\frac{96}{\Delta^2}\sum_{j=0}^{\ell-1}\left(\lambda \ln\left(\frac{6\Delta\lambda}{4+z_{(j,2)}\Delta\lambda}\right)+\frac{1}{z_{(j,2)}}\right)\\
&+\frac{96}{\Delta^2}\left(\lambda \ln\left(\frac{6\Delta\lambda}{4+z_{(\ell,2)}\Delta\lambda}\right)+\frac{1}{z_{(\ell,2)}}\right)\\
&+\frac{96}{\Delta^2}\sum_{j=\ell+1}^{n-1}\left(\lambda \ln\left(\frac{6\Delta\lambda}{4+z_{(j,1)}\Delta\lambda}\right)+\frac{1}{z_{(j,1)}}\right)\\
={}& O\Bigg(\lambda+(\ell+1)\left(\lambda\log(n)+n\right)+\ell\left(\lambda\log(n)+n\right)\\
&+\left(\lambda\log(n)+n^3\right)+(n-\ell-2)\left(\lambda\log(n)+n^2\right)\Bigg)\\
={}& O(n\lambda\log(n)+n^3).
\end{aligned}$$

For case (B) $\ell \geq n-1$, we know that $\frac{1+\delta}{2(1-q)} \leq \left(1-\frac{\chi_{\text{high}}}{n}\right)^{n-1} = \left(1-\frac{\chi_{\text{high}}}{n}\right)^{n}/(1-o(1))$. We use the level-based theorem (Theorem 2.2.1) on the level partition applied in the proof of Theorem 4.6.1. Condition (G2) can be verified by definitions of Δ and γ_0:

$$\begin{aligned}
\Pr\left((y,\chi'/n)\in A_{\geq(j,i)+1}\right) &= \Pr\left((z,\chi/n)\in A_{\geq(j,i)+1}\right)\\
&\quad\cdot\Pr\left((y,\chi'/n)\in A_{\geq(j,i)+1}\mid(z,\chi/n)\in A_{\geq(j,i)+1}\right)\\
&\geq (\gamma^2+2\gamma(1-\gamma))(1-q)(1-p_{\text{c}})\left(1-\frac{\chi'}{n}\right)^n\\
&\geq 2\gamma(1-\gamma)(1-q)(1-p_{\text{c}})\left(1-\frac{\chi_{\text{high}}}{n}\right)^n\\
&\geq \gamma(1+\Delta).
\end{aligned}$$

Condition (G1) is similar with the proof of Theorem 4.6.1. Then, we know that the runtime is $O(n\lambda\log(n)+n^2)$ if using population size $\lambda > c\log(n)$ for a sufficiently large constant c.

Therefore, the overall runtime is $O(n\lambda\log(n)+n^3)$. □

4.7 Experiments

As a complement to our theoretical analysis, we expand our investigation to include both the symmetric, one-bit and bit-wise noise models, as well as self-adaptation of mutation rates within a given interval. In this section, we empirically analyse the performance of 2-tournament EAs using fixed and self-adaptive mutation rates on LeadingOnes and OneMax under different levels of noise. Additionally, we investigate the behaviour of mutation rates in self-adaptation under noise.

We use the parameter settings satisfying the runtimes analyses in Sections 4.4-4.6. For fixed mutation rates, we use $\frac{\chi_{\text{high}}}{n} = 1/(2n)$ and $\frac{\chi_{\text{low}}}{n} = 5/n^2$ which are less than the *error threshold* $\ln(2)/n$ for the 2-tournament EA (Lehre, 2010). For the SA-2mr, we self-adapt mutation rates from $\{\frac{\chi_{\text{high}}}{n}, \frac{\chi_{\text{low}}}{n}\}$ with a self-adaptation parameter $p_{\text{c}} = 1/(10n)$. For the SA, we set self-adaptive parameters $A = 1.2$ and $p_{\text{inc}} = 0.4$ as also utilised in Chapter 7. All algorithms use the same population size of $\lambda = 200\ln(n)$, and a uniformly sampled initial population. For symmetric noise, we study algorithms on LeadingOnes and OneMax with noise levels $q \in \{0.2, 0.3, 0.4\}$ and $q \in \{0.2, 0.3, 0.4\}$, respectively. For one-bit noise, we study algorithms on LeadingOnes and OneMax with noise levels $q \in \{0.4, 0.6, 0.8\}$ and $q \in \{0.85, 0.90, 0.95\}$, respectively. For bit-wise noise, we examine algorithms applied to LeadingOnes and OneMax with noise levels $p \in \{0.8/n, 1.0/n, 1.2/n\}$ and $p \in \{5\ln(n)/n, 6\ln(n)/n, 7\ln(n)/n\}$, respectively, which are set with respect to the problem size n. For each setting, we independently run each algorithm 100 times for LeadingOnes and OneMax with problem size $n = 100$ to 200 with step size 10 and $n = 100$ to 500 with step size 40, respectively, and record runtimes. To monitor the behaviour of self-adaptive algorithms, we record mutation parameters χ of individuals during each run. Additionally, we independently perform each self-adaptive algorithm 30 times on each setting. As a comparison, we also run the same experiments without noise. Outcomes for both the one-bit noise and bit-wise noise

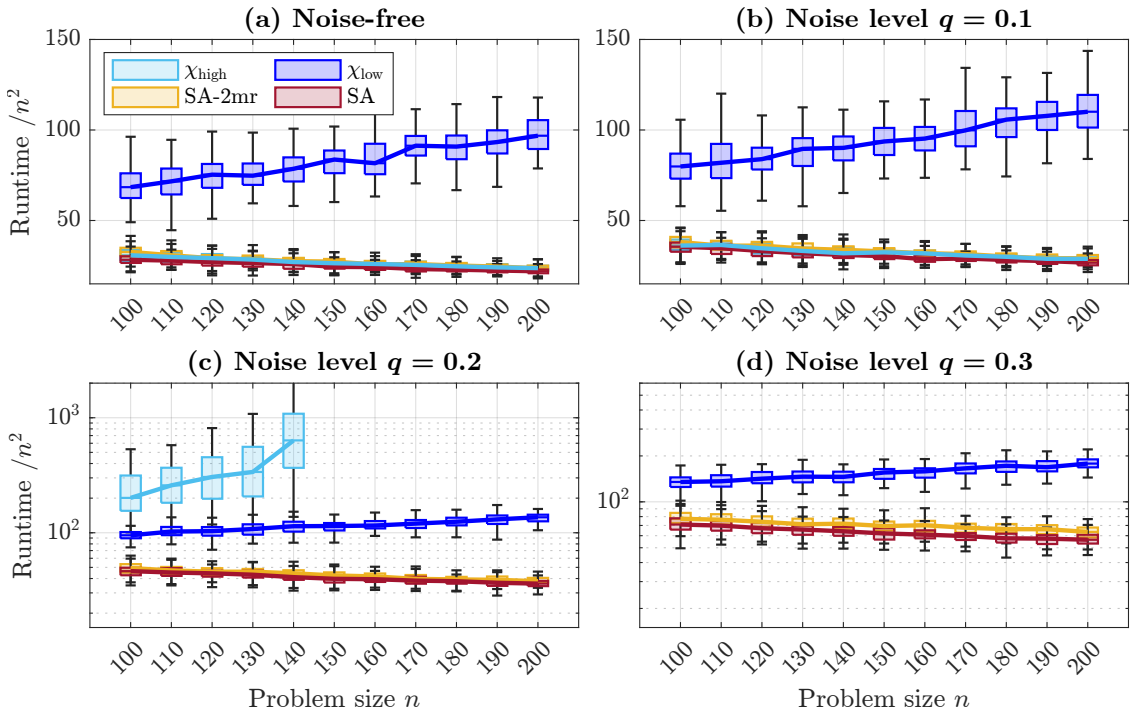

Figure 4.2: Runtimes of 2-tournament EAs on LeadingOnes under symmetric noise with different noise levels ($C = 0$).

models are presented in Sections 4.7.2 and 4.7.3, respectively. Comprehensive statistical results of the experiments can be found in Appendix A.4, including medians and hypothesis test results.

4.7.1 Symmetric Noise

Figures 4.2-4.3 illustrate the runtimes on LeadingOnes and OneMax under symmetric noise, respectively. The corresponding statistical results are displayed in Tables A.1-A.4 and Tables A.5-A.8, respectively. Note that the y-axes in Figures 4.2 (c)-(d) and Figures 4.7 (a)-(d) are log-scaled, and all runtimes are divided by n^2 for LeadingOnes and $n \ln(n)$ for OneMax, respectively. These divisions correspond to the well-known runtime results for the LeadingOnes and OneMax function in noise-free scenarios. Note that the runtime of

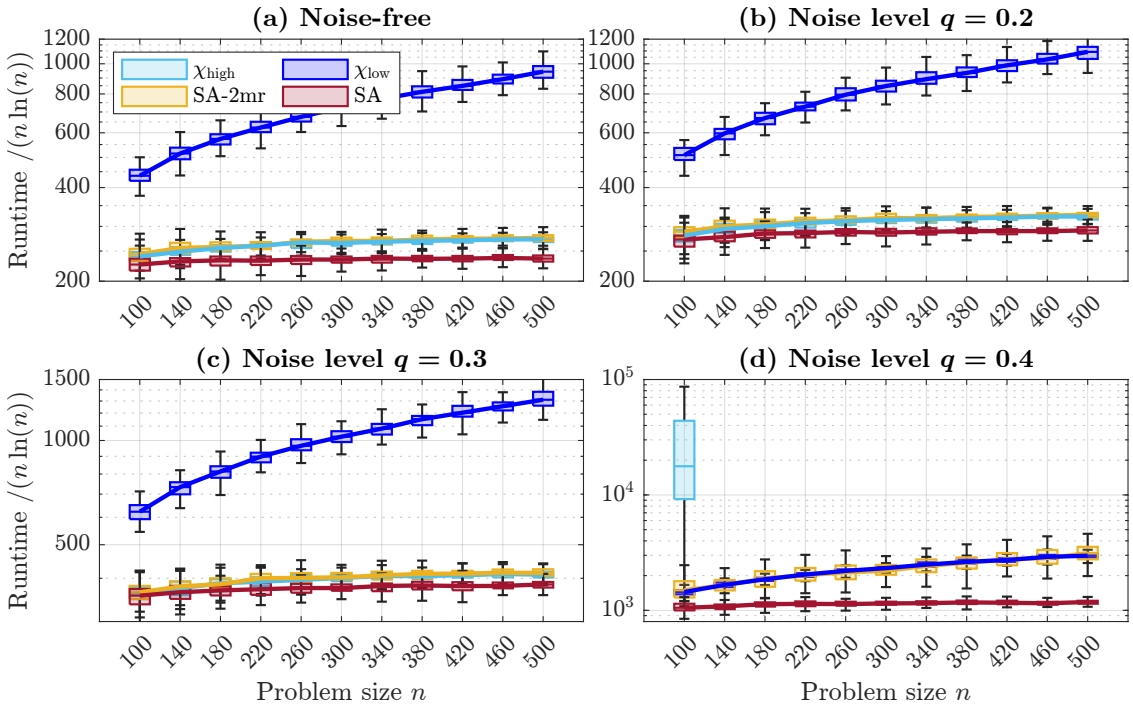

Figure 4.3: Runtimes of 2-tournament EAs on OneMax under symmetric noise with different noise levels ($C = 0$).

the 2-tournament EA using the high mutation rate exceeds the evaluation budget of 5×10^7 for optimising LeadingOnes for $n \geq 150$ under symmetric noise with noise level $q = 0.2$, and for $n \geq 100$ with noise level $q = 0.3$. Similarly, on OneMax, the runtime of the 2-tournament EA using the high mutation rate exceeds the evaluation budget of 2×10^8 for $n \geq 110$ under symmetric noise with noise level $q = 0.4$.

From Theorems 3.3.8 and 3.3.6, we can conclude that the runtime of the 2-tournament EA using the mutation rate $\chi/n = 0.5/n$ on LeadingOnes under symmetric noise is polynomial when the noise level $q < 0.1756$, and exponential when $q > 0.1757$. Figure 4.2 supports these theoretical results, indicating that using a high mutation rate of $\chi/n = 0.5/n$ may fail to optimise LeadingOnes under high-level noise $q \geq 0.2$. On the other hand, employing a low mutation rate is slower than using a high mutation rate under low-level symmetric noise ($q \leq 0.1$) when optimising LeadingOnes. However, the 2-tournament EA using

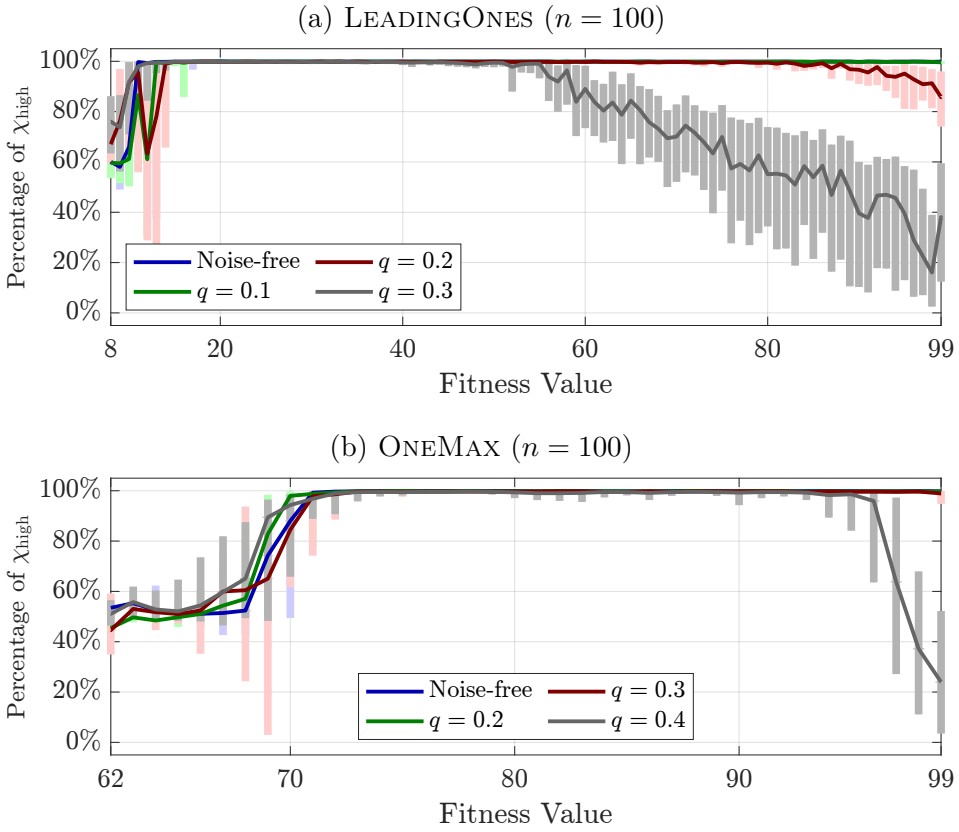

Figure 4.4: The percentage of $\frac{\chi_{high}}{n}$ individuals and the highest real fitness value per generation for 2-tour' EA with SA-2mr under symmetric noise with different noise levels ($C = 0$, 30 runs).

SA-2mr achieves comparable performance to the high mutation rate when the noise level is $q \leq 0.1$. Furthermore, it outperforms the low mutation rate under high-level symmetric noise, specifically when $q \geq 0.2$. Most notably, the 2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

From Figures 4.3 (a), (b), (c), it is evident that the 2-tournament EA employing a low mutation rate is slower than using a high mutation rate under low-level symmetric noise (i.e., $q \leq 0.3$) when optimising OneMax. However, as shown in Figure 4.3 (d), the high mutation rate may fail to optimise LeadingOnes under high-level noise, whereas employing a low mutation rate can be more efficient. Similarly, the 2-tournament EA using SA-2mr

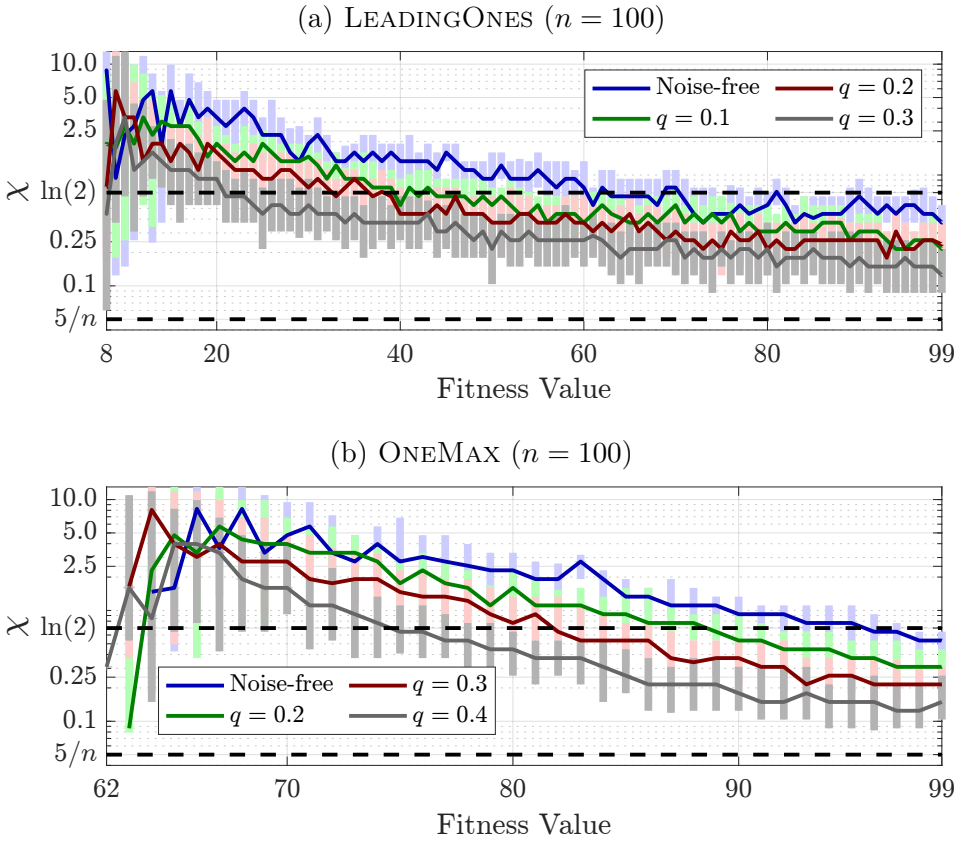

Figure 4.5: Real fitness and mutation parameter of the highest real fitness individual per generation of 2-tour' EA with SA under symmetric noise with different noise levels ($C = 0$, 30 runs).

achieves performance comparable to the high mutation rate when the noise level is $q \leq 0.3$. Furthermore, it outperforms the low mutation rate under high-level symmetric noise, specifically when $q = 0.4$. Most notably, the 2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

Figures 4.4-4.5 present the relationships between mutation rates and real fitness values under different levels of one-bit noise in SA-2mr and SA, respectively. The lines indicate the median of values of 30 runs. The corresponding shadows indicate the IQRs. We observe a decrease in the mutation rate when the noise level increases on LeadingOnes and OneMax

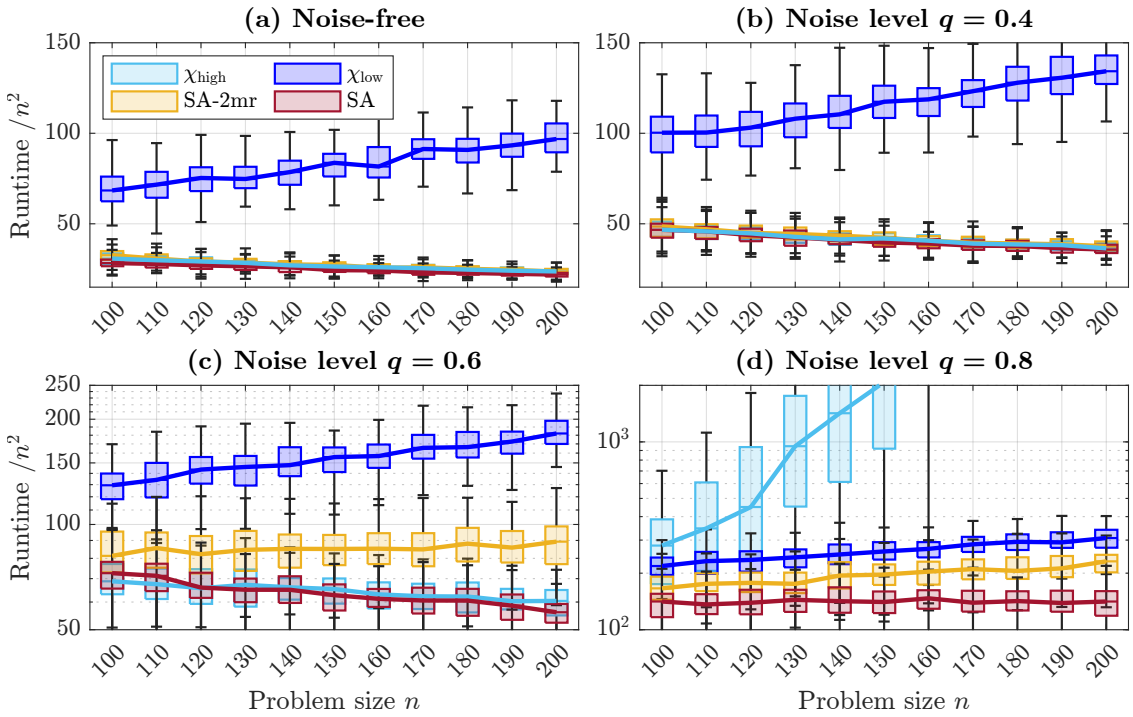

Figure 4.6: Runtimes of 2-tournament EAs on LeadingOnes under one-bit noise with different noise levels.

in both self-adaptations. Particularly, using SA not only reduces the mutation rate below the error threshold ($\chi/n < \ln(2)/n$) (Lehre, 2010), but also furthermore reduces it with respect to the noise level on LeadingOnes and OneMax.

4.7.2 One-bit Noise

Figures 4.6-4.7 illustrate runtimes on LeadingOnes and OneMax under one-bit noise, respectively. The corresponding statistical results are displayed in Tables A.9-A.11 and Tables A.12-A.14, respectively. Note that the y-axes in Figures 4.6 (c)-(d) and Figures 4.7 (a)-(d) are log-scaled, and all runtimes are divided by n^2 for LeadingOnes and $n \ln(n)$ for OneMax, respectively.

From Figures 4.6 (a), (b), (c), it is evident that the 2-tournament EA employing a low

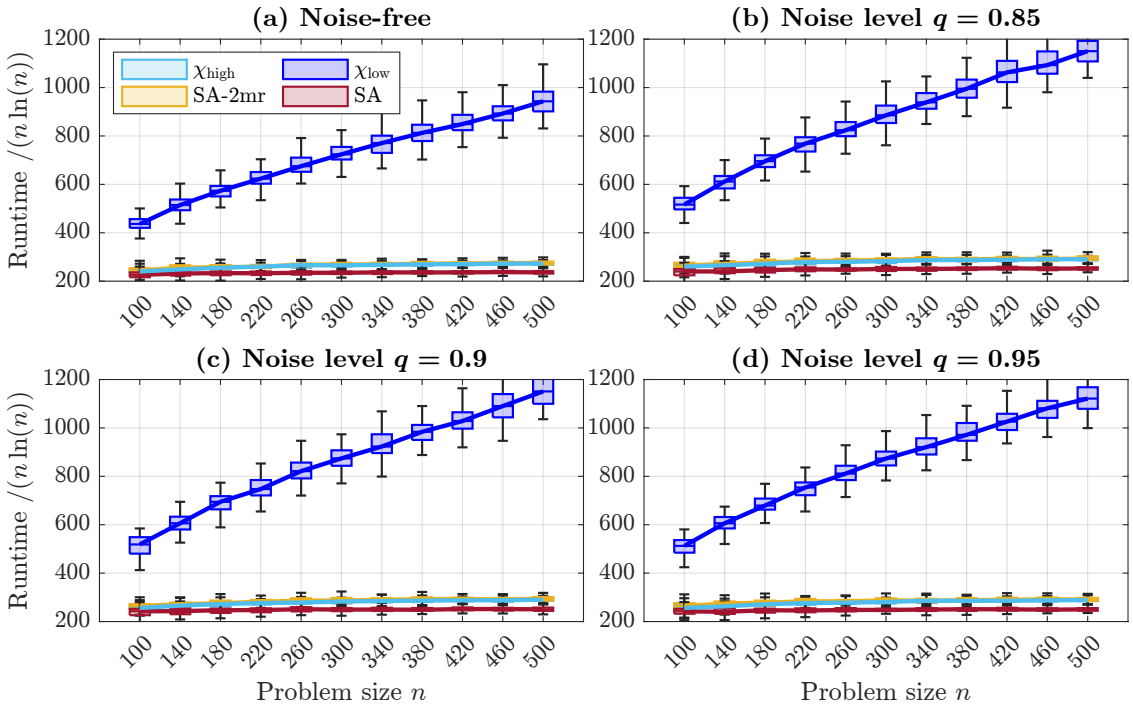

Figure 4.7: Runtimes of 2-tournament EAs on OneMax under one-bit noise with different noise levels.

mutation rate is slower than using a high mutation rate under low-level one-bit noise (i.e., $q \leq 0.6$) when optimising LeadingOnes. Conversely, utilising a high mutation rate results in faster optimisation. However, as shown in Figure 4.6 (d), the high mutation rate may fail to optimise LeadingOnes under high-level noise. Note that the runtime of the 2-tournament EA utilising a high mutation rate exceeds the evaluation budget of 2×10^8 when optimising LeadingOnes for $n \geq 170$ under one-bit noise with noise level $q = 0.8$. Specifically, the runtimes of using a high mutation rate increase sharply as the problem size grows under high-level one-bit noise ($q = 0.8$), whereas employing a low mutation rate can be more efficient. This observation is consistent with the theoretical study presented in Section 4.4. On the other hand, the 2-tournament EA using SA-2mr achieves performance comparable to the high mutation rate when the noise level is $q \leq 0.4$ and is only slightly slower than the high mutation rate when the noise level is $q = 0.6$. Furthermore, it outperforms the low mutation rate under high-level one-bit noise, specifically when $q = 0.8$. Most notably, the

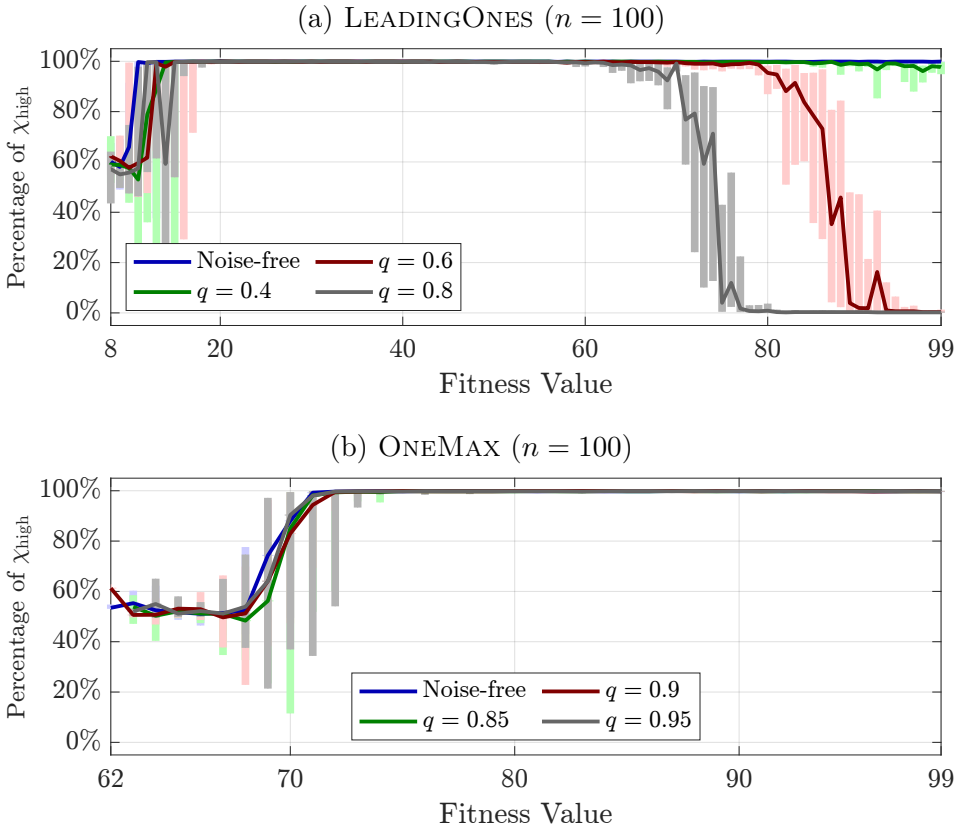

Figure 4.8: The percentage of $\frac{\chi_{\text{high}}}{n}$ individuals and the highest real fitness value per generation for 2-tour' EA with SA-2mr under one-bit noise with different noise levels (30 runs).

2-tournament EA using SA outperforms all other algorithms across all tested noise levels.

In Figure 4.7, which presents results on the OneMax problem, both the high mutation rate EA and self-adaptive EAs outperform the low mutation rate EA across all noise levels. The efficiency of the high mutation rate under high-level noise on OneMax can be explained by Theorem 3.3.1, which states that the 2-tournament EA with a constant mutation parameter χ can achieve the optimum in expected time $O(n \log(n))$ on OneMax under any level of one-bit noise. Despite this, the 2-tournament EA using SA-2mr is only marginally slower than the high mutation rate for all noise levels, and using SA results in faster performance compared to all others.

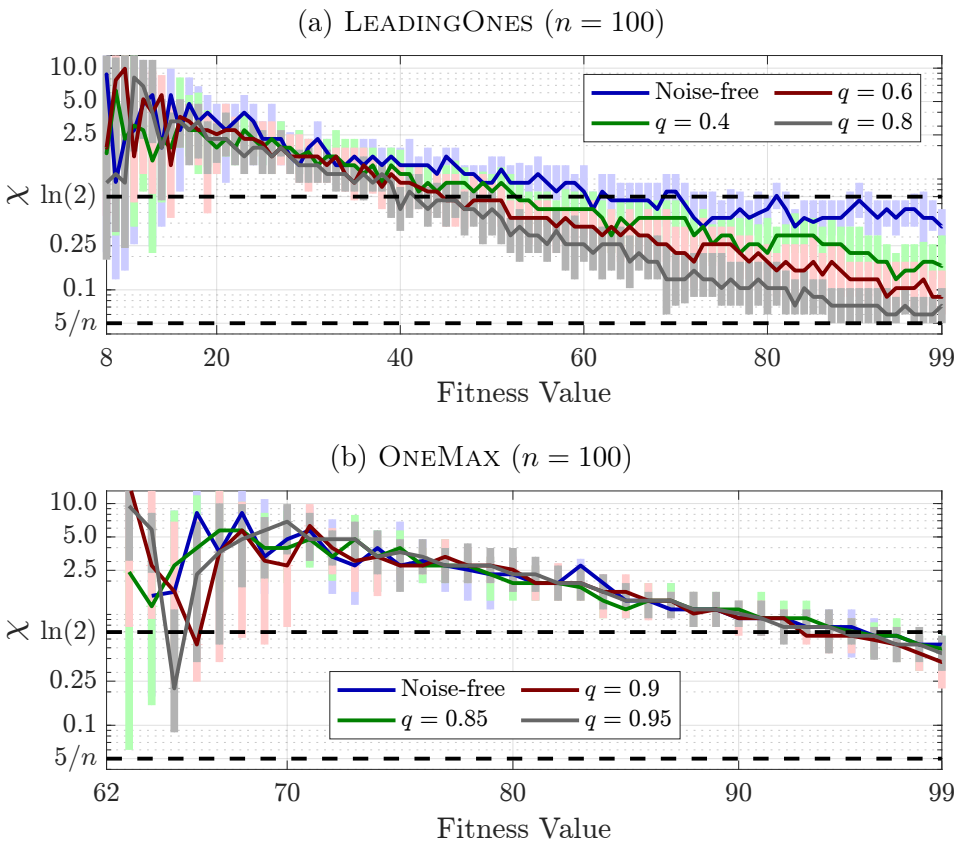

Figure 4.9: Real fitness and mutation parameter of the highest real fitness individual per generation of 2-tour' EA with SA under one-bit noise with different noise levels (30 runs).

Similar with Section 4.7.1, Figures 4.8-4.9 show a decrease in the mutation rate when the noise level increases on LeadingOnes in both self-adaptations.

4.7.3 Bit-wise Noise

Figures 4.10-4.11 illustrate runtimes on LeadingOnes and OneMax under bit-wise noise, respectively. The corresponding statistical results are displayed in Tables A.16-A.17 and Tables A.18-A.20, respectively. Note that the y-axes in Figures 4.11 (c)-(d) are log-scaled. In Figure 4.10, we observe that the 2-tournament EA employing SA-2mr is faster than the low mutation rate and slower than the high mutation rate for noise levels $p = 0.8/n$

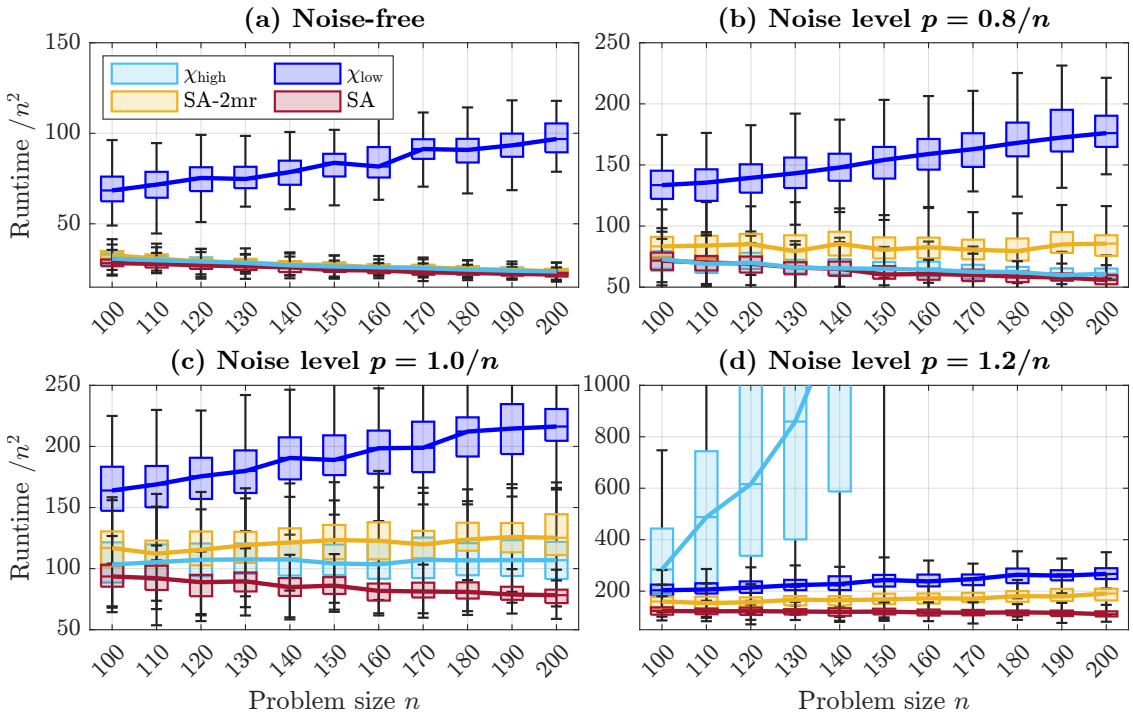

Figure 4.10: Runtimes of 2-tournament EAs on LeadingOnes under bit-wise noise with different noise levels.

and $1.0/n$, but the gap is not substantial. Consistent with previous observations in the one-bit noise model, under high-level noise $p = 1.2/n$, using SA-2mr outperforms all static algorithms. Note that the runtime of the 2-tournament EA utilising a high mutation rate exceeds the evaluation budget of 2×10^8 when optimising LeadingOnes and OneMax for $n \geq 160$ and $n \geq 100$ under bit-wise noise with noise levels $p = 1.2/n$ and $7\ln(n)/n$, respectively. Furthermore, the 2-tournament EA using SA consistently achieves the best performance regardless of the noise level. In Figure 4.11, the 2-tournament EA employing SA-2mr demonstrates better performance than the low mutation rate. Additionally, it exhibits comparable performance to the high mutation rate when the noise level is $p = 5\ln(n)/n$ and faster performance for other noise levels, namely $p = 6\ln(n)/n$ and $7\ln(n)/n$. As always, the 2-tournament EA using SA demonstrates consistently the best performance in this setting. Similar to the results of symmetric and one-bit noise, Figures 4.12-4.13 show

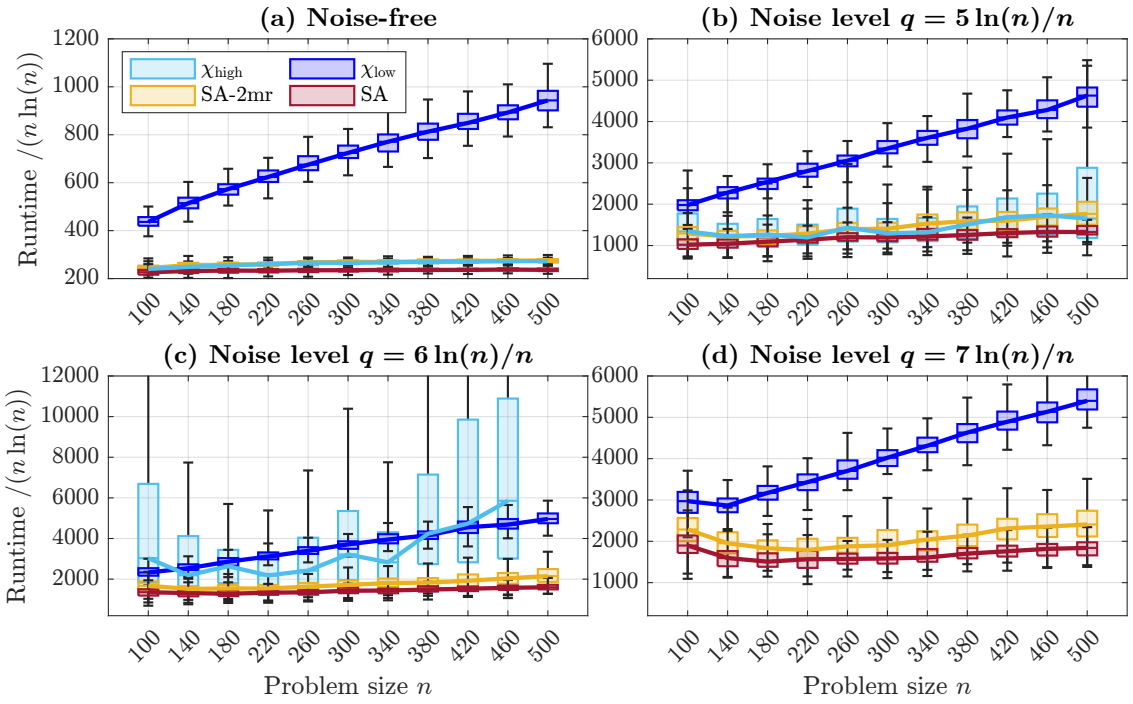

Figure 4.11: Runtimes of 2-tournament EAs on OneMax under bit-wise noise with different noise levels.

that self-adaptive EAs self-adapt the mutation rate to the noise level.

4.8 Conclusion

In this chapter, we conducted runtime analysis and empirical analysis on the 2-tournament EAs with self-adaptive mutation rates in a noisy environment. Although the noise model examined in the theoretical study is relatively simplistic and artificial, our findings still provide a compelling indication that the self-adaptive EA remarkably adapts to the presence of noise. The empirical results further affirm that self-adaptation can adjust mutation rates according to noise, thereby leading to more efficient optimisation than other algorithms.

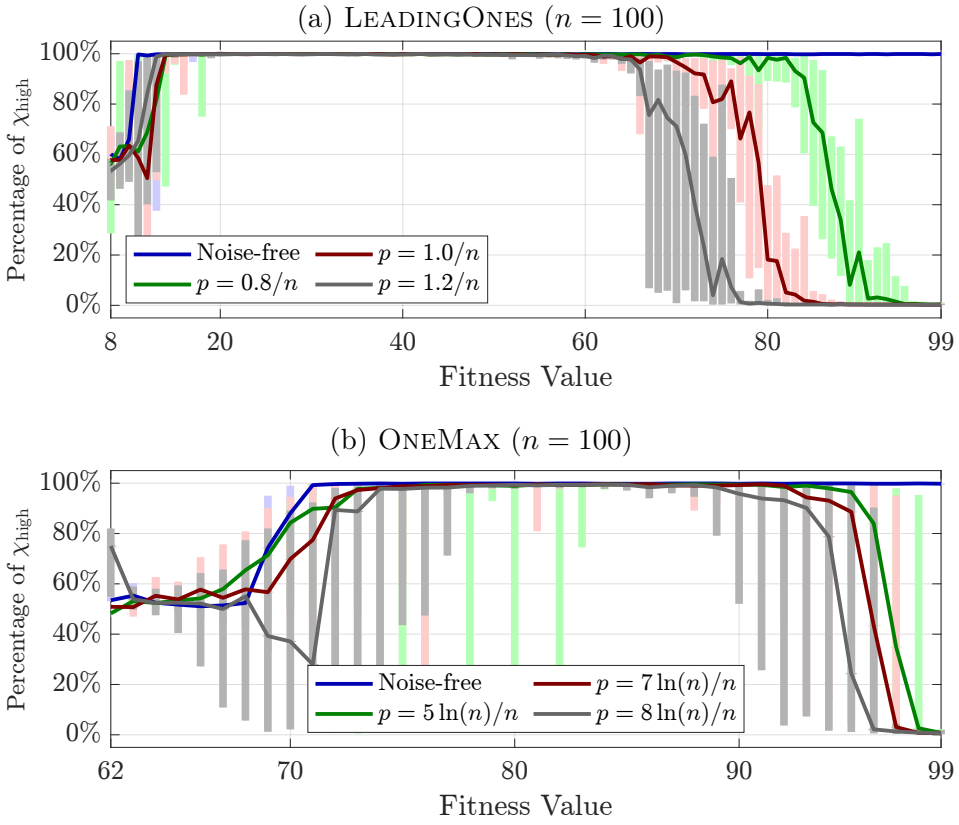

Figure 4.12: The percentage of $\frac{\chi_{\text{high}}}{n}$ individuals and the highest real fitness value per generation for 2-tour' EA with SA-2mr under bit-wise noise with different levels (30 runs).

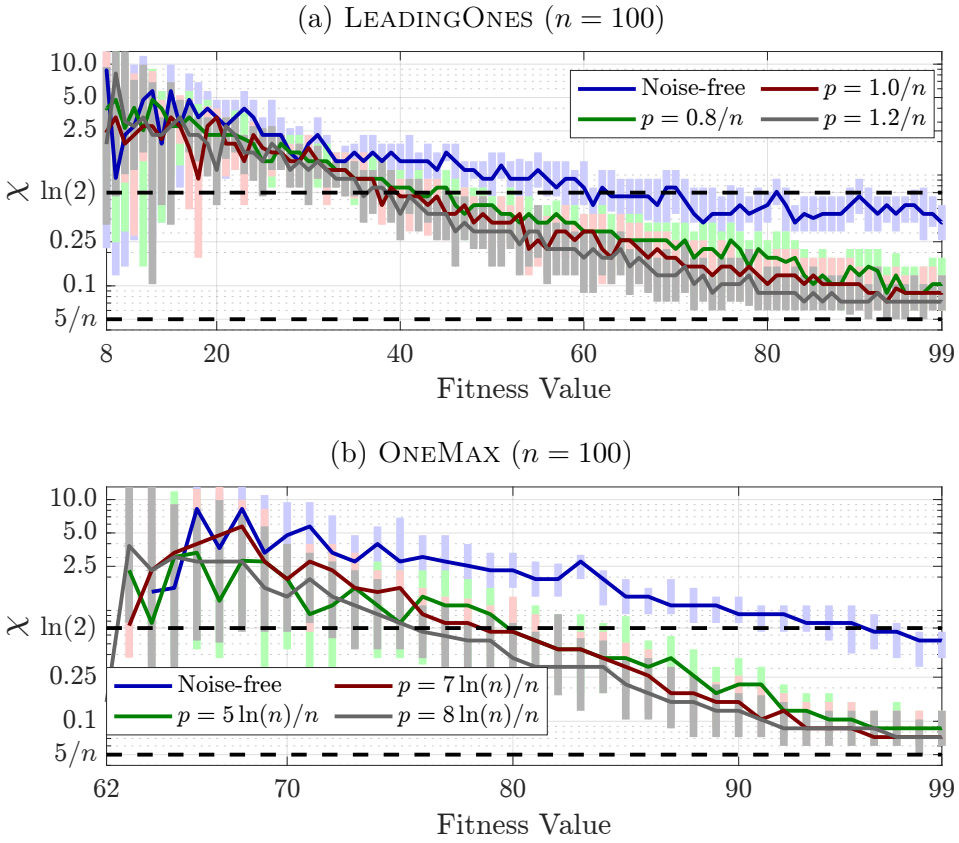

Figure 4.13: Real fitness and mutation parameter of the highest real fitness individual per generation of 2-tour' EA with SA under bit-wise noise with different noise levels (30 runs).

Chapter Five

Self-adaptation on Dynamic Optimisation

Authors: Per Kristian Lehre and Xiaoyu Qin

This chapter is based on the following publication:

Self-adaptation Can Help Evolutionary Algorithms Track Dynamic Optima (Lehre and Qin, 2023a) which is published in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'23).

5.1 Introduction

EAs can solve a wide variety of dynamic optimisation problems, where the objective function changes over time (Jin and Branke, 2005). In this context, algorithms need to adapt and update solutions quickly since previously best solutions might no longer be good. As introduced in Section 2.3.3.2, many rigorous analyses of EAs and other randomised search heuristics in dynamic environments have been published in the previous two decades. Changes in the objective function could lead to modifications in the appropriate parameter settings. Therefore, dynamic parameter settings might be necessary for dynamic optimisation problems. Self-adaptation could contribute to the control of parameters during optimisation. As discussed in Section 2.3.2, an empirical study illustrated that the self-adaptive parameter control mechanism could effectively respond to changes in the fitness function, transitioning from OneMax to ZeroMax (Smith, 2001). However, the benefit of self-adaptation on dynamic optimisation problems remains unknown.

In this chapter, we explore whether self-adaptation can be beneficial in dynamic optimisation. We specifically examine a tracking dynamic optima problem with changing structure that requires adjustable parameter settings. The structure, as previously discussed, refers to the number of relevant bits. This problem, the so-called *Dynamic Substring Matching* (DSM) problem, requires algorithms to successively find and hold the solutions that match a sequence of bit-flipping and length-varying target substrings (structure-changing optima) within specified evaluation budgets. We show that EAs with any fixed mutation rate get lost with constant probability somewhere during tracking the DSM problem (Lemma 5.5.1), resulting in an exponentially small probability of achieving the final optimum (Theorem 5.5.1). Therefore, variable mutation rates may be necessary to successfully track the optimum.

The primary contribution of this work lies in conducting the first rigorous study of self-adaptive parameter control mechanisms in dynamic optimisation. We analyse the (μ, λ) self-

adaptive EA, an algorithm studied in (Case and Lehre, 2020) and described as Algorithm 7 employing the fitness-first sorting partial order with a preference for high mutation rate (Definition 2.2.4(b)), the (μ, λ) selection (Algorithm 9), and the self-adapting mutation rate strategy presented in Algorithm 10 on solving the DSM problem. Our study shows that this algorithm can track every optimum in the DSM problem (Lemma 5.4.1) and achieve the final optimum with an overwhelmingly high probability (Theorem 5.4.1). Conversely, we demonstrate that static mutation-based EAs (both Algorithms 2 and 3) struggle with tracking the DSM problem.

Another contribution is a level-based theorem with tail bounds (Theorem 5.3.1). To assess the capacity of the self-adaptive EA in tracking dynamic optima, it is necessary to determine a lower bound of the probability of achieving the current optimum within the specified evaluation budget. To address our requirements, we develop a level-based theorem with tail bounds.

The chapter is organised as follows: Section 5.2 introduces the DSM function. Section 5.3 develops a level-based theorem with tail bounds for later analysis. Sections 5.4-5.5 demonstrate that the (μ, λ) self-adaptive EA can track dynamic optima of the DSM function, and show the inefficiency of static mutation-based EAs, respectively. The chapter concludes in Section 5.6.

5.2 Dynamic Substring Matching Problem

In dynamic environments, the number of pertinent bits may fluctuate (structure changing). This chapter considers the DSM problem, which involves this situation. Let $\varepsilon \in (0, 1)$, $k > 0$, $\varkappa \in \{0, 1\}^{\ell_1}$ where $\ell_1 \in [n - 1]$, and $m \in [n - \ell_1]$ be the parameters of the DSM problem, then the $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ problem aims to match a sequence of bit-flipping and length-varying

target substrings $(\mathscr{x}^i)_{i\in[4m]}$ in a sequence of corresponding evaluation budgets $(\mathcal{T}_i)_{i\in[4m]}$. The length of target substrings varies between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$ resulting in structure changing. The dynamics of the DSM problem is updated in four phases.

1. for $i \in [m]$, the previous target substring $\mathscr{x}^{i-1}$ and the current target substring $\mathscr{x}^i$ are the same length but one bit different: $\mathscr{x}^i$ generated by uniformly at random change one bit of $\mathscr{x}^{i-1}$;

2. for $i \in [m+1..2m]$, the target substrings are becoming longer: $\mathscr{x}^i$ generated by appending one random bit in the end of $\mathscr{x}^{i-1}$;

3. for $i \in [2m+1..3m]$, similar to stage (1): $\mathscr{x}^{i-1}$ and $\mathscr{x}^i$ are the same length but one-bit different;

4. for $i \in [3m+1..4m]$, the target substrings are becoming shorter: $\mathscr{x}^{i-1}$ generated by removing the last bit of $\mathscr{x}^{i-1}$ and uniformly at random flip one of the rest of bits.

Figure 5.1 illustrates a sequence of target substrings in an example DSM problem. The sequence of corresponding evaluation budgets $(\mathcal{T}_i)_{i\in[4m]}$ depends on the lengths of the target substrings, i.e., $kn^{\varepsilon}|\mathscr{x}^i|$. The target substrings are changed after evaluation budgets run out. We call a period between two times of the target change a phase. We assume that the algorithms start from a starting substring $\mathscr{x}^0$. The algorithms are required to find and hold solutions matching the current target substring before the target changes. The DSM problem is formally defined in Definition 5.2.1.

Definition 5.2.1. *Let $\mathscr{x}$ be some starting target substring where $|\mathscr{x}| =: \ell_1 \in [n-1]$, and m be a positive integer where $\ell_1 + m =: \ell_2 \leq n$. Let $(\mathscr{x}^i)_{i\geq 0}$ be a sequence of target substrings*

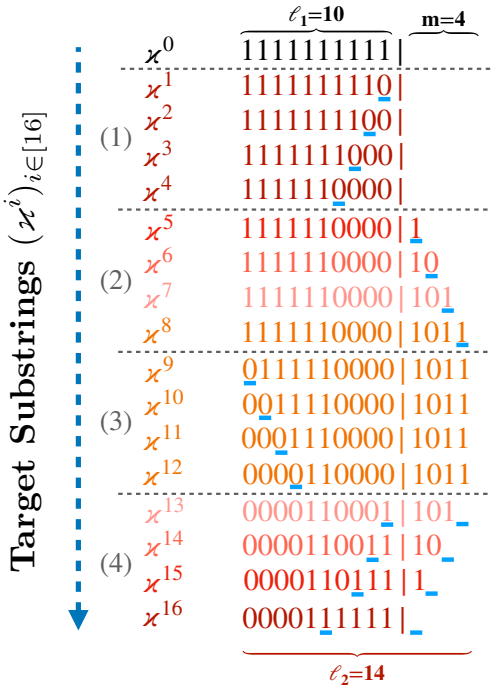

Figure 5.1: A sequence of target substrings in an example of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

generated by

$$\varkappa^i := \begin{cases} \varkappa & \text{if } i = 0, \\ z, \ \text{where } z \sim \mathrm{Unif}(\mathrm{N}(\varkappa^{i-1})) & \text{if } 1 \le i \le m, \\ \varkappa^{i-1} \diamond a, \ \text{where } a \sim \mathrm{Unif}(\{0,1\}) & \text{if } m+1 \le i \le 2m, \\ z, \ \text{where } z \sim \mathrm{Unif}(\mathrm{N}(\varkappa^{i-1})) & \text{if } 2m+1 \le i \le 3m, \\ z, \ \text{where } z \sim \mathrm{Unif}\left(\mathrm{N}\left(\varkappa^{i-1}_{1:(|\varkappa^{i-1}|-1)}\right)\right) & \text{if } 3m+1 \le i \le 4m. \end{cases} \tag{5.1}$$

Let $(\mathcal{T}_i)_{i\in\mathbb{N}}$ be a sequence of the numbers of evaluation moving from $\varkappa^{i-1}$ to $\varkappa^i$ (evaluation budget for $\varkappa^i$) generated by $\mathcal{T}_i := kn^{\varepsilon}|\varkappa_{i+1}|$, where $\varepsilon \in (0,1)$ and $k > 0$ are some constants. For $t \in \mathbb{N}$, the dynamic substring matching (DSM) problem with the starting target substring

$\varkappa$ is defined as:

$$DSM_t^{\varkappa,m,\varepsilon,k}(x) := \begin{cases} 2 & \text{if } \mathrm{M}(\varkappa(t), x) = 1, \\ 1 & \text{else if } \mathrm{M}(\varkappa'(t), x) = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where $\varkappa(t) := \varkappa^i$, *and* $\varkappa'(t) := \varkappa^{i-1}$, *for* $i = \begin{cases} 1 & \text{if } t \leq \mathcal{T}_1, \\ 1 + \max\left\{j \mid \sum_{i=1}^{j} \mathcal{T}_i \leq t\right\} & \text{otherwise.} \end{cases}$

Several methods to evaluate how well algorithms track dynamic optima have been proposed (Dang et al., 2017; Kötzing and Molter, 2012). Regarding the DSM problem, algorithms may fail to reach the final optimum if they lose track during some phases. Therefore, we define the criteria for tracking in Definition 5.2.2.

Definition 5.2.2. *A sequence of sets of solutions* $(Q_\tau)_{\tau\in\mathbb{N}}$, *where* $Q_\tau \in \mathcal{X}^\lambda$ *and* $\lambda \in \mathbb{N}$, *tracks the* $DSM^{\varkappa,m,\varepsilon,k}$ *if it begins with the initial set* Q_0, *where all solutions* x *satisfy* $\mathrm{M}(x, \varkappa) = 1$, *and at least one solution* x' *in* $Q_{\bar\tau}$ *satisfies* $\mathrm{M}(x', \varkappa^{4m}) = 1$, *where* $\bar\tau = \lceil(\sum_{i=1}^{4m} \mathcal{T}_i)/\lambda\rceil$ *denotes the end of the final phase.*

5.3 Level-based Theorem (Tail Bounds)

The level-based theorems (Corus et al., 2018; Dang et al., 2021b; B. Doerr and Kötzing, 2021) are general tools that provide an upper bound of the runtime of non-elitist algorithms which follow the scheme of Algorithm 11 with a population $P_\tau \in \mathcal{X}^\lambda$, where $\mathcal{X}^\lambda$ is the space of all populations of size λ (see Section 2.2.3.1). We employ the same notations as described in Section 2.2.3.1: Assume that the search space $\mathcal{X}$ is partitioned into ordered disjoint subsets (called levels) $A_1, \ldots, A_m$. Let $A_{\geq j} := \cup_{k=j}^{m} A_k$ be the search points in level j and higher, and let D be some mapping from the set of all possible populations

$\mathcal{X}^\lambda$ into the space of probability distributions of $\mathcal{X}$. Given any subset $A \subseteq \mathcal{X}$, we define $|P_\tau \cap A| := |\{i \mid P_\tau(i) \in A\}|$, i.e., the number of individuals in P_τ that belong to A. To estimate an upper bound on the runtime using level-based theorems, three conditions must typically be satisfied: (G1) requires the probability of level "upgrading", i.e., creating an individual in higher levels; (G2) requires the probability of the number of individuals in higher levels "growing"; (G3) requires a sufficient population size. Specifically, in (Dang et al., 2021b), a new level-based theorem has been proposed to address the issue of "deceptive" regions B that contains individuals with a higher selection probability but at a lower level. The theorem includes an additional condition (G0) that requires the probability of producing a "deceptive" individual to decrease if too many such individuals are in the population.

In the theory of EC, besides the expected runtime, tail bounds can be other performance criteria of EAs, which indicate the probability of runtime within a given evaluation budget. Several theoretical tools were developed to derive tail bounds on the runtime of EAs (B. Doerr and L. A. Goldberg, 2010; Lehre and Sudholt, 2020; Lehre and Witt, 2021; Oliveto and Witt, 2011). In this chapter, we are interested in the probability that an algorithm finds the current optimum in a specific evaluation budget. To address our requirements, we derive a level-based theorem with tail bounds (shown in Theorem 5.3.1). We assume that the search space $\mathcal{X}$ partitions into $B, A_0, A_1, \ldots, A_m$, then assume that the initial population P_0 contains sufficient individuals in level A_1 and the termination condition is to gain a population with enough individuals in A_m. Theorem 5.3.1 also considers a "deceptive region" B (condition (C0)). The assumption that there are not too many individuals in the region B holds for the initial population , i.e., $|P_0 \cap B| \leq \psi_0 \lambda$. Conditions (C1)-(C2) correspond to conditions (G1)-(G2) in the original level-based theorems (Corus et al., 2018; Dang et al., 2021b; B. Doerr and Kötzing, 2021), and no condition corresponds to condition (G3) since the tail bound is determined by the population size λ. Eventually, Theorem 5.3.1 gives the lower bound of the probability of runtime within $\eta\mathcal{T}$ evaluation times by choosing η and λ. Compared

to the original level-based theorems, the runtime from Theorem 5.3.1 is mainly one more multiplicative factor $\eta\lambda^{17/\delta^3}$, where η is used to tune the upper bound for $\Pr(T \leq \eta\mathcal{T})$. In order to fulfil the requirements into our scenario in Section 5.4, we refrain from employing the multiple restarts argument that was utilised in the proofs of the original level-based theorems (B. Doerr and L. A. Goldberg, 2010; Lehre and Sudholt, 2020; Lehre and Witt, 2021; Oliveto and Witt, 2011).

Theorem 5.3.1. *Let $(B, A_0, A_1, \ldots, A_m)$ be a partition of $\mathcal{X}$. Suppose there exist $z_1, \ldots, z_{m-1}$, $\delta \in (0,1)$, and $\gamma_0, \psi_0 \in (0,1)$, such that the following conditions hold for any population $P \in \mathcal{X}^\lambda$ in Algorithm 11,*

(C0) for all $\psi \in [\psi_0, 1]$, if $|P \cap B| \leq \psi\lambda$, then

$$\Pr_{y\sim D(P)}(y \in B) \leq (1-\delta)\psi,$$

(C1) for all $j \in [m-1]$, if $|P \cap B| \leq \psi_0\lambda$ and $|P \cap A_{\geq j}| \geq \gamma_0\lambda$, then

$$\Pr_{y\sim D(P)}(y \in A_{\geq j+1}) \geq z_j,$$

(C2) for all $j \in [0..m-1]$, and $\gamma \in [1/\lambda, \gamma_0]$ if $|P \cap B| \leq \psi_0\lambda$ and $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ and $|P \cap A_{\geq j+1}| \geq \gamma\lambda$, then

$$\Pr_{y\sim D(P)}(y \in A_{\geq j+1}) \geq (1+\delta)\gamma.$$

Let $T := \min\{\tau\lambda \mid |P_\tau \cap A_m| \geq \gamma_0\lambda \text{ and } |P_\tau \cap B| \leq \psi_0\lambda\}$, and assume the algorithm with population size $\lambda \in \mathbb{N}$ and an initial population P_0 satisfying $|P_0 \cap A_1| \geq \gamma_0\lambda$ and $|P_0 \cap B| \leq \psi_0\lambda$, then

$$\Pr(T \leq \eta\tau) > \left(1 - 2\eta\tau e^{-\delta^2 \min\{\psi_0,\gamma_0\}\lambda/4}\right)\left(1 - me^{-\eta\rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}}\right)$$

for any $\eta \in \left(0, e^{\delta^2 \min\{\psi_0,\gamma_0\}\lambda/4}/\mathcal{T}\right)$, where

$$\rho = \frac{e^{\delta^2/8}}{e^{\delta^2/8}-1} \text{ and } \mathcal{T} := \lambda^{17/\delta^3}\left(\sum_{j=1}^{m-1}\frac{1}{z_j} + m\lambda\left(\frac{\ln(\gamma_0\lambda)}{\ln(1+\delta/2)}+1\right)\right).$$

To apply Theorem 5.3.1, we need to satisfy conditions (C0)-(C2), which is the same as applying the original level-based theorems, then we need to choose η and λ to gain a desired upper bound for $\Pr(T \leq \eta\mathcal{T})$. For example, if conditions (C0)-(C2) hold for some constants $\delta, \psi_0, \gamma_0 \in (0,1)$, $z_j := \Omega(1/n)$ and $m \in \text{poly}(n)$ by an instantiated algorithm with population size λ, then the upper bound for $\Pr(T \leq \eta\mathcal{T})$ is $\left(1 - \eta\mathcal{T}e^{-\Omega(\lambda)}\right)\left(1 - me^{-\Omega(\eta)}\right)$. If choosing $\lambda = \eta = n^{\varepsilon}$ for a constant $\varepsilon \in (0,1)$, then $\Pr(T \leq \eta\mathcal{T}) = 1 - e^{-\Omega(n^{\varepsilon})}$. In Section 5.4, we will use Theorem 5.3.1 to prove that the (μ, λ) self-adaptive EA can generate enough individuals matching the current target substring in evaluation budgets from the previous target substring with an overwhelmingly high probability. In comparison to the original level-based theorems, Theorem 5.3.1 typically necessitates a larger population size, e.g., cn^{ε}, in order to achieve a sufficiently large tail probability, i.e., $1 - e^{-\Omega(n^{\varepsilon})}$, where constants $c, \varepsilon > 0$.

Now we informally explain the proof idea. Pessimistically, the algorithm gradually increases its level from A_1 to A_m. There are m steps from A_1 to A_m. In each step, the algorithm needs to generate a higher-level individual, and then accumulate such individuals until the population contains a sufficient number of such individuals. We estimate the lower bound of the successful probability of each step in certain evaluation times. Next, we consider the probability of the "failure events", i.e., the algorithm goes back to the previous level, or produces too many "bad" (B region) individuals during the algorithm running. Eventually, we compute the lower bound of the probability that every step is completed without "failure events".

Proof of Theorem 5.3.1. We first define some notation for later use. For any level $j \in [m]$ and $\tau \in \mathbb{N}_0$, let the random variable $X_\tau^{(j)} := |P_\tau \cap A_{\geq j}|$ denote the number of individuals in levels $A_{\geq j}$ at time τ. Let the random variable $Y_\tau := |P_\tau \cap B|$ denote the number of individuals in the region B at time τ. The level J_τ of population P_τ at time τ is defined as

$J_\tau := \max\{j \in [m] : X_\tau^j \geq \gamma_0\lambda\}$. We say the algorithm *upgrades its level* in h generations if $J_{\tau+h} \geq J_\tau + 1$.

We now estimate the probability that no "failure events" occur. More precisely, "failure events" include less than $\gamma_0\lambda$ individuals of population P_τ in level $A_{\geq J_{\tau-1}}$ and more than $\psi_0\lambda$ individuals of population P_τ in the region B. Given $\tau \geq 1$, let $\mathcal{E}_\tau$ be the event that $J_\tau \geq J_{\tau-1}$ and $Y_\tau \leq \psi_0\lambda$, and define $\hat{\mathcal{E}}_\tau := \mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_\tau$. By condition (C2), the random variable $(X_\tau^{(J_{\tau-1})} \mid \hat{\mathcal{E}}_{\tau-1})$ stochastically dominates $Z \sim \mathrm{Bin}(\lambda, (1+\delta)\gamma_0)$. By condition (C0), the random variable $(Y_\tau \mid \hat{\mathcal{E}}_{\tau-1})$ is stochastically dominated by $Z' \sim \mathrm{Bin}(\lambda, (1-\delta)\psi_0)$. Therefore, using a union bound, we have

$$\begin{aligned}
\Pr\left(\hat{\mathcal{E}}_\tau \mid \hat{\mathcal{E}}_{\tau-1}\right) &\geq 1 - \Pr\left(J_\tau < J_{\tau-1} \mid \hat{\mathcal{E}}_{\tau-1}\right) - \Pr\left(Y_\tau > \psi_0\lambda \mid \hat{\mathcal{E}}_{\tau-1}\right) \\
&= 1 - \Pr\left(X_\tau^{(J_{\tau-1})} < \gamma_0\lambda \mid \hat{\mathcal{E}}_{\tau-1}\right) - \Pr\left(Y_\tau > \psi_0\lambda \mid \hat{\mathcal{E}}_{t-1}\right) \\
&\geq 1 - \Pr\left(Z < \gamma_0\lambda\right) - \Pr\left(Z' > \psi_0\lambda\right).
\end{aligned}$$

We then use Chernoff bounds to estimate the upper bounds of $\Pr(Z < \gamma_0\lambda)$ and $\Pr(Z' > \psi_0\lambda)$:

$$\begin{aligned}
\Pr\left(Z < \gamma_0\lambda\right) &= \Pr\left(Z < \gamma_0\lambda(1+\delta)\left(1 - \frac{\delta}{1+\delta}\right)\right) \\
&= \Pr\left(Z < E[Z]\left(1 - \frac{\delta}{1+\delta}\right)\right) \\
&< e^{-\frac{\delta^2}{(1+\delta)^2}E[Z]/2} \\
&= e^{-\frac{\delta^2\gamma_0\lambda}{2(1+\delta)}} \\
&< e^{-\delta^2\gamma_0\lambda/4},
\end{aligned}$$

and

$$\begin{aligned}
\Pr\left(Z' > \psi_0\lambda\right) &= \Pr\left(Z' > \psi_0\lambda(1-\delta)\left(1+\frac{\delta}{1-\delta}\right)\right) \\
&= \Pr\left(Z' > E[Z']\left(1+\frac{\delta}{1-\delta}\right)\right) \\
&< e^{-\frac{\delta^2}{(1-\delta)^2}E[Z']/\left(2+\frac{\delta}{1-\delta}\right)} \\
&= e^{-\frac{\delta^2\psi_0\lambda}{1-\delta}/\left(2+\frac{\delta}{1-\delta}\right)} \\
&< e^{-\frac{\delta^2\psi_0\lambda}{2-\delta}} \\
&< e^{-\delta^2\psi_0\lambda/2}.
\end{aligned}$$

Thus,

$$\begin{aligned}
\Pr\left(\hat{\mathcal{E}}_\tau \mid \hat{\mathcal{E}}_{\tau-1}\right) &> 1 - e^{-\delta^2\gamma_0\lambda/4} - e^{-\delta^2\psi_0\lambda/2} \\
&\geq 1 - 2e^{-\delta^2\min\{\gamma_0,\psi_0\}\lambda/4}. \qquad (5.3)
\end{aligned}$$

Next, we consider the number of generations to increase the level of the algorithm. We note that, by condition (C2), the random variable $(X_{\tau+1}^{(J_\tau+1)} \mid X_\tau^{(J_\tau+1)} \geq \gamma\lambda, \hat{\mathcal{E}}_\tau)$ stochastically dominates $Z'' \sim \mathrm{Bin}\left(\lambda, (1+\delta)\min\{\gamma,\gamma_0\}\right)$ for any $\gamma\lambda \geq 1$. Therefore, if $1 \leq \gamma\lambda \leq \gamma_0\lambda$, we have

$$\begin{aligned}
\Pr\left(X_{\tau+1}^{(J_\tau+1)} \geq (1+\delta/2)\gamma\lambda \mid X_\tau^{(J_\tau+1)} \geq \gamma\lambda, \hat{\mathcal{E}}_\tau\right) &\geq \Pr\left(Z'' \geq (1+\delta/2)\gamma\lambda\right) \\
&= 1 - \Pr\left(Z'' < (1+\delta/2)\gamma\lambda\right) \\
&= 1 - \Pr\left(Z'' < (1+\delta)\gamma\lambda\left(1-\frac{\delta}{2(1+\delta)}\right)\right) \\
&= 1 - \Pr\left(Z'' < E[Z'']\left(1-\frac{\delta}{2(1+\delta)}\right)\right)
\end{aligned}$$

then by a Chernoff bound,

$$\begin{aligned}
&> 1 - e^{-\frac{\delta^2}{4(1+\delta)^2} E[Z'']} \\
&= 1 - e^{-\frac{\delta^2 \gamma\lambda}{4(1+\delta)^2}} \\
&> 1 - e^{-\delta^2 \gamma\lambda/8} \\
&\geq 1 - e^{-\delta^2/8}.
\end{aligned}$$

Then we define $h := \lceil \log_{1+\delta/2}(\gamma_0\lambda) \rceil \leq \ln(\gamma_0\lambda)/\ln(1+\delta/2) + 1$. Informally, h is the number of consecutive "growing" steps required for the algorithm to upgrade its level. If there exists an individual of the population P_τ in level $A_{\geq J_\tau+1}$, then the probability of the algorithm upgrading its level in h generations is at least

$$\begin{aligned}
&\Pr\left(X_{\tau+h}^{(J_\tau+1)} \geq \gamma_0\lambda \mid X_t^{(J_\tau+1)} \geq 1, \hat{\mathcal{E}}_{\tau+h-1}\right) \\
\geq &\prod_{i=1}^{h} \Pr\left(X_{\tau+i}^{(J_\tau+1)} \geq (1+\delta/2)^i \mid X_{\tau+i-1}^{(J_\tau+1)} \geq (1+\delta/2)^{i-1}, \hat{\mathcal{E}}_{\tau+i-1}\right) \\
\geq &\prod_{i=1}^{h} \left(1 - e^{-\delta^2/8}\right) \\
= &\left(1 - e^{-\delta^2/8}\right)^h \\
= &\rho^{-h}
\end{aligned}$$

where we define $\rho := \frac{e^{\delta^2/8}}{e^{\delta^2/8}-1} > 1/(1-1/e)$ by $\delta \in (0,1)$.

Then, we note that by condition (C1) and following by $(1+x/n)^n \leq e^x$ for $n \geq 1, |x| \leq n$, for any t,

$$\begin{aligned}
\Pr\left(\exists s_j \leq \lceil 1/(z_j\lambda) \rceil : X_{\tau+s_j}^{(J_\tau+1)} \geq 1 \mid \hat{\mathcal{E}}_{\tau+s_j-1}\right) &\geq 1 - (1-z_j)^{\lambda\lceil 1/(z_j\lambda)\rceil} \\
&\geq 1 - 1/e \\
&> \rho^{-1}.
\end{aligned}$$

Thus, if we define $q(j) := \lceil 1/(z_j\lambda) \rceil + h$, we obtain the following lower bound for the

probability of the algorithm upgrading its level in $q(j)$ generations,

$$\begin{aligned}
&\Pr\left(J_{\tau+q(J_\tau)} \geq J_\tau + 1 \mid \hat{\mathcal{E}}_{\tau+q(J_\tau)-1}\right) \\
\geq &\Pr\left(\exists s_j \leq \lceil 1/(z_j\lambda)\rceil : X_{\tau+s_j}^{(J_\tau+1)} \geq 1 \mid \hat{\mathcal{E}}_{\tau+s_j-1}\right) \cdot \Pr\left(X_{\tau+q(J_\tau)}^{(J_\tau+1)} \geq \gamma_0\lambda \mid X_\tau^{J_\tau+1} \geq 1, \hat{\mathcal{E}}_{t+q(J_\tau)-1}\right) \\
\geq &\rho^{-h-1}.
\end{aligned}$$

In particular, we have the following upper bound for the probability that the algorithm does not upgrade its level in $\eta\lambda^{17/\delta^3}q(J_\tau)$ generations,

$$\Pr\left(J_{\tau+\eta\lambda^{17/\delta^3}q(J_\tau)} \leq J_\tau \mid \hat{\mathcal{E}}_{t+\eta\lambda^{17/\delta^3}q(J_\tau)}\right) \leq \left(1-\rho^{-h-1}\right)^{\eta\lambda^{17/\delta^3}}.$$

Define $\hat{q}(j) := \eta\lambda^{17/\delta^3}\sum_{i=1}^{j} q(i)$, and note that $\mathcal{T} \geq \hat{q}(m-1)\lambda/\eta$. We then have

$$\begin{aligned}
\Pr\left(T \leq \eta\mathcal{T} \mid \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) &\geq \Pr\left(\bigcap_{j=1}^{m-1}\left(J_{\hat{q}(j)} \geq j+1\right) \mid \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \\
&= 1 - \Pr\left(\bigcup_{j=1}^{m-1}\left(J_{\hat{q}(j)} < j+1\right) \mid \hat{\mathcal{E}}_{\eta\mathcal{T}}\right)
\end{aligned}$$

by a union bound,

$$\begin{aligned}
&\geq 1 - \sum_{j=1}^{m-1}\Pr\left(J_{\hat{q}(j)} < j+1 \mid J_{\hat{q}(j-1)} \geq j, \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \\
&\geq 1 - \sum_{j=1}^{m-1}\Pr\left(J_{\hat{q}(j)} < j+1 \mid J_{\hat{q}(j-1)} = j, \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \\
&= 1 - \sum_{j=1}^{m-1}\Pr\left(J_{\hat{q}(j-1)+\eta\lambda^{17/\delta^3}q(j)} \leq j \mid J_{\hat{q}(j-1)} = j, \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \\
&\geq 1 - m\left(1-\rho^{-h-1}\right)^{\eta\lambda^{17/\delta^3}}.
\end{aligned}$$

Recall Eq. (5.3), we know that, using the Bernoulli's inequality $(1+x)^r \geq 1+rx$ for $-1 < x < 0$ and $r \in \mathbb{N}_0$,

$$\begin{aligned}
\Pr\left(\hat{\mathcal{E}}_{\eta\mathcal{T}}\right) &\geq \prod_{\tau=1}^{\eta\mathcal{T}}\Pr\left(\hat{\mathcal{E}}_\tau \mid \hat{\mathcal{E}}_{\tau-1}\right) \\
&\geq \left(1-2e^{-\delta^2\min\{\gamma_0,\psi_0\}\lambda/4}\right)^{\eta\mathcal{T}} \\
&\geq 1 - 2\eta\mathcal{T}e^{-\delta^2\min\{\gamma_0,\psi_0\}\lambda/4}.
\end{aligned}$$

Hence,

$$\begin{aligned}\Pr\left(T \leq \eta\mathcal{T}\right) &\geq \Pr\left(T \leq \eta\mathcal{T} \mid \hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \cdot \Pr\left(\hat{\mathcal{E}}_{\eta\mathcal{T}}\right) \\ &\geq \left(1 - m\left(1-\rho^{-h-1}\right)^{\eta\lambda^{17/\delta^3}}\right) \cdot \left(1 - 2\eta\mathcal{T}e^{-\delta^2\min\{\gamma_0,\psi_0\}\lambda/4}\right).\end{aligned}$$

Since

$$\begin{aligned}\rho^{-h-1} &\geq \rho^{-\frac{\ln(\gamma_0\lambda)}{\ln(1+\delta/2)}-2} \\ &\geq \rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-\frac{\ln(\lambda)}{\ln(1+\delta/2)}-2} \\ &= \rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}\lambda^{-\frac{1}{\log_\rho(e)\ln(1+\delta/2)}} \\ &= \rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}\lambda^{-\frac{\delta^2/8-\ln(e^{\delta^2/8}-1)}{\ln(1+\delta/2)}} \\ &\geq \rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}\lambda^{-\frac{\delta^2/8+8/(2\delta^2)}{\delta/2-\delta^2/4}} \\ &\geq \rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}\lambda^{-17/\delta^3},\end{aligned}$$

then by $(1+x/n)^n \leq e^x$ for $n \geq 1, |x| \leq n$,

$$\Pr\left(T \leq \eta\mathcal{T}\right) \geq \left(1 - me^{-\eta\rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}}\right)\left(1 - 2\eta\mathcal{T}e^{-\delta^2\min\{\gamma_0,\psi_0\}\lambda/4}\right).$$

□

5.4 The Self-adaptive EA on DSM

We now show that the (μ, λ) self-adaptive EA can track the DSM problem. To prove this, we first derive Lemma 5.4.1 which shows that the algorithm obtains a population with a sufficient number of individuals matching the current target substring within the evaluation budget with an overwhelming probability in a phase if it successfully tracks in the previous phases. Then, Theorem 5.4.1 via Lemma 5.4.1 states the efficiency of the (μ, λ) self-adaptive EA on solving the DSM problem.

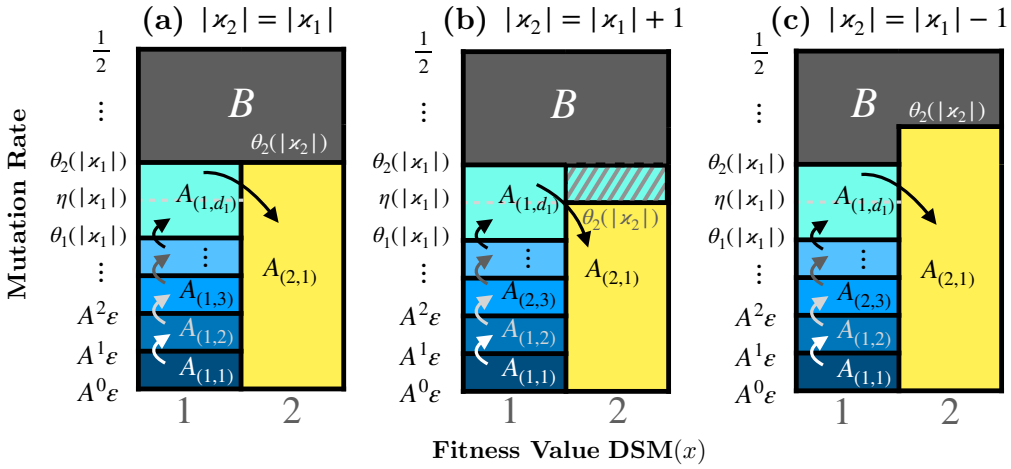

Figure 5.2: Level partitions for three cases in the proof of Lemma 5.4.1. Note that level A_0 is omitted in the subfigures.

Case and Lehre (2020) proved that the runtime of the (μ, λ) self-adaptive EA on optimising $\textsc{LeadingOnes}_k(x) := \sum_{i=1}^{k} \prod_{j=1}^{i} x_j$ is $O(k^2)$, where k is unknown for the algorithm. They divided the search space $\mathcal{Y}$ into a two-dimensional level partition, fitness levels and mutation rate sub-levels. Informally, to estimate the runtime, they counted the number of generations to increase the mutation rate until it is sufficiently high, and then counted the number of generations until the solution improved. It is well-known that too high mutation rates might fail non-elitist EAs. More precisely, there exist error thresholds for non-elitist EAs, which lead to exponential runtime on any function with a unique optimum if the mutation rate exceeds the threshold (additional details can be found in Section 2.2.3.2). Since the (μ, λ) self-adaptive EA applies a non-elitist selection mechanism, they also defined a "bad" region B which contains individuals with too high mutation rates. They assumed that the algorithms restart from the first level if there are too many individuals in the B region. However, algorithms cannot be allowed to restart, since algorithms should keep the previous optimal solution in our scenario. Therefore, it is essential to limit the number of individuals in the B region in every generation to avoid losing track while tracking the dynamic function.

Lemma 5.4.1 provides the probability of obtaining sufficient individuals matching the

current target substring $\varkappa_2$ in a phase of the $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ function within a given evaluation budget $\mathcal{T}_2'$, by assuming that enough individuals match the previous target substring $\varkappa_1$ at the beginning of the phase. Our proof idea is similar to (Case and Lehre, 2020). We first define the level partition in a phase of the DSM problem. We partition the search space $\mathcal{Y}$ into three fitness levels: $A_2 =: \{x \mid \mathrm{M}(x, \varkappa_2) = 1\}$, $A_1 =: \{x \mid \mathrm{M}(x, \varkappa_1) = 1\} \backslash A_2$, and $A_0 =: \mathcal{X} \backslash (A_2 \cup A_1)$. We further divide A_1 into mutation rate sub-levels. Finally, we use the three threshold values θ_2, η, θ_1 based on the lengths of $\varkappa_1, \varkappa_2$ to describe mutation rate sub-levels, which will be defined later. Informally, a mutation rate between θ_1 and θ_2 is suitable for increasing fitness level, i.e., from A_1 to A_2. Let $d_1 := \min\left\{\ell \in \mathbb{N} \mid \epsilon A^\ell \geq \theta_1(|\varkappa_1|)\right\}$ be the number of sub-levels of A_1, and let A_2 only contain one sub-level $A_{(2,1)}$ with the number of sub-levels $d_2 := 1$. We cannot allow too many individuals with too high mutation rates, thus a "bad" region B is defined which contains all search point above a threshold value θ_2. Additionally, the important assumption in Lemma 5.4.1 is $|P \cap B| \leq \psi_0 \lambda$ for the population at beginning of the phase. In overall,

$$A_{(1,\ell)} := A_1 \times \left[A^{\ell-1}\epsilon, \min\left(A^\ell \epsilon, \theta_1(|\varkappa_1|)\right)\right) \text{ for } \ell \in [d_1 - 1]; \tag{5.4}$$

if $|\varkappa_2| = |\varkappa_1| + 1$, we define

$$\begin{aligned} A_{(1,d_1)} := A_1 \times \left[\theta_1(|\varkappa_1|), \min\left(\frac{1}{2}, \theta_2(|\varkappa_1|)\right)\right] \cup \\ A_2 \times \left(\min\left(\frac{1}{2}, \theta_2(|\varkappa_2|)\right), \min\left(\frac{1}{2}, \theta_2(|\varkappa_1|)\right)\right], \end{aligned} \tag{5.5}$$

if $|\varkappa_2| = |\varkappa_1|$ or $|\varkappa_2| = |\varkappa_1| - 1$, we define

$$A_{(1,d_1)} := A_1 \times \left[\theta_1(|\varkappa_1|), \min\left(\frac{1}{2}, \theta_2(|\varkappa_1|)\right)\right], \tag{5.6}$$

$$A_{(2,1)} := A_2 \times \left[\epsilon, \min\left(A^\ell \epsilon, \theta_2(|\varkappa_2|)\right]\right); \tag{5.7}$$

$$B := \cup_{i=1}^2 A_i \times \left(\theta_2(|\varkappa^i|), \frac{1}{2}\right]. \tag{5.8}$$

Note that the sub-levels definition depends on the lengths of substrings $\varkappa_1$ and $\varkappa_2$. Figure 5.2 illustrates the three cases of the definition of the level partitions: (a) $|\varkappa_2| = |\varkappa_1|$ correspond-

ing stages (1) and (3), (b) $|\varkappa_2| = |\varkappa_1| + 1$ corresponding stages (2), and (c) $|\varkappa_2| = |\varkappa_1| - 1$ corresponding stages (4).

We apply the same threshold values defined in (Case and Lehre, 2020): for $j \geq 1$, let

$$\eta(j) := \frac{1}{2A}\left(1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{1/j}\right) \tag{5.9}$$

$$\theta_1(j) := b\eta(j) \tag{5.10}$$

$$\theta_2(j) := 1 - q^{1/j}, \text{ where} \tag{5.11}$$

$$\alpha_0 := \lambda/\mu, q := \frac{1-\zeta}{\alpha_0}, r_0 := \frac{1+\delta}{\alpha_0(1-p_{\text{inc}})}, \text{ and } \zeta := 1 - \alpha_0(r_0)^{1+\sqrt{r_0}}. \tag{5.12}$$

Lemma 5.4.1. *Let $\varepsilon, a, \beta \in (0,1)$ and $k > 0$ be some constants satisfying $1/34 \leq a < \beta \leq 1$. Let $\varkappa \in \{0,1\}^{\ell_1}$ be some starting target substring where $\ell_1 \in \Theta(n^a)$, and $m \in \Theta(n^\beta)$. Let $\varkappa_1$ and $\varkappa_2$ be any two neighbouring substrings of the sequence of target substrings $(\varkappa^i)_{i\geq 0}$ in the $DSM^{\varkappa,m,\varepsilon,k}$. Let $\mathcal{T}_2'$ be the evaluation budget for $\varkappa_2$ in the $DSM^{\varkappa,m,\varepsilon,k}$. Suppose that the initial population P_0 in the (μ,λ) self-adaptive EA satisfies $|P_0 \cap A_{\geq(1,1)}| \geq \gamma_0\lambda$ and $|P_0 \cap B| \leq \psi_0\lambda$, where γ_0 and ψ_0 are constants in $(0,1)$. Then, there exist constants δ and ξ in $(0,1)$ such that the probability of the (μ,λ) self-adaptive EA with parameters $\lambda, \mu = \Theta(n^\xi)$, $\lambda/\mu = \alpha_0 \geq 4$, $A > 1$, $0 < b < 1/(1+\sqrt{1/\alpha_0(1-p_{\text{inc}})})$, $(1+\delta)/\alpha_0 < p_{\text{inc}} < 2/5$, $\epsilon = b'/n$ for any constant $b' > 0$, where A and b are constants, obtaining a population P_t with $|P_t \cap A_{\geq(2,1)}| \geq \gamma_0\lambda$ and $|P_t \cap B| \leq \psi_0\lambda$, where $t \leq \mathcal{T}_2'$, is $1 - e^{-\Omega(n^\xi)}$.*

We apply the level-based theorem with tail bounds (Theorem 5.3.1) to prove Lemma 5.4.1. We use serval lemmas to break down the proof of Lemma 5.4.1. Lemma 5.4.2 implies the probability of selecting individuals in a self-adaptive population.

Lemma 5.4.2. *(Selection probability) If at least $\gamma\lambda$ individuals of a population P_τ of the (μ,λ) self-adaptive EA with $\lambda/\mu = \alpha_0$ are in $A_{\geq(i,\ell)}$ and at most $\psi_0\lambda$ individuals in B satisfying $\psi_0 + \gamma \leq 1/\alpha_0$ for $\psi_0, \gamma \in (0,1]$, then for all $i \in \{1,2\}$ and $\ell \in [d_i]$, a parent $(x, \chi/n)$ selected in the (μ,λ) selection step satisfies, $\Pr\left((x,\chi/n) \in A_{\geq(i,\ell)}\right) \leq \gamma\alpha_0$.*

Proof. By the definition of levels, for any $(x_1, \chi_1) \in A_{(1,j)}$ and $(x_2, \chi_2) \in A_{(1,k)}$, we know that $(x_1, \chi_1) \preceq_{P_\tau, f} (x_2, \chi_2)$ if and only if $1 \leq j \leq k < d_1$. For any $(x_1, \chi_1) \in A_{(1,j)}$ and $(x_2, \chi_2) \in A_{(2,1)}$, we know that $(x_1, \chi_1) \preceq_{P_\tau, f} (x_2, \chi_2)$ where $j \in [d_1]$. We also pessimistically assume that $(x_b, \chi_b) \succeq_{P_\tau, f} (x, \chi)$ where $(x_b, \chi_b) \in B$ and $(x, \chi) \in \mathcal{Y} \backslash B$. Thus, the higher level individual alway has a higher rank after sorting in the (μ, λ) self-adaptive EA, except individuals in the B region. Since $0 < \gamma\lambda \leq \mu - \psi_0\lambda$, all $\gamma\lambda$ individuals in $A_{\geq(i,\ell)}$ can be selected with probability $1/\mu$ in the (μ, λ) selection. Thus the probability of selecting a parent $(x, \chi/n) \in A_{\geq(i,\ell)}$ is at least $\gamma\lambda/\mu = \gamma\alpha_0$. □

Lemma 5.4.3 corresponds to condition (C0) in Theorem 5.3.1.

Lemma 5.4.3. *(Condition (C0)) Assume that the parameters A, b and p_{inc} satisfy the constraints in Lemma 5.4.1. Then there exists a constant $\delta \in (0, 1)$ such that if the (μ, λ) selection step of the (μ, λ) self-adaptive EA selects a parent $(x, \chi/n)$, then the offspring $(x', \chi'/n)$ created by self-adapting mutation rate and mutating bitstring satisfies,* $\Pr((x', \chi'/n) \in B) \leq \frac{1-\zeta}{\alpha_0}$.

Proof. We divide into two cases based on the individual created by mutating bitstring in the (μ, λ) self-adaptive EA (after selection and mutation rate adaptation):

- If $(x, \chi'/n) \in B$, then $\chi'/n > \theta_2(|\varkappa_j|)$ and $\mathrm{M}(x, \varkappa_j) = 1$ for $j \in \{1, 2\}$. The probability that $(x', \chi'/n)$ still in B, i.e., does not flip any matched bit, is $(1 - \chi'/n)^{|\varkappa_j|}$. By the definition $\theta_2(|\varkappa_j|)$, such probability is at most $\frac{1-\zeta}{\alpha_0}$.
- If $(x, \chi'/n) \notin B$, then it is necessary to match any substring or switch the matching substring, i.e., at least flipping only one matched bit, to obtain $(x', \chi'/n) \in B$. This probability is at most $(1 - \chi'/n)^{|\varkappa_j|-1} \frac{\chi'}{n} < (1 - \chi'/n)^{|\varkappa_j|} < \frac{1-\zeta}{\alpha_0}$.

□

Lemma 5.4.4 corresponds to conditions (C2) in Theorem 5.3.1. Lemma 5.4.4 looks similar to Lemma 4 in (Case and Lehre, 2020), but their proofs are different since the definitions of the level partitions are different.

Lemma 5.4.4. *(Condition (C2)) Assume that the parameters A, b and p_{inc} satisfy the constraints in Lemma 5.4.1. Then there exists a constant $\delta \in (0, 1/10)$ such that for all $j \in \{1, 2\}$ and $\ell \in [d_j]$, if the (μ, λ) selection step of the (μ, λ) self-adaptive EA selects a parent $(x, \chi/n) \in A_{\geq(i,\ell)}$, then the offspring $(x', \chi'/n)$ created by self-adapting mutation rate and mutating bitstring satisfies,* $\Pr\left((x', \chi'/n) \in A_{\geq(i,\ell)}\right) \geq \frac{1+\delta}{\alpha_0}$.

Proof. If the selected parent $(x, \chi/n)$ is in $A_{(2,1)}$, then, by the definition of the levels, we know $\chi/n \leq \theta_2(|\varkappa_2|)$. To estimate the probability of producing an offspring $(x', \chi'/n) \in A_{\geq(i,\ell)}$, it is sufficient to only consider the case that the mutation rate reduces $\chi' = b\chi$, and no matched bit is flipped, in which their probabilities are $1 - p_{\text{inc}}$ and $\frac{1+\delta}{\alpha_0(1-p_{\text{inc}})}$ (by Lemma A.2.12 (8)), respectively. Thus, $\Pr\left((x', \chi'/n) \in A_{\geq(i,\ell)}\right) \geq \frac{1+\delta}{\alpha_0}$.

If the selected parent $(x, \chi/n)$ in $A_{(1,d_1)}$, then we know $\theta_1(|\varkappa_1|) \leq \chi/n \leq \theta_2(|\varkappa_1|)$ by the levels definition. We distinguish two cases based on the mutation rate:

- $\theta_1(|\varkappa_1|) \leq \chi/n \leq \eta(|\varkappa_1|)$: Since $(x, A\chi)$ is still in $A_{(1,d_1)}$ by Lemma A.2.12 (3), we only consider the case that the mutation rate increases $\chi' = A\chi$ and no matched bit is flipped, in which the probabilities are p_{inc} and $\frac{1+\delta}{\alpha_0 p_{\text{inc}}}$ (by Lemma A.2.12 (7)), respectively. Then, $\Pr\left((x', \chi'/n) \in A_{\geq(1,d_1)}\right) \geq \frac{1+\delta}{\alpha_0}$.

- $\eta(|\varkappa_1|) \leq \chi/n \leq \theta_2(|\varkappa_1|)$: Since $(x, b\chi)$ is still in $A_{(1,d_1)}$ by Lemma A.2.12 (4), we only consider the case that the mutation rate reduces $\chi' = b\chi$ and no matched bit is flipped, in which the probabilities are $1 - p_{\text{inc}}$ and $\frac{1+\delta}{\alpha_0(1-p_{\text{inc}})}$ (by Lemma A.2.12 (8)), respectively. Then this probability is at least $\frac{1+\delta}{\alpha_0}$.

If the selected parent $(x, \chi/n)$ in $A_{(1,\ell)}$ for $\ell \in [d_1 - 1]$, then we know $\chi/n < \theta_1(|\varkappa_1|) < \eta(|\varkappa_1|)$ by the definition of the levels. We only consider the case that the mutation rate increases $\chi' = A\chi$ and no matched bit is flipped, in which the probabilities are p_{inc} and $\frac{1+\delta}{\alpha_0 p_{\text{inc}}}$ (by Lemma A.2.12 (7)), respectively. Overall, $\Pr\left((x', \chi'/n) \in A_{\geq(1,\ell+1)}\right) \geq \frac{1+\delta}{\alpha_0}$. □

Lemma 5.4.5 corresponds to condition (C1) in Theorem 5.3.1.

Lemma 5.4.5. *(Condition (C1)) Assume that the parameters A, b and p_{inc} satisfy the constraints in Lemma 5.4.1. Then there exists a constant $\delta \in (0, 1/10)$ such that if the (μ, λ) selection step of the (μ, λ) self-adaptive EA selects a parent $(x, \chi/n) \in A_{\geq(1,\ell)}$ for $\ell \in [d_j - 1]$, then the offspring $(x', \chi'/n)$ created by self-adapting mutation rate and mutating bitstring satisfies,* $\Pr\left((x', \chi'/n) \in A_{\geq(1,\ell+1)}\right) \geq \frac{1+\delta}{\alpha_0}$, *and if the selected parent $(x, \chi/n) \in A_{(1,d_1)}$, then the offspring $(x', \chi'/n)$ satisfies,* $\Pr\left((x', \chi'/n) \in A_{\geq(2,1)}\right) = \Omega(1/|\varkappa_2|)$.

Proof. If the parent $(x, \chi/n) \in A_{(1,\ell)}$ for $\ell \in [d_j - 1]$, we only consider to produce an offspring $(x', \chi'/n) \in A_{(1,\ell+1)}$ by increasing the mutation rate and not flipping any matched bit, such that $\Pr\left((x', \chi'/n) \in A_{\geq(1,\ell+1)}\right) \geq p_{\text{inc}} (1 - A\chi/n)^{|\varkappa_1|} \geq \frac{1+\delta}{\alpha_0}$ by the definition of levels and Lemma A.2.12 (7).

If the selected parent $(x, \chi/n) \in A_{(1,d_1)}$, we distinguish three cases:

- $|\varkappa_1| = |\varkappa_2| + 1$: $b\theta_2(|\varkappa_1|) < \theta_2(|\varkappa_2|)$ by Lemma A.2.12 (5).
- $|\varkappa_1| = |\varkappa_2| - 1$: $\theta_2(|\varkappa_1|) < \theta_2(|\varkappa_2|)$.
- $|\varkappa_1| = |\varkappa_2|$: $\theta_2(|\varkappa_1|) = \theta_2(|\varkappa_2|)$.

Thus, we only consider to produce an offspring $(x', \chi'/n) \in A_{(2,1)}$ by reducing the mutation rate, and flipping one mismatched bit and not flipping other matched bits, such that $\Pr\left((x', \chi'/n) \in A_{\geq(1,\ell+1)}\right) \geq (1 - p_{\text{inc}}) (1 - b\chi/n)^{|\varkappa_2|-1} \frac{b\chi}{n} = \Omega(1/|\varkappa_2|)$, since $1 - b\chi/n \leq$

$1-\theta_2(|\varkappa_2|) \leq 1-\theta_2(1) \in \Omega(1)$ and $\frac{b\chi}{n} \leq b\theta_1(|\varkappa_1|) = b^2\eta(|\varkappa_1|) \geq b^2\theta_2(|\varkappa_1|)/A \in \Omega(1/|\varkappa_2|)$ by Lemma A.2.12 (3) and (1). □

Now, we use Lemmas 5.4.3-5.4.5 to prove Lemma 5.4.1.

Proof of Lemma 5.4.1. With the assumptions on P_0, we can apply Theorem 5.3.1 to prove Lemma 5.4.1. We first list some values from the DSM$^{\varkappa,m,\varepsilon,k}$ for later use: $\mathcal{T}_2' := kn^{\varepsilon}|\varkappa_2|$ for some constant $k > 0$, $||\varkappa_1| - |\varkappa_2|| \leq 1$, $|\varkappa_1|, |\varkappa_2| \in \Omega(n^a) \cap O(n^\beta)$. We also define some values $\psi_0 := (1-\zeta/2)/\alpha_0$, $\gamma_0 := \zeta/2$, $\xi := \delta^3\varepsilon/34$, $\eta := n^{\varepsilon/2}$. We then know $\delta, b, p_{\text{inc}}, \gamma_0, \psi_0, \zeta, \varepsilon, a, \beta, \xi \in (0,1)$ and $\alpha_0 \geq 4$ are constants. We use the level partition defined in Eq. (5.4)-(5.8).

Condition (C0) implies not too many individuals in the B region, i.e. at most $\psi_0\lambda$ individuals, which is verified by Lemma 5.4.3, such that $\Pr((x',\chi') \in B) < \frac{1-\zeta}{\alpha_0} \leq (1-\delta)\frac{1-\zeta/2}{\alpha_0} \leq (1-\delta)\psi$ for some constant $\delta \leq \frac{\zeta}{2-\zeta}$.

To verify condition (C2), we must estimate the probability of producing an offspring in $A_{\geq(i,\ell)}$ for $i \in \{1,2\}$ and $\ell \in [d_i]$, assuming at least $\gamma\lambda$ individuals in $A_{\geq(i,\ell)}$ for any $\gamma \in (0,\gamma_0]$ and at most $\psi_0\lambda$ individuals in B. To produce such offspring, it is necessary to first select a parent (x,χ) in $A_{\geq(i,\ell)}$, and to create an offspring (x',χ') in $A_{\geq(i,\ell)}$. The probability of selecting a parent $(x,\chi) \in A_{\geq(i,\ell)}$ is at least $\gamma\lambda/\mu = \gamma\alpha_0$ by Lemma 5.4.2. Then the probability to create an offspring $(x',\chi') \in A_{\geq(i,\ell)}$ is at least $(1+\delta)/\alpha_0$ by Lemma 5.4.4. Thus, condition (C2) is satisfied by $\gamma\alpha_0(1+\delta)/\alpha_0 = \gamma(1+\delta)$.

To verify condition (C1), we need to estimate the probability of producing an offspring in a level higher than $A_{\geq(1,\ell)}$ for $\ell \in [d_i]$, if at least $\gamma_0\lambda$ individuals in $A_{\geq(1,\ell)}$. We only consider the case that the parent (x,χ) is selected from $A_{\geq(1,\ell)}$, in which its probability is $\gamma_0\alpha_0$ from Lemma 5.4.2. Then by Lemma 5.4.5, the probability of producing an offspring in $A_{(1,\ell+1)}$ is $(1+\delta)\gamma_0 =: z_{(1,\ell)}$ for all $\ell \in [d_1-1]$, and the probability of producing an offspring in $A_{(2,1)}$

is $z_{(1,d_1)} = \Omega(1/|\varkappa_2|)$ for $\ell = 1$.

Now, we can compute the lower bound of probability of runtime $T \leq \mathcal{T}_2'$. By Theorem 5.3.1,

$$\mathcal{T} = \eta\lambda^{17/\delta^3}\left(\frac{1}{z_{(1,d_1)}} + ((1+\delta)\gamma_0)(d_1 - 1) + d_1\lambda\left(\frac{\ln(\gamma_0\lambda)}{\ln(1+\delta/2)} + 1\right)\right)$$

since $\delta, \gamma_0 \in (0,1)$ are constants and $d_1 \in \log(n)$,

$$= O\left(\eta\lambda^{17/\delta^3}\left(|\varkappa_2| + \lambda\log(n)\log(\log(n))\right)\right)$$

since $\lambda \in \Theta(n^\xi)$, $|\varkappa_2| \in \Omega(n^a)$ and $\xi = \delta^3\varepsilon/34 < 1/34 \leq a$,

$$= O\left(\eta\lambda^{17/\delta^3}|\varkappa_2|\right) = O\left(n^\varepsilon|\varkappa_2|\right)$$

if we let $\eta = \Theta\left(n^{\varepsilon/2}\right)$. Thus there exists $\mathcal{T} \in O\left(n^\varepsilon|\varkappa_2|\right)$ and constant $c > 0$ satisfying $\mathcal{T}_2' = cn^\varepsilon|\varkappa_2| \geq \eta\mathcal{T}$, Such that,

$$\begin{aligned}\Pr\left(T \leq \mathcal{T}_2'\right) &\geq \Pr\left(T \leq \eta\mathcal{T}\right)\\ &> \left(1 - 2\eta\tau e^{-\delta^2\min\{\psi_0,\gamma_0\}\lambda/8}\right)\left(1 - me^{-\eta\rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}}\right)\end{aligned}$$

since $\delta, \psi_0, \gamma_0 \in (0,1)$ are constants, and $\eta = \Theta(n^{\varepsilon/2})$, $\lambda \in \Theta(n^\xi)$,

$$\begin{aligned}&= \left(1 - e^{-\Omega(\lambda)}\right)\left(1 - e^{-\Omega(\eta)}\right)\\ &= 1 - e^{-\Omega(n^\xi)}.\end{aligned}$$

□

Theorem 5.4.1 then presents the efficiency of the (μ, λ) self-adaptive EA in addressing the DSM problem. Apart from population size, other parameters such as A, b, and p_{inc} align with the previous study about unknown-structure optimisation in (Case and Lehre, 2020). This suggests that these parameters may not necessarily require tuning for specific problems.

Utilising a larger population size may be necessary for tracking dynamic optima. In previous studies (Dang et al., 2015, 2017), a similarly large population size of $\Theta(n^\xi)$ was employed to track optima.

Theorem 5.4.1. *Let $\varepsilon, a, \beta \in (0,1)$ and $k > 0$ be some constants satisfying $1/34 \leq a < \beta \leq 1$. Let $\epsilon := b'/n$ for any constant $b' > 0$. Consider a starting target substring $\varkappa \in \{0,1\}^{\ell_1}$ with $\ell_1 \in \Theta(n^a)$ and $m \in \Theta(n^\beta)$. Assume that all individuals in the initial population P_0 in the (μ, λ) self-adaptive EA match $\varkappa$ and have a mutation rate of ϵ. Then, there exists a constant $\xi \in (0,1)$ such that the probability of the (μ, λ) self-adaptive EA with parameters $\lambda, \mu = \Theta(n^\xi)$, $\lambda/\mu = \alpha_0 \geq 4$, $A > 1$, $0 < b < 1/(1 + \sqrt{1/\alpha_0(1-p_{\text{inc}})})$, $(1+\delta)/\alpha_0 < p_{\text{inc}} < 2/5$, and ϵ, where A and b are constants, tracking $DSM^{\varkappa,m,\varepsilon,k}$ is $1-e^{-\Omega(n^\xi)}$.*

Proof of Theorem 5.4.1. By the level definition in Eq. (5.4)-(5.8), we know that the initial population P_0 satisfies the assumption of Lemma 5.4.1. Then by Lemma 5.4.1, the probability of the algorithms successfully tracking in all phases is $\left(1 - e^{-\Omega(n^\xi)}\right)^{4m} = 1 - e^{-\Omega(n^\xi)}$. □

5.5 Static Mutation-based EAs Get Lost on DSM

This section shows that static mutation-based EAs (both Algorithms 2 and 3) get lost in tracking the DSM problem. We first derive Lemma 5.5.1, which provides the upper bound of the probability of the static mutation-based EAs moving from one optimum to the next optimum in the evaluation budget of the DSM problem. This upper bound is with respect to the length of the current target substring and the mutation rate of the static mutation-based EAs. From Lemma 5.5.1, too high or too low mutation rates lead the algorithms to achieve the current optimum in the given evaluation budget with an insufficient probability, i.e., at most, a constant probability. More precisely, too high mutation rates are χ/n where $\chi \in \Omega(n^{1+\varepsilon}/\ell)$ and too low mutation rates are $\chi \in O(n^{1-\varepsilon}/\ell)$. Then Theorem 5.5.1 is proved

via Lemma 5.5.1, which shows that there is no existing mutation rate that tracks the DSM problems with a high probability.

Lemma 5.5.1. *Let $\varkappa_1$ and $\varkappa_2$ be two substrings where $\varkappa_1, \varkappa_2 \in \{0,1\}^\ell$ and $\mathrm{H}(\varkappa_1, \varkappa_2) = 1$ for $\ell \in [n]$. Let $\varepsilon \in (0,1)$, $\gamma_0 \in (0,1]$ and $k > 0$ be arbitrary constants. Assume that all individuals of the initial population P_0 of static mutation-based EAs are matching $\varkappa_1$. Then the probability that static mutation-based EAs using mutation rate χ/n where $\chi \in (0, n/2]$ find a solution matching $\varkappa_2$ in $kn^\varepsilon\ell$ evaluations is $p < 1 - \exp\left(-\chi e^{-\chi\frac{\ell-1}{n}}\frac{kn^\varepsilon\ell}{n}\right)\left(1 - \frac{\chi^2 e^{-2\chi\frac{\ell-1}{n}}}{n}\right)^{\frac{kn^\varepsilon\ell}{n}}$.*

Proof. By the assumption $\mathrm{H}(\varkappa_1, \varkappa_2) = 1$, in any generation τ, if there is no individual in P_τ matching $\varkappa_2$, then any individual in P_τ has at least one mismatched bit to $\varkappa_2$. We optimistically assume that the selection operators (Line 3 of Algorithm 2 and Line 3 of Algorithm 3) always return the closest neighbour of the set of solutions matching $\varkappa_2$, i.e., with one mismatched bit and $\ell - 1$ matched bits. To obtain a solution matching $\varkappa_2$, the mutation operator must flip the mismatched bit of x, and do not flip any mismatched bit in which the probability is $(1-\chi/n)^{\ell-1}(\chi/n)$. Then, such an event happens at least once in $kn^\varepsilon\ell$ evaluations with $p =$

$$\begin{aligned}1 - \left(1 - \left(1-\frac{\chi}{n}\right)^{\ell-1}\frac{\chi}{n}\right)^{kn^\varepsilon\ell} &= 1 - \left(1 - \left(1-\frac{\chi}{n}\right)^{n\cdot\frac{\ell-1}{n}}\frac{\chi}{n}\right)^{n\cdot\frac{kn^\varepsilon\ell}{n}} \\ &\leq 1 - \left(1 - e^{-\chi\frac{\ell-1}{n}}\frac{\chi}{n}\right)^{n\cdot\frac{kn^\varepsilon\ell}{n}}\end{aligned}$$

by $(1+x/n)^n \leq e^x$ and Lemma A.2.5

$$\leq 1 - \exp\left(-\chi e^{-\chi\frac{\ell-1}{n}}\frac{kn^\varepsilon\ell}{n}\right)\left(1 - \frac{\chi^2 e^{-2\chi\frac{\ell-1}{n}}}{n}\right)^{\frac{kn^\varepsilon\ell}{n}}.$$

□

Theorem 5.5.1. *Let $\varepsilon, a, \beta \in (0,1)$ and $k > 0$ be some constants satisfying $1/2 + \varepsilon < a + 2\varepsilon < \beta \leq 1 - \varepsilon$. Let $\varkappa \in \{0,1\}^{\ell_1}$ be some starting target substring where $\ell_1 \in \Theta(n^a)$, and*

$m \in \Theta(n^\beta)$. Then the static mutation-based EAs using any mutation rate $\chi/n \in (0, 1/2]$ and population size $\lambda \in \mathbb{N}$ tracks $DSM^{\varkappa,m,\varepsilon,k}$ with probability $e^{-\Omega(n^\beta)}$.

Proof. We prove Theorem 5.5.1 via Lemma 5.5.1. We consider two cases in tracking dynamic optima of the DSM problem to show that the static mutation-based EAs are not able to achieve the final optimum with high probability.

Let $\ell_2 := \ell_1 + m$, then $\ell_2 \in \Theta(n^\beta)$. Based on the mutation rate, we can categorise the scenarios into two situations:

(1) Assume that all solutions in the current population of the static mutation-based EAs using a mutation rate χ/n satisfying $n/2 \geq \chi \in \Omega(n^{1-\beta+\varepsilon})$ match $\varkappa^{i-1}$ where $i \in [2m+1..3m]$. Then by Lemma 5.5.1, the probability of matching $\varkappa^i$ in $\mathcal{T}_i = kn^\varepsilon\ell_1$ evaluations is at most

$$p < 1 - \exp\left(-\chi e^{-\chi\frac{\ell_2-1}{n}}\frac{kn^\varepsilon\ell_2}{n}\right)\left(1 - \frac{\chi^2 e^{-2\chi\frac{\ell_2-1}{n}}}{n}\right)^{\frac{kn^\varepsilon\ell_2}{n}}$$

by the assumptions, we know that there exists some constants $c', c'' > 0$, such that $c'n^{1-\beta+\varepsilon} \leq \chi \leq n/2$ and $\ell_2 = c''n^\beta$, then $\chi e^{-\chi\frac{\ell_2-1}{n}}\frac{kn^\varepsilon\ell_2}{n} \leq \frac{n}{2}\exp\left(-c'n^{1-\beta+\varepsilon}\frac{c''n^\beta-1}{n}\right)\frac{kc''n^\varepsilon n^\beta}{n} = O\left(\frac{n^{\beta+\varepsilon}}{e^{n^\varepsilon}}\right) \in o(1)$, and $\frac{\chi^2 e^{-2\chi\frac{\ell_2-1}{n}}}{n} \leq \frac{(n/2)^2\exp\left(-c'n^{1-\beta+\varepsilon}\frac{c''n^\beta-1}{n}\right)}{n} = O\left(\frac{n}{e^{n^\varepsilon}}\right) \in o(1)$, then

$$= 1 - e^{-o(1)}\left(1 - o(1)\right)^{\frac{kc''n^\varepsilon n^\beta}{n}}$$

since $\frac{kn^\varepsilon\ell_2}{n} = kc''n^{\beta+\varepsilon-1} \leq 1$ by $\beta \leq 1-\varepsilon$,

$$= 1 - e^{-o(1)}\left(1 - o(1)\right) = o(1) < \xi$$

where $\xi \in (0, 1)$ is some constant.

(2) Now we assume that all solutions in the current population of the static mutation-based EA using a mutation rate χ/n satisfying $\chi \in O(n^{1-a-\varepsilon})$, are matching $\varkappa^{i-1}$ where

$i \in [m]$. Then, by Lemma 5.5.1, the probability of matching $\varkappa^i$ in $\mathcal{T}_i = kn^\varepsilon \ell_2$ evaluations are

$$p < 1 - \exp\left(-\chi e^{-\chi\frac{\ell_1-1}{n}} \frac{kn^\varepsilon \ell_1}{n}\right)\left(1 - \frac{\chi^2 e^{-2\chi\frac{\ell_1-1}{n}}}{n}\right)^{\frac{kn^\varepsilon \ell_1}{n}}$$

by the assumptions, we know that there exist some constants $d', d'' > 0$, such that $d'n^{1-a-\varepsilon} \geq \chi > 0$ and $\ell_1 = d''n^a$, then $\chi e^{-\chi\frac{\ell_1-1}{n}} \frac{kn^\varepsilon \ell_1}{n} \leq d'n^{1-a-\varepsilon} e^0 \frac{kd''n^{a+\varepsilon}}{n} = kd'd''$, and $\frac{\chi^2 e^{-2\chi\frac{\ell_1-1}{n}}}{n} < \frac{(d'')^2 n^{2(1-a-\varepsilon)} e^0}{n} = n^{1-2a-2\varepsilon} = o(1)$ by $a + \varepsilon > 1/2$, then

$$< 1 - e^{-kd'd''}(1 - o(1))^{\frac{kd''n^\varepsilon n^a}{n}} < \xi$$

the last equation holds for some constant $\xi \in (0, 1)$ since $1 - \varepsilon \geq \beta$.

By the assumption $1-a-\varepsilon > 1-\beta+\varepsilon$, we know that $O(n^{1-a-\varepsilon}) \cap \Omega(1-\beta+\varepsilon) = (0, n/2]$, so that all mutation rates in $(0, 1/2]$ lead the static mutation-based EAs to track the optimum $\varkappa^i$ in the evaluation budget $\mathcal{T}_i$ with probability at most ξ for either $i \in [m]$ or $i \in [2m+1..3m]$. Thus, the total probability of tracking every target of $\text{DSM}^{\varkappa,m,\varepsilon,k}$ is at most $\xi^m \in e^{-\Omega(n^\beta)}$. □

5.6 Conclusion

This chapter demonstrates the benefits of self-adaptation in dynamic optimisation. Our analyses show that static mutation-based EAs have a negligible chance of tracking these dynamic optima with changing structure, while the self-adaptive EA can track them. We also provided a level-based theorem with tail bounds to evaluate the performance of the self-adaptive EA on the DSM problem.

Chapter Six

Self-adaptation in Multi-modal Landscapes

Authors: Per Kristian Lehre and Xiaoyu Qin

This chapter is based on the following publication:

Self-adaptation via Multi-objectivisation: A Theoretical Study (Lehre and Qin, 2022b) which is published in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'22).

6.1 Introduction

EAs can be good solvers for multi-modal optimisation problems if they balance exploring new but potential less fit regions of the fitness landscape while also focusing on the regions near the fittest individuals (Črepinšek et al., 2013). In the past decade, several studies in the area of runtime analysis investigated how EAs can cope with local optima. In addition to mechanisms like crossover (Antipov et al., 2022; Dang et al., 2018), stagnation detection (Bambury et al., 2021; Rajabi and Witt, 2021) and adapting population size (Hevia Fajardo and Sudholt, 2021a), a large mutation rate was shown to help some mutation-EA only escaping certain local optima. Non-elitist EAs can "jump" a large Hamming distance. But they can potentially also maintain less fit individuals in the population, allowing the population to cross a fitness valley. They might keep some currently low but potentially high fitness individuals in the population and optimise them "smoothly". As mentioned in Section 2.3.4, a tunable problem class SparseLocalOpt was proposed to describe a kind of fitness landscapes with sparse deceptive regions (local optima) and dense fitness valleys (Dang et al., 2021b). Informally, every search point in a dense set has many neighbours in that set, and every search point in a sparse set has few members in any direction. Dang et al. (2021b) show that EAs with a non-linear selection and a sufficiently high mutation rate, i.e., close to the error threshold, can cope with sparse local optima. Non-linear selection is a type of non-elitist selection, in which the probability of each individual to be selected non-linear with respect to its rank in the population, e.g., tournament and linear ranking selections (Lehre and Yao, 2012). Typically, the fitter individual has a higher probability to be selected, but the worse individual still has some chance to be chosen. From their analysis, non-linear selections and sufficiently high mutation rates can limit the percentage of "sparse" local optimal individuals in the population by choosing a sufficiently high mutation rate. The reason is that the sparse local optimal individuals can have a higher chance to be selected but can only survive a small percentage of such individuals after mutation, while the dense fitness valley

individuals may have less chance of being selected but can have higher chance of surviving mutation. However, the performance of the EA depends critically on choosing the "right" mutation rate, which should be sufficiently high but below the error threshold. Moreover, finding such a mutation rate might be difficult or infeasible for some problem instances with not too sparse local optima.

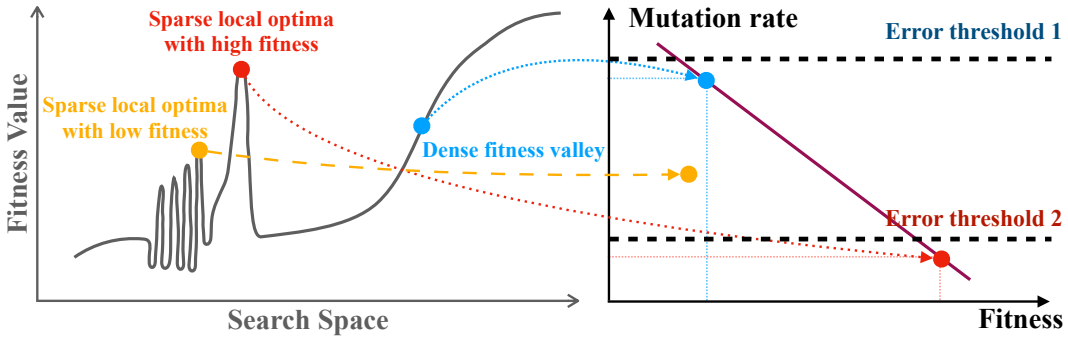

Figure 6.1: Intuition of MOSA-EA

The self-adaptive parameter control mechanism is a promising method to configure parameters. Case and Lehre (2020) also showed that self-adaptation can find the (nearly) maximal "right" mutation rate for each search point on some benchmark functions. We say the "right" mutation rate for a search point is below a threshold, where the expected number of non-worse offsprings of an individual with a mutation rate below this threshold is greater than 1. For SparseLocalOpt, the maximal "right" mutation rate for dense fitness valleys can be higher, while the maximal "right" mutation rate for sparse local optima may be lower. In self-adaptation, individuals can be configured to different mutation rates. If we maximise the mutation rates for both sparse local optima and dense fitness valleys, then the sparse local optimal individual and the dense fitness valley individual could potentially co-exist in a non-dominated Pareto front (Figure 6.1). Therefore, we can treat parameter control from the perspective of multi-objective optimisation to find the optimal solution and the maximal "right" mutation rates for each search point simultaneously.

In this chapter, we propose the *multi-objective self-adaptive EA* (MOSA-EA) for single-objective optimisation, which treats parameter control from the perspective of multi-objective optimisation: The algorithm simultaneously maximises the fitness and the mutation rates. To present advantages of the MOSA-EA, we first propose the PeakedLO$_{m,k}$ function, which is a version of PeakedLO (Dang and Lehre, 2016b) with tunable *sparsity*, where the fitness value is m if a local optimal individual $x = 0^k *^{n-k}$ and is LeadingOnes value otherwise, where $*$ represents an arbitrary bit. The sparsity of the local optima can be tuned by the sparse parameter $k \in [n]$. With the PeakedLO$_{m,k}$ function, we simulate a situation in that algorithms have already been trapped into unknown sparse local optima and estimate the expected time to reach the global optimum. The runtime analyses show that the MOSA-EA can cope with local optima for any $k \in \Omega(n)$, while the elitist EA, the (μ, λ) EA, and the tournament EA fails for some $k \in \Omega(n)$. Table 6.1 demonstrates the theoretical results obtained in this study. The used parameters for the algorithms can be found in the corresponding theorems. Additionally, a comprehensive experimental study on MOSA-EA is presented in Chapter 7.

The chapter is structured as follows. Section 6.2 provides the definition of PeakedLO$_{m,k}$. Section 6.3 introduces the MOSA-EA. Section 6.4 shows that the elitist EA, the (μ, λ) EA and the tournament EA with a fixed mutation rate is inefficient to cope with local optima in PeakedLO$_{m,k}$ for some $k \in \Omega(n)$. In Section 6.5, we analyse the runtime of the MOSA-EA on PeakedLO$_{m,k}$. Section 6.6 summarises the chapter.

Table 6.1: Theoretical results of EAs on PeakedLO$_{m,k}$ (for some constant $c, \delta > 0$)

Algorithm	**PeakedLO$_{m,k}$**	**Runtime T**	**Theorem**
$(\mu + \lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$	Theorem 6.4.1
(μ, λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$	Theorem 6.4.2
2-tour. EA	Any $k < (\ln(3/2) - \delta)\, n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$	Theorem 6.4.3
(μ, λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1 + \ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\, k$ †	$E[T] = O(n^2 \log(n))$	Theorem 6.5.1

6.2 PeakedLO$_{m,k}$ Problems

Dang and Lehre (Dang and Lehre, 2016b) proposed a bi-modal function PeakedLO$_m$ where the fitness value is m if a peak individual $x = 0^n$ and LO(x) otherwise (defined in Eq. (2.21)). The "density" of the fitness valley can be tuned by m. In this section, we introduce a more general version of PeakedLO, where the "sparsity" of the peak individual can be tuned by a parameter k. Let $*$ represent an arbitrary bit. Then the problem class is defined as

$$\textsc{PeakedLO}_{m,k}(x) := \begin{cases} m & \text{if } x = 0^k *^{n-k}, \\ \text{LO}(x) & \text{otherwise} \end{cases}$$

for any $k \in [n]$ and any $1 < m \in \mathbb{R}$. The PeakedLO$_m$ function (Dang and Lehre, 2016b) is a special case of PeakedLO$_{m,k}$ where $k = n$. Similarly to (Dang and Lehre, 2016b), we assume that the algorithm is initiated from the search points $0^k *^{n-k}$. It is unlikely that the algorithm will end up on this particular local optima if the population is initialised uniformly at random. However, here, we are interested in the capability of an algorithm to escape a local optimum, wherever it is encountered. Our assumption is therefore reasonable for the purposes of the analysis.

6.3 Multi-objective Self-adaptive EA

This section will introduce the multi-objective self-adaptive evolutionary algorithm (MOSA-EA). The basic idea of the MOSA-EA is to treat the parameter optimisation task as another objective, i.e., maximising fitness mutation rate simultaneously. Compared to existing self-adaptive EAs, the MOSA-EA is characterised by the population sorting method. Previous self-adaptive EAs (Case and Lehre, 2020; Dang and Lehre, 2016b; B. Doerr et al., 2021) sort the population by considering the fitness of individuals first, then the mutation rate.

[†]For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 6.5.1.

In contrast, the MOSA-EA sorts the population based on the multi-objective sorting partial order (Definition 6.3.1) where we have adapted from two famous multi-objective EAs (Deb et al., 2002; Srinivas and Deb, 1994).

6.3.1 Multi-objective Sorting Partial Order

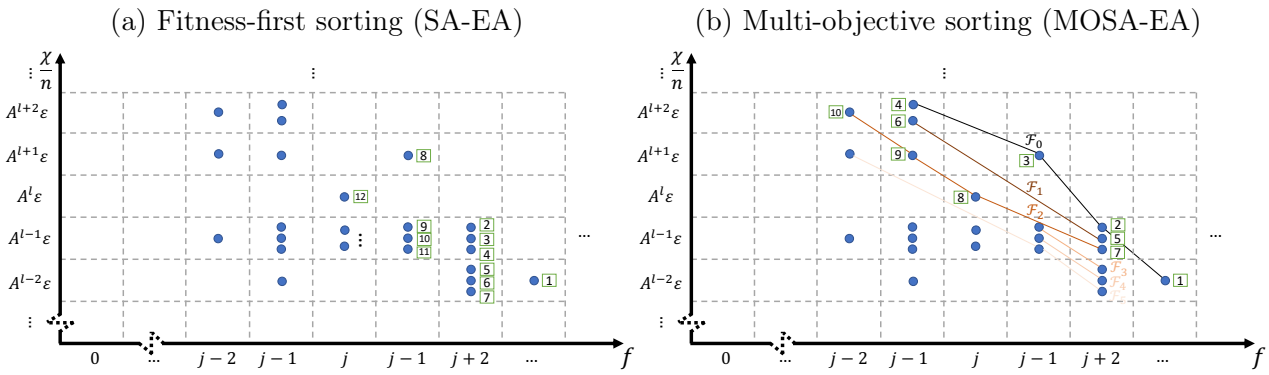

Figure 6.2: Illustration of population sorting in (a) fitness first sorting partial order (Definition 2.2.4 (b)) and (b) multi-objective sorting partial order. The points in the same cell have the same fitness and the same mutation rate.

The MOSA-EA aims to maximise fitness and mutation rates simultaneously. To achieve this goal, we sort the population based on multi-objective sorting partial order before producing the next population, as shown in Definition 6.3.1. Specifically, the multi-objective sorting partial order based on the strict non-dominated fronts of the population produced by the strict non-dominated sorting algorithm (Algorithm 15) based on maximising two objective functions, i.e., the fitness value of the solution $f_1(x, \chi/n) := f(x)$ and the mutation rate of the individual $f_2(x, \chi/n) := \chi/n$. In each strict non-dominated Pareto front, Algorithm 15 guarantees that there do not exist two individuals with the same fitness value and mutation rate. Note that we assume that the aim is to maximise all objectives in Algorithm 15, and we say an individual a dominates another individual b, written as $a \prec b$

if (1) the objective values of a are no lower than b for all functions, and (2) at least one objective value of a is strictly higher than that objective value in b. The main difference between the fast non-dominated sorting algorithm used in NSGA-II (Deb et al., 2002) and the strict non-dominated sorting algorithm shown in Algorithm 15 is that the latter can avoid too many similar or copies of the same individual, i.e., same objective values in all functions, in one front, by Lines 9-11. This characteristic can avoid the situation that some individuals dominate the population, i.e., occupying the top positions of the sorted population. In other words, the strict non-dominated sorting algorithm can increase the diversity of the population.

Subsequently, Definition 6.3.1 outlines the multi-objective sorting partial order, which prefer a smaller index of the front, then a higher fitness value, since the priority of optimisation is to find better solutions.

Definition 6.3.1 (Multi-objective sorting partial order)**.** *Consider a self-adaptive population $P \in \mathcal{Y}^\lambda$ and strict non-dominated fronts of P: $\mathcal{F}_0, \mathcal{F}_1, \ldots$ based on $f_1(x, \chi/n) := f(x)$ and $f_2(x, \chi) := \chi/n$, where $\mathcal{Y} = \{0,1\}^n \times (0, 1/2]$ and $n \in \mathbb{N}$. Let $f : \{0,1\}^n \to \mathbb{R}$ be a Pseudo-Boolean function. Given any two individuals $P(i)$ and $P(j)$ where $i \neq j$ and $i, j \in [\lambda]$, the multi-objective sorting partial order is defined as: for all $(x, \chi/n), (x', \chi'/n) \in \mathcal{Y}$,*

$$P(i) \succeq_{P,f} P(j) \iff g(i) < g(j) \vee (g(i) = g(j) \wedge f_1(P(i)) > f_1(P(j)))\,, \tag{6.1}$$

where $g(k)$ indicates the index of the strict non-dominated front of k-th individual in P.

Then we can get a sorted population based on this sorting partial order, in which any individual a is ranked before an individual b if a has a higher fitness value or a higher mutation rate than b. Figure 6.2 illustrates an example of the order of a population after multi-objective sorting. Note that the points in the same cell have the same fitness value and the same mutation rate. Algorithm 16 shows an alternative way to do multi-objective sorting step in MOSA-EA (step 2 in Algorithm 7).

Algorithm 15 Strict non-dominated sorting

Require: Population sizes $\lambda \in \mathbb{N}$. Population $P \in \mathcal{Z}^{\lambda}$, where $\mathcal{Z}$ is a finite state space. Objective functions $f_1, f_2, \ldots : \mathcal{Z} \to \mathbb{R}$ (assume to maximise all objective functions).

for each individual $P(i)$ **do**
    Set $S_i := \emptyset$ and $n_i := 0$
for $i = 1, \ldots, \lambda$ **do**
    for $j = 1, \ldots, \lambda$ **do**
        if $P(i) \prec P(j)$ based on $f_1, f_2, \ldots$ **then**
            $S_i := S_i \cup \{P(i)\}$,
        else if $P(j) \prec P(i)$ based on $f_1, f_2, \ldots$ **then**
            $n_i := n_i + 1$,
        else if $f_\ell(P(i)) = f_\ell(P(j))$ where $\ell = 1, 2, \ldots$ **then**
            if $P(i) \notin S_j$ **then** $S_i := S_i \cup \{P(i)\}$ **else** $n_i := n_i + 1$.
    if $n_i = 0$ **then** $\mathcal{F}_0 = \mathcal{F}_0 \cup \{P(i)\}$.
Set $k := 0$.
while $\mathcal{F}_k \neq \emptyset$ **do**
    $Q := \emptyset$.
    for each individual $P(i) \in \mathcal{F}_k$ and $P(j) \in S_i$ **do**
        Set $n_j := n_j - 1$.
        if $n_j = 0$ **then** $Q := Q \cup \{P(j)\}$.
    Set $k := k + 1$, $\mathcal{F}_k := Q$.
return $\mathcal{F}_0, \mathcal{F}_1, \ldots$.

6.3.2 Self-adapting Mutation Rate

Self-adapting mutation rate is the key step of self-adaptive EAs. Case and Lehre (2020) and B. Doerr et al.(B. Doerr et al., 2021) use a strategy to increase the mutation rate by

Algorithm 16 Alternative multi-objective sorting step in MOSA-EA

Require: Population sizes $\lambda \in \mathbb{N}$.

Require: Population $P \in \mathcal{Y}^{\lambda}$.

Require: Fitness function f.

1: Sort P_t into $P^1, P^2, \ldots$ where P^1 containing all individuals with the highest fitness f, P^2 containing all individuals with the 2-nd highest fitness f, ...

2: **for** $i = 1, \ldots, \lambda$ **do**

3: Set $\hat{\chi} := -\infty$.

4: **for** $Q = P^1, P^1, \ldots$ **do**

5: Find (x', χ') which is the element with the highest χ in Q.

6: **if** $Q \neq \emptyset$ and $\chi' > \hat{\chi}$ **then**

7: $P(i) := (x', \chi')$ and $\hat{\chi} := \chi'$.

8: Pop (x', χ') from Q.

9: Break.

10: **return** P.

a multiplicative factor $A > 1$ with some probability p_{inc}, decrease otherwise. We apply a similar strategy in Algorithm 17 on the MOSA-EA in experiments (Chapter 7), where adapting mutation rate from ε to nearly $1/2$. To make theoretical analysis easier, we adopt Algorithm 18 on the MOSA-EA in Theorem 6.5.1. The difference is that Algorithm 18 has an additional tiny probability p_{inc2} to increase the mutation rate to the highest of possible mutation rates.

Finally, we say the MOSA-EA is Algorithm 7 using the multi-objective sorting partial order (Definition 6.3.1). In this thesis, we only consider the (μ, λ) MOSA-EA with comma selection (Algorithm 9). For self-adapting mutation rate strategy, we apply Algorithms 17 and 18 in Chapter 7 and Section 6.5, respectively.

Algorithm 17 Self-adapting mutation rate strategy

Require: Parameters $A > 1$, $\varepsilon > 0$ and $p_{\text{inc}} \in (0, 1)$.

Require: Mutation parameter χ/n.

1: $\chi' = \begin{cases} \min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}) \text{ with probability } p_{\text{inc}}, \\ \max(\chi/A, \varepsilon n) \text{ otherwise.} \end{cases}$

2: **return** χ'/n.

Algorithm 18 Self-adapting mutation rate strategy (theoretical)

Require: Parameters $A > 1$, $\varepsilon > 0$, $p_{\text{inc2}} > 0$, $p_{\text{inc}} > 0$ and $p_{\text{inc}} + p_{\text{inc2}} < 1$.

Require: Mutation rate χ/n.

1: $\chi' = \begin{cases} \min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}) \text{ with probability } p_{\text{inc}}, \\ \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor} \text{ with probability } p_{\text{inc2}}, \\ \max(\chi/A, \varepsilon n) \text{ otherwise.} \end{cases}$

2: **return** χ'/n.

6.4 Inefficiency of Fixed Mutation Rate

The $(\mu + \lambda)$ EA and the (μ, λ) EA are proved inefficient when optimising the BBFunnel function which belongs to the SparseLocalOpt problem class (Dang et al., 2021a,b). We use similar proof ideas to prove the inefficiency on PeakedLO$_{m,k}$ for the $(\mu + \lambda)$ EA and (μ, λ) EA, which is shown in Theorems 6.4.1 and 6.4.2, respectively. Furthermore, the tournament EA can solve SparseLocalOpt problems with very "sparse" local optima (Dang et al., 2021a,b). In Theorem 6.4.3, we show the 2-tournament EA with any fixed mutation rate cannot handle a too sparse local optimum on PeakedLO$_{m,k}$.

Theorem 6.4.1. *The expected runtime of the $(\mu + \lambda)$ EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu \geq 1$, initial population $P_0 = \{0^k *^{n-k}\}^\mu$, and mutation parameter $\chi \in O(1)$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ satisfies $\Pr(T \leq e^{cn^d}) \leq e^{-\Omega(n)}$ for some constants $c, d > 0$.*

Proof. For the $(\mu+\lambda)$ EA (Algorithm 2), from initial population $P_0 = \left\{0^k *^{n-k}\right\}^\mu$, if there is no individual with fitness greater than m, then the initial population will be always the μ best individuals. We now prove that with probability $1 - e^{-\Omega(n)}$ during the first e^{cn^d} function evaluations, none of search point $1^{\min\{n,\lceil m\rceil\}} *^{n-\min\{n,\lceil m\rceil\}}$ is created. To created a search point $1^{\min\{n,\lceil m\rceil\}} *^{n-\min\{n,\lceil m\rceil\}}$, it is necessary to mutate from one of individuals in the initial population $P_0 = \{0^k *^{n-k}\}^\mu$, where at least $\min\{k, \min\{n, \lceil m\rceil\}\}$ 0-bits must be flipped. The probability of such an event is $n^{-\Omega(n)}$ for the mutation rate $\frac{\chi}{n} = O\left(\frac{1}{n}\right)$. Thus by a union bound, the probability of no search point $1^{\min\{n,\lceil m\rceil\}} *^{n-\min\{n,\lceil m\rceil\}}$ is created within e^{cn^d} fitness evaluations is $1 - e^{cn^d} n^{-\Omega(n)} = 1 - e^{-\Omega(n)}$. □

Theorem 6.4.2. *For any constant $\delta > 0$, the (μ,λ) EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu = \alpha_0$ where $\alpha_0 > 1$ is a constant, mutation parameter $0 < \chi \notin [\ln(\lambda/\mu) - \delta, \ln(\lambda/\mu) + \delta]$, and initial population $P_0 = \left\{0^k *^{n-k}\right\}^\lambda$ has runtime $\Pr\left(T \leq e^{cn}\right) \leq e^{-\Omega(n)}$ on PEAKEDLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ for some constant $c > 0$.*

Proof. The result for $\chi \geq \ln(\lambda/\mu) + \delta$ follows directly from Theorem 2.2.1 by that P_0 is still at distance $\Omega(n)$ away from 1^n.

When $\chi \leq \ln(\lambda/\mu) - \delta$, it suffices to prove by induction that with high probability, the μ best individuals are still $0^k *^{n-k}$ during the first $e^{c\lambda}$ generations for some constant c sufficiently small. Since the μ best individuals of P_0 are $0^k *^{n-k}$, it is certain that the selected parents to produce offspring in the next generation are $0^k *^{n-k}$. Then, to create an offspring which is $0^k *^{n-k}$, it is suffices to not modify any bit from the parent, the corresponding probability is $\left(1 - \frac{\chi}{n}\right)^k \geq \left(1 - \frac{\chi}{n}\right)^n \geq (1-\sigma)e^{-\chi} \geq \frac{\mu(1-\sigma)e^\delta}{\lambda}$ for any constant $\sigma > 0$ by Lemma A.2.5. Choosing σ so that $\frac{1+\sigma}{1-\sigma} = e^\delta$ then this probability is at least $\frac{\mu}{\lambda}(1+\delta)$. By a Chernoff bound, the probability that the population has less than μ individuals are $0^k *^{n-k}$ in the next generation is $e^{-\Omega(\lambda)} = e^{-\Omega(n)}$. In order for those individuals to be the best of the population, no individual $1^{\min\{n,\lceil m\rceil\}} *^{n-\min\{n,\lceil m\rceil\}}$ must be created, but the probability of such creation

by mutation 0^k*^{n-k} is $n^{-\min(k,\lceil m\rceil)} = n^{-\Omega(n)}$ and still $n^{-\Omega(n)}$ for the λ sampling. By induction and an union bound, with probability $1-e^{-\Omega(n)}$, the μ best individuals are those from 0^k*^{n-k} during the next e^{cn} generations for some sufficiently small constant c.

□

Theorem 6.4.3. *For some constant $\xi \in (0, 2/3)$, any $k \leq an$ where $a = \ln(3(1-\xi)/2)$ is a constant and $m \in \Omega(n)$, the runtime of the 2-tournament EA with population size $\lambda \in \text{poly}(n)$ on PeakedLO$_{m,k}$ with the initial population $P_0 = \{0^k*^{n-k}\}^\lambda$ and any fixed mutation rate satisfies $\Pr(T \leq e^{cn}) = e^{-\Omega(\lambda)}$ for a constant $c > 0$.*

Proof. By the negative drift theorem for populations (Theorem 2.2.1), we know that for any mutation rate $\chi \geq \ln(2) + \delta$ where $\delta \in (0, 1)$ is some constant, the probability that the algorithm optimises PeakedLO$_{m,k}$ within e^{cn} generations is $e^{-\Omega(n)}$ for some constant $c > 0$.

We then prove that for any $k \leq an$, the runtime of the 2-tournament EA on PeakedLO$_{m,k}$ problems any mutation rate $\chi \leq \ln(2) + \delta - \varepsilon$ satisfied $\Pr(T \leq e^{cn}) = e^{-\Omega(\lambda)}$ for a constant $c > 0$.

We will prove a stronger statement that with probability $1 - e^{-\Omega(\lambda)}$, all individuals during the first e^{cn} generations have less than $\min\{n, \lceil m\rceil\}$ leading 1-bits, where $c > 0$ is a constant.

Choose the parameter $\delta' \in (0, 1)$ such that $\ln\left(\frac{1+\delta'}{1-\delta'}\right)/a = \varepsilon \in (0, 1)$. We call an individual is a peak individual if it is 0^k*^{n-k}. We first show by induction that with probability $1-e^{-\Omega(\lambda)}$, there are at least $\frac{\lambda}{2}\left(1 + \frac{\delta'}{2}\right)$ peak individuals in each of the first $e^{c\lambda}$ generations, and we call the run of the algorithm failure otherwise. Then, by Lemma A.2.5, the probability of not mutating any of the first k bits is

$$\begin{aligned}\left(1 - \frac{\chi}{n}\right)^k &\geq e^{-a\chi}(1-\delta') \\ &\geq \left(\frac{1}{2}\right)^a \left(\frac{1}{e^\delta}\right)^a e^{a\varepsilon}(1-\delta') \geq \left(\frac{1}{2}\right)^a \frac{1}{e^\delta}(1+\delta')\end{aligned}$$

let $\xi := 1 - 1/e^{\delta}$, then for all $\delta \in (0,1)$ we have $\xi \in (0, 2/3)$, such that

$$= \left(\frac{1}{2}\right)^{a} (1-\xi)(1+\delta') > \left(\frac{1}{e}\right)^{a} (1-\xi)(1+\delta')$$
$$\geq \frac{2}{3}(1+\delta').$$

Assume that there are $\gamma\lambda \geq \frac{\lambda}{2}$ peak individuals in the current population. A peak individual is produced if a peak individual is selected and none of its first k 0-bits flipped. The probability of this event is

$$\left(\gamma^2 + 2(1-\gamma)\gamma\right)\left(1-\frac{\chi}{n}\right)^{k} \geq \gamma(2-\gamma)\left(1-\frac{\chi}{n}\right)^{an}$$
$$\geq \frac{3}{4}\left(1-\frac{\chi}{n}\right)^{an} \geq \frac{1}{2}(1+\delta').$$

Hence, by a Chernoff bound, the probability that the next generation contains less than $\frac{\lambda}{2}\left(1+\frac{\delta'}{2}\right)$ peak individuals is $e^{-\Omega(\lambda)}$. By induction and an union bound, the probability that the next $e^{c\lambda}$ generations contain less $\frac{\lambda}{2}\left(1+\frac{\delta'}{2}\right)$ peak individuals is still $e^{-\Omega(\lambda)}$, if $c > 0$ is a sufficiently small constant.

We now assume that the run is not a failure. Furthermore, we assume that the algorithm is optimising the function $g(x) := \min\left(\lceil m \rceil, \textsc{PeakedLO}_{m,k}\,(x)\right)$ instead of $\textsc{PeakedLO}_{m,k}$. Clearly, the time to reach at least $\lceil m \rceil$ leading 1-bits is the same, whether the algorithm optimises g or $\textsc{PeakedLO}_{m,k}$. Assume that there are more than $\frac{\lambda}{2}\left(1+\frac{\delta'}{2}\right)$ peak individuals, the reproductive rate of any non-peak individual is always less than

$$\lambda\left(\left(\frac{1}{\lambda}\right)^{2} + \frac{2}{\lambda}\left(1 - \frac{1}{\lambda} - \frac{1+\delta'/2}{2}\right)\right) < 1 - \frac{\delta'}{2} =: \alpha_0$$

For non-peak individuals, the last $n - \min(n, \lceil m \rceil)$ bit-positions are irrelevant when the algorithm optimises g. We can therefore apply the negative drift theorem for populations (Theorem 2.2.1) with respect to the algorithm limited to the first $\min(n, \lceil m \rceil)$ bit-positions only. The mutation operator flips each of the $\min(n, \lceil m \rceil)$ bits independently with probability

$\frac{\chi'}{\min(n,\lceil m \rceil)}$, where $\chi' = \frac{\chi m}{n}$. Hence, we have $e^{-\chi'} < 1 = \frac{1-\delta'/2}{\alpha_0}$, and the conditions of the theorem are satisfied.

□

6.5 Efficiency of MOSA-EA

We now analyse the runtime of the MOSA-EA on PeakedLO$_{m,k}$ (Theorem 6.5.1). In a previous study, Case and Lehre (2020) derived an upper bound on the runtime of the (μ, λ) self-adaptive EA on an unknown structure version of LeadingOnes function via the level-based theorem. They first divide the search space $\mathcal{Y}$ into a two-dimensional level partition including fitness levels and mutation rate sub-levels. Then they define two threshold values $\theta_1(j)$ and $\theta_2(j)$ which is an ideal range of mutation rate to improve the solution for each fitness level. Informally, they count the number of generations to increase the mutation rate to enter this ideal mutation rate range, and the number of generations to improve the solution to the next fitness level if with an ideal mutation rate.

Theorem 6.5.1. *For some constant $\delta \in (0,1)$, the MOSA-EA with $\frac{\lambda}{\mu} = \alpha_0 \geq 4$ and $c\log^2(n) \leq \lambda \in \text{poly}(n)$ where α_0 is a constant and c is a large enough constant, has expected runtime $O\left(n\lambda\log(n)\log\left(\log(n)\right) + n^2\log(n)\right)$ on PeakedLO$_{m,k}$ started with any initial population, if satisfying*

- *$p_{\text{inc}} \in \left(\frac{1+\delta}{\alpha_0}, \frac{2}{5}\right)$, $A > (1 + \sqrt{1/(\alpha_0(1-p_{\text{inc}}))})$ are constants,*
- *$\varepsilon = \frac{b}{n}$, $p_{\text{inc2}} = \frac{d}{n}$ for any small constants $b, d > 0$,*
- *$\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right)/\left(2A\ln\left(\frac{\alpha_0}{1-\delta}\right)\right)k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$, where $k = an$ for any constant $a \in (0,1]$.*

Although the MOSA-EA analysed in Theorem 6.5.1 is significantly different from the (μ, λ) self-adaptive EA, e.g., different sorting mechanisms and different self-adapting mutation rate strategies, we can use a similar proof idea for these two varieties of LeadingOnes. We first divide the search space $\mathcal{Y}$ into regions based on k and m of PeakedLO$_{m,k}$. Then we partition each region into fitness levels and mutation rate sub-levels. We formally define these in Section 6.5.1. Section 6.5.2 defines some threshold values and functions, similaly to (Case and Lehre, 2020), and we also introduce some useful lemmas. Finally, in Section 6.5.3, we apply the level-based theorem (Theorem 2.2.1) to the level partition to get an upper bound of runtime on PeakedLO$_{m,k}$.

6.5.1 Partitioning the Search Space into Levels

We partition the two-dimensional search space $\mathcal{Y} = \mathcal{X} \times [\varepsilon, 1/2]$ into "levels". We first divide the search space $\mathcal{Y}$ into three parts A', A and B_k based on fitness value and mutation rate, which are coloured by yellow, blue and grey in Figure 6.3, respectively. Note that Figure 6.3 is an informal illustration since the formal definitions are complicated. Formally, we define the regions with the PeakedLO$_{m,k}$ parameters k, m and a threshold value θ_2' as

- $A'_j := \{(x, \chi) \mid \mathrm{LO}(x) = j \text{ and } x \neq 0^k *^{n-k} \text{ and } \chi < \theta_2'\}$ for $j \in [0..\lceil m \rceil - 1]$;
- $A'_{\lceil m \rceil} := \{(x, \chi) \mid x = 0^k *^{n-k} \text{ and } \chi < \theta_2'\}$;
- $A_j := \{(x, \chi) \in \mathcal{X} \mid \mathrm{LO}(x) = j \text{ and } x \neq 0^k *^{n-k} \text{ and } \chi \geq \theta_2'\}$ for $j \in [0..\lceil m \rceil - 1]$;
- $A_j := \{(x, \chi) \mid \mathrm{LO}(x) = j\}$, for $j \in [\lceil m \rceil..n]$;
- $B_k := \{(x, \chi) \mid x = 0^k *^{n-k} \text{ and } \chi \geq \theta_2'\}$.

We will define θ_2' in the next subsection. Note that B_k contains no level which is applied in

the level-based theorem, and we will prove that there are not too many individuals in B_k region later.

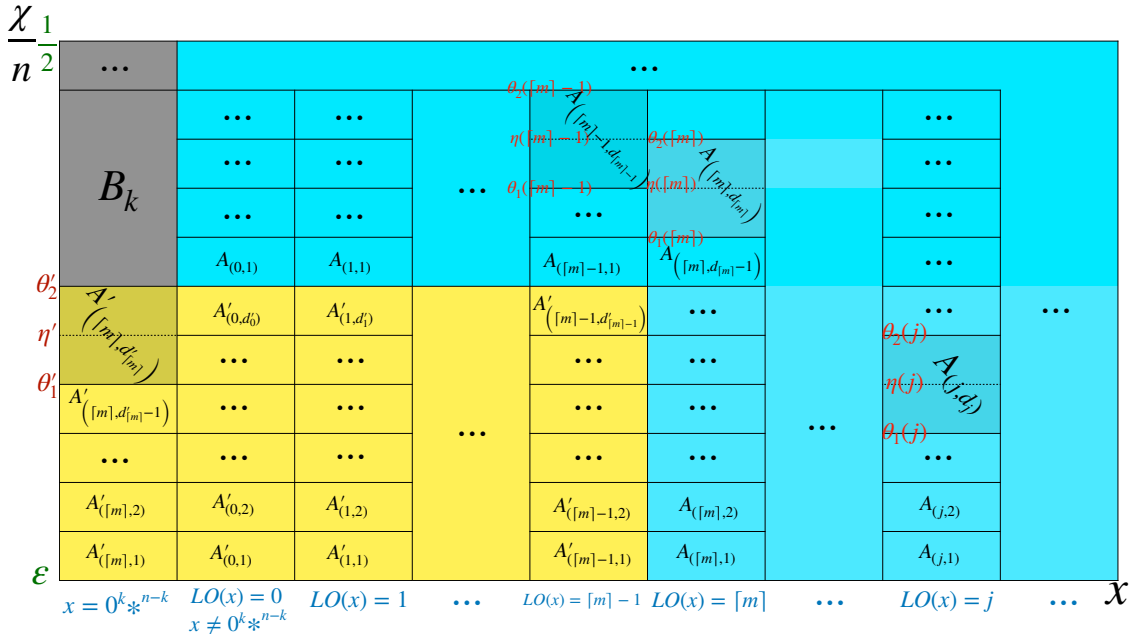

Figure 6.3: Informal illustration of the level partition on PEAKEDLO$_{m,k}$ function (Regions A', A, B_k are coloured by yellow, blue, grey, respectively; x represents a bistring with length n)

To describe the ranking of sub-levels, we define that any level in A is higher than all levels in A', any sub-level in A_j is higher than all sub-levels in A_{j-1}, and any sub-level in A'_j is higher than all sub-levels in A'_{j-1} for $j \geq 1$.

Now, we define sub-levels in regions A' and A. For region A', we first define the *depth* d'_j of levels A'_j:

- For $j \in [0..\lceil m \rceil - 1]$, $d'_j := \min\{\ell \in \mathbb{N} \mid \varepsilon A^\ell \geq \theta'_2\}$;
- For $j = \lceil m \rceil$, $d'_{\lceil m \rceil} := \min\{\ell \in \mathbb{N} \mid \varepsilon A^\ell \geq \theta'_1\}$.

The depth of level in A' implies the number of sub-levels to enter higher region, i.e., A.

Then,

- For $j \in [0..\lceil m \rceil - 1]$, we define the sub-levels for $\ell \in [d'_j]$ as $A'_{(j,\ell)} := A'_j \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^{\ell}, \theta'_2))$,
- For $j = \lceil m \rceil$, we define the *low levels* for $\ell \in [d'_j - 1]$ as $A'_{(\lceil m \rceil,\ell)} := A'_{\lceil m \rceil} \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^{\ell}, \theta'_1))$, and we define the *edge level* as $A'_{(\lceil m \rceil, d'_{-1})} := A'_{\lceil m \rceil} \times [\theta'_1, \min(\frac{1}{2}, \theta'_2)]$.

To define sub-levels in region A_j, we use two threshold values $\theta_1(j)$ and $\theta_2(j)$ which will be defined in the following subsection. Similarly to (Case and Lehre, 2020), $\theta_1(j)$ and $\theta_2(j)$ imply the ideal range of mutation rate to mutate the solution to level A_j. We also begin at defining the *depth* d_j of levels A_j,

- For $j \in [0..\lceil m \rceil - 1]$, $d_j := \min\{\ell \in \mathbb{N} \mid \theta'_2 A^{\ell} \geq \theta_1(j)\}$;
- For $j \in [\lceil m \rceil..n-1]$, $d_j := \min\{\ell \in \mathbb{N} \mid \varepsilon A^{\ell} \geq \theta_1(j)\}$.

The depth of level implies the number of sub-levels to tune the mutation rate from the lowest to an ideal mutation rate.

- For $j \in [0..\lceil m \rceil - 1]$, we define the *low levels* as $A_{(j,\ell)} := A_j \times [\theta'_2 A^{\ell-1}, \min(\theta'_2 A^{\ell}, \theta_1(j)))$, for $\ell \in [d_j - 1]$, and define the *edge levels* as $A_{(j,d_j)} := A_j \times [\theta_1(j), \min(\frac{1}{2}, \theta_2(j))] \cup A_{>j} \times (\min(\frac{1}{2}, \theta_2(j+1)), \min(\frac{1}{2}, \theta_2(j))]$.
- For $j \in [\lceil m \rceil..n-1]$, we define the *low levels* as $A_{(j,\ell)} := A_j \times [\varepsilon A^{\ell-1}, \min(\varepsilon A^{\ell}, \theta_1(j)))$, and we define the *edge levels* as $A_{(j,d_j)} := A_j \times [\theta_1(j), \min(\frac{1}{2}, \theta_2(j))] \cup A_{>j} \times (\min(\frac{1}{2}, \theta_2(j+1)), \min(\frac{1}{2}, \theta_2(j))]$.
- We let A_n contain only one sub-level $A_{(n,1)} := A_j \times [\varepsilon, 1/2]$.

6.5.2 Definitions and Useful Lemmas

We now define the functions θ_1, η and θ_2 for levels A_j where $j \in [0..n-1]$, and the values θ_1', η' and θ_2' for levels $A'_{\lceil m \rceil}$, which use in the levels definitions above.

- For A_j where $j \in [n-1]$, let

$$\eta(j) := \frac{1}{2A}\left(1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{\frac{1}{j}}\right), \theta_1(j) := \frac{\eta(j)}{A}, \theta_2(j) := 1 - q^{\frac{1}{j}}, \text{ where}$$
$$q := \frac{1-\zeta}{\alpha_0}, \; r_0 := \frac{1+\delta}{\alpha_0\left(1 - p_{\text{inc}} - p_{\text{inc2}}\right)}, \; \zeta := 1 - \alpha_0 (r_0)^{1+\sqrt{r_0}}$$

- For A_0, let $\eta(0)$ be any function such that $\eta(0) = \frac{\eta(1)}{A} - o(1)$, and we define $\theta_1(0) := \frac{\eta(0)}{A}$, and $\theta_2 := \frac{\theta_2(1)}{A}$.

- For $A'_{\lceil m \rceil}$, we define $\eta' := \eta(k)$, $\theta_1' := \theta_1(k)$ and $\theta_2' := \theta_2(k)$.

For convenience, let $x' \sim p_{\text{mut}}(x, \chi)$ denote that individual x' is sampled by independently flipping each bit of x with probability χ/n, which is another expression of Line 6 in Algorithm 7. Then, for all individuals (x, χ) in A_j where $j \in [0..n-1]$ and $\chi \in [\varepsilon n, n/2]$, we define the *survival probability* as $r(j, \chi) := \min_{x \in A_j} \Pr_{x' \sim p_{\text{mut}}(x,\chi)}(x' \in A_{\geq j})$, and for all individuals (x, χ) in A'_j, where $j \in [0..\lceil m \rceil]$ and $\chi \in [\varepsilon n, n/2]$, we define the *survival probability* as $r'(j, \chi) := \min_{x \in A'_j} \Pr_{x' \sim p_{\text{mut}}(x,\chi)}(x' \in A'_{\geq j})$.

We then explain that condition $\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right)/\left(2A \ln\left(\frac{\alpha_0}{1-\delta}\right)\right)k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$ in Theorem 6.5.1 essentially implies $\theta_2' < \theta_1(j)$ for all $j \in [0..\lceil m \rceil - 1]$ by Lemma 6.5.1. Informally, it means that the local optima is "sparse" and the fitness valley is "dense".

Lemma 6.5.1. *Assume that the parameters A, α_0 and p_{inc} satisfy the constraints in Theorem 6.5.1. For any constant $\delta \in (0,1)$, if $\lceil m \rceil < \left(\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\right)/\left(2A \ln\left(\frac{\alpha_0}{1-\delta}\right)\right)k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1$, where $k = an$ for any constant $a \in (0,1]$, then $\theta_2' < \theta_1(j)$ for all $j \in [0..\lceil m \rceil - 1]$.*

Proof. By the assumption,

$$\begin{aligned}
\lceil m \rceil &< \frac{\ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)}{2A\ln\left(\frac{\alpha_0}{1-\delta}\right)} k - \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right) + 1 \\
\lceil m \rceil - 1 &< \ln\left(\frac{\alpha_0 p_{\text{inc}}}{1+\delta}\right)\left(\frac{an}{2A\ln\left(\frac{\alpha_0}{1-\delta}\right)} - 1\right) \\
\lceil m \rceil - 1 &< \ln\left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)\left(1 + \frac{1}{-\ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}}\right)
\end{aligned}$$

by Lemma A.2.3,

$$< \ln\left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right) / \ln\left(1 - \ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}\right)$$

we know $\frac{1+\delta}{\alpha_0 p_{\text{inc}}} < 1$ by condition $p_{\text{inc}} > (1+\delta)/\alpha_0$, then

$$\begin{aligned}
\left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{\frac{1}{\lceil m \rceil - 1}} &< \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{-\frac{\ln\left(1-\ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}\right)}{\ln\left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)}} \\
&= e^{-\ln\left(1-\ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}\right)} = 1 - \ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}.
\end{aligned}$$

By the definition, $\theta_1(j)$ is a monotone decreasing function, then for all $j \in [0..\lceil m \rceil - 1]$,

$$\theta_1(j) \geq \theta_1\left(\lceil m \rceil - 1\right) = 1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{\frac{1}{\lceil m \rceil - 1}} > \ln\left(\frac{\alpha_0}{1-\delta}\right)\frac{2A}{an}$$

by Lemma A.2.2,

$$\geq 2A\left(1 - \left(\frac{1-\delta}{\alpha_0}\right)^{\frac{1}{k}}\right) = \theta_2'. \qquad \square$$

Then we introduce three lemmas to support the proofs for conditions (G2) and (G1) of Theorem 2.2.1. Lemma 6.5.2 presents some useful inequalities, and Lemmas 6.5.3 and 6.5.4 can be used to prove conditions (G2) and (G1), respectively. The lemmas and the proofs may look similar to (Case and Lehre, 2020), but they are separate statements since different algorithms and level partitions are considered in Theorem 6.5.1 and (Case and Lehre, 2020).

Lemma 6.5.2. *Assume that the parameters A, α_0, p_{inc}, p_{inc2} and k satisfy the constraints in Theorem 6.5.1. Then there exists a constant $\delta \in (0, 1/10)$ such that for all $j \in [0..n-1]$ and $\chi \in [\varepsilon n, n/2]$,*

(i) $\theta_1(0) < \eta(0) < \varepsilon A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor} \leq 1/2 < \theta_2(0)$,

(ii) $\theta_2(j) = \Omega(1/j)$,

(iii) $\theta_1(j) = O(1/j)$,

(iv) $A\eta(j) \leq \theta_2(j)$,

(v) $\eta(j)/A = \theta_1(j)$,

(vi) $\theta_2(j)/A < \theta_2(j+1)$,

(vii) $A\theta_1(j) \leq \theta_2(j+1)$,

(viii) if $\frac{\chi}{n} \leq \eta(j)$, *then* $r(j, A\chi) \geq \frac{1+\delta}{\alpha_0 p_{\text{inc}}}$,

(ix) if $\frac{\chi}{n} \leq \theta_2(j)$, *then* $r(j, \chi/A) \geq \frac{1+\delta}{\alpha_0(1-p_{\text{inc}}-o(1))}$.

Furthermore,

(x) if $\frac{\chi}{n} \leq \eta'$, *then* $r'(\lceil m \rceil, A\chi) \geq \frac{1+\delta}{\alpha_0 p_{\text{inc}}}$, *and*

(xi) if $\frac{\chi}{n} \leq \theta_2'$, *then* $r'(\lceil m \rceil, \chi/A) \geq \frac{1+\delta}{\alpha_0(1-p_{\text{inc}}-o(1))}$.

Proof. Before proving statements (i)–(ix), we derive bounds on the three constants q, ζ, and r_0. By the assumptions $p_{\text{inc}} < 2/5$ and $\alpha_0 \geq 4$ from Theorem 6.5.1 and $\delta < 1/10$,

$$r_0 < \frac{11}{6\alpha_0} < 1. \tag{6.2}$$

Furthermore, since $r_0 = \frac{1+\delta}{\alpha_0(1-p_{\text{inc}}-o(1))} < 1$ and $\alpha_0 \geq 4$, we have

$$\zeta > 1 - \alpha_0(r_0)^2 > 1 - \frac{1}{\alpha_0}\left(\frac{11}{6}\right)^2 \geq \frac{23}{144}. \tag{6.3}$$

Finally, since $\delta, \zeta, p_{\text{inc}} \in (0,1)$, we have from the definitions of r_0 and q that

$$0 < q < r_0. \tag{6.4}$$

From the definition of the functions θ_1, η, and the constant $\delta \in (0, 1/10)$, it follows that

$$\theta_1(0) < \eta(0) < \eta(1) < \frac{1}{2A}\left(1 - \frac{1}{\alpha_0 p_{\text{inc}}}\right) < \frac{1}{2A}$$
$$\leq \varepsilon A^{\log_A(\frac{1}{2\varepsilon})-1} \leq \varepsilon A^{\lceil \log_A(\frac{1}{2\varepsilon})\rceil} \leq \frac{1}{2}.$$

Also, we have from the definition of q, the constraint $\alpha_0 \geq 4$ from Theorem 6.5.1, and the bound $\zeta > 23/144$ from Eq. (6.3) that

$$\theta_2(0) > \theta_2(1) > 1 - q = 1 - \frac{1-\zeta}{\alpha_0} > 1 - \frac{1 - \frac{23}{144}}{4} = \frac{455}{576}.$$

Thus, we have proven statement (i).

Statement (ii) follows directly from Lemma A.2.7, the definition of θ_2 and the constant q,

$$\theta_2(j) = 1 - q^{1/j} \geq (1-q)/j = \Omega(1/j).$$

For statement (iii), we define $c := \frac{1+\delta}{\alpha_0 p_{\text{inc}}} < 1$, and observe that the inequality $e^x \geq 1 + x$ implies

$$\theta_1(j) < 1 - c^{1/j} = 1 - e^{(1/j)\ln(c)} \leq -(1/j)\ln(c) = O(1/j).$$

For statement (iv), first note that Eq. (6.4) and the assumption $p_{\text{inc}} < 2/5$ imply

$$0 < q < r_0 < \frac{1+\delta}{\alpha_0 p_{\text{inc}}}.$$

For $j \geq 1$, we therefore have $1/j > 0$ and

$$\theta_2(j) = 1 - q^{1/j} \geq 1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{1/j} \geq A\eta(j).$$

For $j = 0$, the definition of $\eta(0)$, statement (iv) for the case $j = 1$ shown above, and the definition of $\theta_2(0)$ gives

$$A\eta(0) < \eta(1) \leq \frac{\theta_2(1)}{A} = \theta_2(0).$$

Statement (v) follows from the definition of $\theta_1(j)$.

We now show statement (vi). The statement is true by definition for $j = 0$, so assume that $j \geq 1$. We first derive an upper bound on the parameter b in terms of the constant q. In particular, the constraint on A from Theorem 6.5.1 and the relationship $0 < q < r_0$ from Eq. (6.4) gives

$$\frac{1}{A} \leq \frac{1}{1+\sqrt{r_0}} < \frac{1}{1+\sqrt{q}} = \frac{1-\sqrt{q}}{1-q}. \tag{6.5}$$

The right hand side of Eq. (6.5) can be further bounded by observing that the function $g(j) := \frac{1-q^{1/(j+1)}}{1-q^{1/j}}$ with $q > 0$ increases monotonically with respect to j. Thus, for all $j \in \mathbb{N}$, we have

$$\frac{1}{A} < \frac{1-\sqrt{q}}{1-q} \leq \frac{1-q^{1/(j+1)}}{1-q^{1/j}}. \tag{6.6}$$

This upper bound on parameter $1/A$ now immediately leads to the desired result

$$\frac{\theta_2(j)}{A} = \frac{(1-q^{1/j})}{A} \leq 1 - q^{1/(j+1)} = \theta_2(j+1). \tag{6.7}$$

Statement (vii) follows by applying the previous three statements in the order (v), (iv), and (vi)

$$A\theta_1(j) = \eta(j) \leq \theta_2(j)/A \leq \theta_2(j+1).$$

Next we prove statement (viii). The statement is trivially true for $j = 0$, because $r(0, A\chi) = 1$, so assume that $j \geq 1$. By the assumption $\chi/n \leq \eta(j)$ and the definition of $\eta(j)$,

$$\begin{aligned} r(j, A\chi) &= \left(1 - \frac{A\chi}{n}\right)^j \geq (1 - A\eta(j))^j \\ &\geq \left(1 - \left(1 - \left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{1/j}\right)\right)^j = \frac{1+\delta}{\alpha_0 p_{\text{inc}}}. \end{aligned} \tag{6.8}$$

Finally, we prove statement (ix). Again, the statement is trivially true for $j = 0$, because $r(0, \chi/A) = 1$, so assume that $j \geq 1$. We derive an alternative upper bound on parameter $1/A$ in terms of r_0 and q. By the constraint on A in Lemma 6.5.2,

$$\begin{aligned}\frac{1}{A} &\leq \frac{1}{1+\sqrt{r_0}} = \frac{\ln(r_0)}{\ln(r_0)+\sqrt{r_0}\ln(r_0)} \\ &= \frac{\ln(r_0)}{\ln\left(r_0 r_0^{\sqrt{r_0}}\right)} = \frac{\ln r_0}{\ln q}. \end{aligned} \tag{6.9}$$

Furthermore, note that the function $h(j) := \frac{1-r_0^{1/j}}{1-q^{1/j}}$ decreases monotonically with respect to j when $r_0 > q > 0$, and has the limit $\lim_{j\to\infty} h(j) = \ln(r_0)/\ln(q)$. Using Eq. (6.9), it therefore holds for all $j \in \mathbb{N}$ that

$$\frac{1}{A} \leq \frac{\ln r_0}{\ln q} \leq \frac{1-r_0^{1/j}}{1-q^{1/j}}. \tag{6.10}$$

The assumption $\chi/n \leq \theta_2(j)$, the definition of $\theta_2(j)$, and (6.10) now give

$$\begin{aligned} r(j, \chi/A) &= \left(1-\frac{\chi}{An}\right)^j \geq \left(1-\frac{\theta_2(j)}{A}\right)^j \\ &= \left(1-\frac{\left(1-q^{1/j}\right)}{A}\right)^j \geq \left(1-\left(1-r_0^{1/j}\right)\right)^j = r_0, \end{aligned}$$

which completes the proof of statement (ix).

For statements (x)-(xi), if the first k 0-bits are not flipped, then the individual survives, by the assumption $\chi/n \leq \eta'$ and the definition of η',

$$\begin{aligned} r'(\lceil m \rceil, A\chi) &= \left(1-\frac{A\chi}{n}\right)^k \geq (1-A\eta')^k = (1-A\eta(k))^k \\ &\geq \left(1-\left(1-\left(\frac{1+\delta}{\alpha_0 p_{\text{inc}}}\right)^{1/k}\right)\right)^k = \frac{1+\delta}{\alpha_0 p_{\text{inc}}}, \end{aligned}$$

and by the assumption $\chi/n \leq \theta_2'$ and the definition of θ_2',

$$\begin{aligned} r'(\lceil m \rceil, \chi/A) &= \left(1-\frac{\chi}{An}\right)^k \geq \left(1-\frac{\theta_2'}{A}\right)^k = \left(1-\frac{\theta_2(k)}{A}\right)^k \\ &= \left(1-\frac{\left(1-q^{1/k}\right)}{A}\right)^j \geq \left(1-\left(1-r_0^{1/k}\right)\right)^k = r_0. \end{aligned}$$

□

Lemma 6.5.3. *Assume that the parameters A, p_{inc}, p_{inc2}, k and m satisfy the constraints in Theorem 6.5.1. Then there exists a constant $\delta \in (0, 1/10)$ such that for all $j \in [0..n-1]$ and $\ell \in [d_j]$, if the (μ, λ) selection of the MOSA-EA selects a parent $(x, \chi/n) \in A_{(j,\ell)}$, then the offspring $(x', \chi'/n)$ created by self-adapting mutation rate and mutating bitstring satisfies* $\Pr((x', \chi'/n) \in A_{\geq(j,\ell)}) \geq \frac{1+\delta}{\alpha_0}$.

Furthermore, there exists a constant $\delta \in (0, 1/10)$ such that for all $j \in [0..\lceil m \rceil]$ and $\ell \in [d'_j]$, if the (μ, λ) selection of the MOSA-EA selects a parent $(x, \chi/n) \in A'_{(j,l)}$, then the offspring $(x', \chi'/n)$ created in by self-adapting mutation rate and mutating bitstring satisfies $\Pr((x', \chi'/n) \in A'_{\geq(j,\ell)}) \geq \frac{1+\delta}{\alpha_0}$.

Proof. We will prove the stronger statement for Eq. (6.5.3) that with probability $(1+\delta)/\alpha_0$, we have simultaneously

$$x' \in A_{\geq j} \quad \text{and} \quad \min\left\{\frac{\chi}{n}, \theta_1(j)\right\} \leq \frac{\chi'}{n} \leq \theta_2(j). \tag{6.11}$$

The event Eq. (6.11) is a subset of the event $(x', \chi'/n) \in A_{(j,\ell)}$, because a lower level $A_{(j,\ell)}$ may contain search points $(x', \chi'/n)$ with mutation rates $\chi'/n < \min(\chi/n, \theta_1(j))$.

By the definition of levels, the parent satisfies $x \in A_{\geq j}$ and $\chi/n \leq \theta_2(j)$. We distinguish between two cases.

Case 1: $\chi/n \leq \eta(j)$. By Lemma 6.5.2 (i), and the monotonicity of η, we have $\eta(j) < 1/2$. Note that in this case, it is still "safe" to increase the mutation rate. For a lower bound, we therefore pessimistically only account for offspring where the mutation parameter is increased from χ to $\min\left(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}\right)$.

Note first that since $A > 1$, we have

$$\frac{\chi'}{n} = \frac{\min\left(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor}\right)}{n} > \frac{\chi}{n}.$$

Also, Lemma 6.5.2 (iv) implies the upper bound

$$\frac{\chi'}{n} \leq \frac{A\chi}{n} \leq A\eta(j) \leq \theta_2(j).$$

To lower bound the probability that $x' \in A_{\geq j}$, we consider the event where the mutation rate is increased, and the event that none of the first j bits in the offspring are mutated with the new mutation parameter $\min\left(A\chi, \varepsilon n A^{\left\lfloor \log_A(\frac{1}{2\varepsilon}) \right\rfloor}\right)$. By definition of Algorithm 18 and using Lemma 6.5.2 (viii), these two events occur with probability at least

$$p_{\text{inc}} r(j, A\chi) \geq (1+\delta)/\alpha_0. \tag{6.12}$$

Case 2: $\eta(j) < \chi/n \leq \theta_2(j)$. Note that in this case, it may be "unsafe" to increase the mutation rate. For a lower bound, we pessimistically only consider mutation events where the mutation parameter is decreased from χ to χ/A. Analogously to above, since $1/A < 1$, we have

$$\frac{\chi'}{n} = \frac{\chi}{An} < \frac{\chi}{n} \leq \theta_2(j).$$

Furthermore, Lemma 6.5.2 (v) implies the lower bound

$$\frac{\chi'}{n} = \frac{\chi}{An} > \frac{\eta(j)}{An} \geq \theta_1(j).$$

To lower bound the probability that $x' \in A_{\geq j}$, we consider the event where the mutation parameter is decreased from χ to χ/A, and the offspring x' is not downgraded to a lower level. By the definition of Algorithm 18, p_{inc2}, $r(j, \chi/A)$, and using Lemma 6.5.2 (ix), these two events occur with probability

$$(1 - p_{\text{inc}} - p_{\text{inc2}}) r(j, \chi/A) \geq (1+\delta)/\alpha_0.$$

Hence, we have shown that in both cases, the event in Eq. (6.11) occurs with probability at least $(1+\delta)/\alpha_0$.

Now, we consider Eq. (6.5.3). By the definition of levels and Lemma 6.5.1, we know that individuals in $A'_{(j,\ell)}$ have mutation rates $\chi/n < \theta'_2 < \theta_1(j) < \eta(j)$ for all $j \in [0..\lceil m \rceil - 1]$ and $\ell \in [d'_j]$. Thus, it is "safe" to increase the mutation rate, such that a lower bound of the probability that $x' \in A_{\geq j}$ is

$$p_{\text{inc}} r'(j, A\chi) \geq (1+\delta)/\alpha_0.$$

Note that for the individual $(x, \chi) \in A'_{(j,d'_j)}$, the offspring $(x', \chi') \in A_{\geq j} \subset A'_{\geq j}$ if the mutation rate χ of the individual is increased from χ to $A\chi$ and none of the first j bits of x are flipped.

For the individuals in $A'_{(\lceil m \rceil,\ell)}$, where $\ell \in [d'_j]$, analogously to the proofs of Eq. (6.5.3), we distinguish two cases: $\chi/n \leq \eta'$ and $\eta' \leq \chi/n < \theta'_2$, where lower bounds of such probabilities that $x' \in A_{\geq j}$ are, by Lemma 6.5.2 (x)-(xi),

$$p_{\text{inc}} r'(\lceil m \rceil, A\chi) \geq (1+\delta)/\alpha_0,$$

and

$$(1 - p_{\text{inc}} - p_{\text{inc2}}) r'(\lceil m \rceil, \chi/A) \geq (1+\delta)/\alpha_0,$$

respectively. Thus, the proof is completed. □

Lemma 6.5.4. *Assume that the parameters A, p_{inc}, p_{inc2}, k and m satisfy the constraints in Theorem 6.5.1. Then for any $j \in [0..n-1]$, and any search point $(x, \chi/n) \in A_{(j,d_j)}$ selected by the (μ, λ) selection of the MOSA-EA applied to PeakedLO$_{m,k}$, the offspring $(x', \chi'/n)$ created by self-adapting mutation rate and mutating bitstring satisfies* $\Pr((x', \chi'/n) \in A_{\geq (j+1,1)}) = \Omega(1/j)$.

Proof. By the definition of level $A_{(j,d_j)}$, we have $\theta_1(j) \leq \chi/n \leq \theta_2(j)$ and so by Lemma 6.5.2 (vi), we have $\chi/(An) \leq \theta_2(j+1)$. Given the definition of levels $A_{(j+1,1)}$, it suffices for a lower bound to only consider the probability of producing an offspring $(x', \chi'/n)$ with lowered mutation rate $\chi'/n = \chi/(An) \leq \theta_2(j+1)$ and fitness $\text{LO}(x') \geq j+1$.

We claim that if the mutation rate is lowered, the offspring has fitness $\mathrm{LO}(x') \geq j+1$ with probability $\Omega(1/j)$. Since the parent belongs to level $A_{(j,d_j)}$, it has fitness $\mathrm{LO}(x) \geq j$, so we need to estimate the probability of not flipping the first j bits, and obtain a 1-bit in position $j+1$.

We now estimate the probability of obtaining a 1-bit in position $j+1$, assuming that the parent x already has a 1-bit in this position, for any $j \in [0..n-1]$. Using that $\theta_2(j)$ decreases monotonically in j, the definition of $\theta_2(0)$, and the lower bound on the parameter $\zeta > 23/144$ from Eq. (6.3), the probability of not mutating bit-position $j+1$ with the lowered mutation rate $\chi/(An)$ is

$$\begin{aligned} 1 - \frac{\chi}{An} &\geq 1 - \theta_2(j)/A \geq 1 - \theta_2(0)/A \\ &= 1 - \theta_2(1) = 1 - \frac{1-\zeta}{\alpha_0} > 1 - \frac{1-\frac{23}{144}}{\alpha_0} = \Omega(1). \end{aligned}$$

If the parent x does not have a 1-bit in position $j+1$, we need to flip this bit-position. By the definition of $\theta_1(j)$, the probability of this event is in the case $j \geq 1$

$$\frac{\chi}{An} \geq \theta_1(j)/A = \frac{1}{2A^3}\left(1 - \left(\frac{1+\delta}{\alpha_0 p_{\mathrm{inc}}}\right)^{1/j}\right) \tag{6.13}$$

$$\geq \frac{1}{2A^3 j}\left(1 - \frac{1+\delta}{\alpha_0 p_{\mathrm{inc}}}\right) = \Omega(1/j), \tag{6.14}$$

where the last inequality follows from Lemma A.2.7. If $j = 0$, we use that $\theta_1(j)$ decreases monotonically in j and Eq. (6.13)–(6.14) to show that the probability of flipping bit $j+1 = 1$ is

$$\frac{\chi}{An} \geq \frac{\theta_1(0)}{A} > \frac{\theta_1(1)}{A} = \Omega(1).$$

The claim that we obtain a 1-bit in position $j+1$ with probability $\Omega(1/j)$ is therefore true.

Thus, the probability of lowering the mutation rate to $b\chi/n$, obtaining a 1-bit in position $j+1$, and not flipping the first j positions is, using the definition of $\theta_2(j)$,

$$(1-p_{\mathrm{inc}})\Omega\left(\frac{1}{j}\right)\left(1-\frac{\chi}{An}\right)^j > \Omega\left(\frac{1}{j}\right)(1-\theta_2(j))^j = \Omega\left(\frac{1}{j}\right)\left(\frac{1-\zeta}{\alpha_0}\right) = \Omega\left(\frac{1}{j}\right),$$

which completes the proof. □

Too many individuals with high fitness but incorrect mutation rate would ruin the progress of the population. We therefore need to prove that there are not too many such "bad" individuals in the population. We first define a "bad" region $B \subset \mathcal{Y}$ containing search points with a mutation rate that is too high, and we say an individual $(x, \chi/n) \in B$ has too high mutation rate. For the constant $\zeta \in (0,1)$, let

$$\begin{aligned} B := \Big\{ & (x,\chi/n) \in A_j \times [\varepsilon, 1/2] \mid \\ & \left(j \in [0..n-1] \ \wedge \ \forall y \in \mathcal{X} \backslash \{0^k *^{n-k}\} \underset{x' \sim p_{\mathrm{mut}}(y,\chi)}{\Pr}(x' \in A_{\geq j}) < \frac{1-\zeta}{\alpha_0} \right) \\ & \vee \left(\forall y = 0^k *^{n-k} \underset{x' \sim p_{\mathrm{mut}}(y,\chi)}{\Pr}(x' = 0^k *^{n-k}) < \frac{1-\zeta}{\alpha_0} \right) \Big\}. \end{aligned} \tag{6.15}$$

By $\theta_2(j)$, θ_2' and B_k, the region B can also be expressed as

$$B = \cup_{j=0}^{n-1} A_{>j} \times (\min(1/2, \theta_2(j+1)), \min(1/2, \theta_2(j))] \cup B_k. \tag{6.16}$$

We show that too many such individuals in the population is rare by Lemma 6.5.5.

Lemma 6.5.5. *Let $B \subset \mathcal{Y}$ be as defined in Eq. (6.15) for a constant $\zeta \in (0,1)$. Then for any generation $t \in \mathbb{N}$ of the algorithm used in Theorem 6.5.1 on $\textsc{PeakedLO}_{m,k}$ with k and m described in Theorem 6.5.1,* $\Pr(|B \cap P_t| \geq \mu(1-\zeta/2)) \leq e^{-\Omega(\mu)}$.

Proof. Consider some parent $(x, \chi/n)$ selected in generation $t-1 \geq 0$ and the (μ, λ) selection. First a new mutation parameter χ' is chosen, a new bit-string x' is obtained from x using bitwise mutation with mutation rate χ'/n. To obtain an upper bound on the probability that $(x', \chi'/n)$ is in B, we proceed in cases based on the outcome of sampling χ', namely, whether $(x', \chi'/n)$ is in B.

Case 1: $(x, \chi'/n) \in B$. It follows immediately from the definition of B that independently

of the chosen parent x, it holds

$$\Pr\left((x', \chi'/n) \in B\right) < \frac{1-\zeta}{\alpha_0}.$$

Case 2: $(x, \chi'/n) \notin B$.

- If $x \neq 0^k *^{n-k}$ then for $(x', \chi'/n) \in B$ to end up in B, by Eq. (6.16), it is necessary that $x' \in A_{\geq u}$ for some $u > j$, where $x \in A_j$ and $r(u, \chi') < (1-\zeta)/\alpha_0$. Since $\chi'/n < 1/2$, the probability of obtaining $x' \in A_{\geq u}$ or $x' = 0^k *^{n-k}$ is no more than

$$\left(1 - \frac{\chi'}{n}\right)^{u-1} \left(\frac{\chi'}{n}\right) < \left(1 - \frac{\chi'}{n}\right)^{u} < \frac{1-\zeta}{\alpha_0}.$$

- If $x = 0^k *^{n-k}$ then for $(x', \chi'/n) \in B$ to end up in B, it is necessary that $x' = 0^k *^{n-k}$, where $r'(\lceil m \rceil, \chi') < (1-\zeta)/\alpha_0$. Then the probability of obtaining $x' \in A_{\geq u}$ or $x' = 0^k *^{n-k}$ is no more than

$$\left(1 - \frac{\chi'}{n}\right)^{k} < \left(1 - \frac{\theta_2'}{n}\right)^{k} < \frac{1-\zeta}{\alpha_0}.$$

Since each of the λ individuals in population P_t are sampled independently and identically, $|B \cap P_t|$ is stochastically dominated by a binomially distributed random variable $Z \sim \text{Bin}\left(\lambda, \frac{1-\zeta}{\alpha_0}\right)$, which has expectation $\mu(1-\zeta)$. By a Chernoff bound,

$$\begin{aligned} \Pr\left(|B \cap P_t| \geq \mu\left(1 - \frac{\zeta}{2}\right)\right) & \qquad (6.17) \\ &\leq \Pr\left(Z \geq \mu\left(1 - \frac{\zeta}{2}\right)\right) \\ &= \Pr\left(Z \geq E[Z]\left(1 + \frac{1}{2(1-\zeta)}\right)\right) \qquad (6.18) \\ &= e^{-\Omega(\mu)}. \end{aligned}$$

□

6.5.3 Applying the Level-based Theorem

Now, we use Lemmas 6.5.2, 6.5.3, 6.5.4 and 6.5.5 to prove Theorem 6.5.1 via Theorem 2.2.1.

Proof (Theorem 6.5.1). We say that a generation t is "failed" if the population P_t contains more than $(1-\zeta/2)\,\mu$ individuals in region B. We will optimistically assume that no generation fails. Under this assumption, we will prove that the conditions of Theorem 2.2.1 hold, leading to an upper bound on the expected number of function evaluations $t_0(n)$ until a search point in $A_{(n,1)}$ is created. In the end we will use a restart argument to account for failed generations.

Each strict non-dominated front has at most $\lceil \log_A(1/2\varepsilon) \rceil = c' \log(n)$ individuals where $c' > 0$ is some constant, so there are at least $\lambda/(c'\log(n)) \in \Omega(\log(n))$ fronts. Let $c' := \left\lceil \log_A\left(\frac{n}{2\varepsilon}\right)\right\rceil / \log(n)$, where $c'\log(n)$ is the maximal number of individuals in a front. Let $\gamma_0 := b'/\log(n)$ for some constant $0 < b' < \zeta/(2\alpha_0 c')$. By the assumption, the number of individuals are not in region B and have chance to be selected is at least $\zeta\mu/2$. Since $\gamma_0\lambda = b'\lambda/\log(n) < \zeta\lambda/(2\alpha_0 c'\log(n)) = \zeta\mu/(2\log(n)) < \zeta\mu/2$, therefore any individual ranked in the first $\gamma_0\lambda$ positions will be selected with probability $\frac{1}{\mu}$ by the (μ,λ) selection.

By the ranking mechanism, if an individual (x,χ) ranked in the first $\gamma\lambda$ positions where $\gamma \in (0,\gamma_0]$, then any individual (x',χ') where either $f(x') > f(x)$ or $\chi' > \chi$ will be ranked in the first $\gamma\lambda$ positions, which guarantees individuals in $A'_{>(j,\ell)}$ are ranked in the first $\gamma\lambda$ positions if there are at least $\gamma\lambda$ individuals in $A'_{\geq(j,\ell)}$, and guarantees individuals in $A_{>(j,\ell)}$ are ranked in the first $\gamma\lambda$ positions if there are at least $\gamma\lambda$ individuals in $A_{\geq(j,\ell)}$.

Now we consider (G1)-(G2) of Theorem 2.2.1 in regions A' and A.

Region A': For $\gamma \in (0,\gamma_0]$, if there are $\gamma\lambda$ individuals in level $A'_{>(j,\ell)}$ for some $j \in [0..\lceil m \rceil]$ and $\ell \in [d'_j]$, then the probability of selecting an individual from $A'_{>(j,\ell)}$ is $\gamma\alpha_0$. Since

$|P_t \cap B| \leq (1-\zeta/2)\mu$, it follows that all $\gamma\lambda < (\zeta/2)\mu$ individuals of $A'_{\geq(j,\ell)}$ are among the μ fittest in the population. Therefore, the probability of selecting an individual from $A'_{>(j,\ell)}$ indeed is $\gamma\lambda/\mu = \gamma\alpha_0$. We assume that the current population has at least $\gamma_0\lambda$ individuals in levels $A'_{\geq(j,\ell)}$. To verify condition (G2), we must estimate the probability of producing an offspring in levels $A'_{>(j,\ell)}$, assuming that there are at least $\gamma\lambda$ individuals in levels $A'_{>(j,\ell)}$, for any $\gamma \in (0, \gamma_0]$. We distinguish five cases:

- Case 1: $j \in [0..\lceil m \rceil - 1]$ and $\ell \in [d'_j - 1]$. Assuming that the parent $(x, \chi/n)$ is in $A'_{(u,v)} \subseteq A'_{\geq(j,\ell+1)}$. Since $\chi/n < \theta'_2 < \theta_1(j) < \eta(j)$ by θ^2, η and Lemma 6.5.1, it is "safe" to increase the mutation rate. For a lower bound, we therefore pessimistically only account for offspring where the mutation parameter is increased to $\min(\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$. The probability of producing an offspring in levels $A'_{\geq(j,\ell+1)}$ is at least $p_{\text{inc}}(1 - A\chi/n)^j > p_{\text{inc}}(1 - A\theta'_2)^j > p_{\text{inc}}(1 - A\eta(j))^j \geq (1+\delta)/\alpha_0$.

- Case 2: $j \in [0..\lceil m \rceil - 1]$ and $\ell = d'_j$. We use a stronger assumption that there are at least $\gamma\lambda$ individuals in $A_{\geq(j,1)}$ for any $\gamma \in (0, \gamma_0]$, which is a subset of $A'_{>(j,d'_j)}$. To produce an offspring in levels $A_{\geq(j,1)}$, it suffices to first select a parent from $A_{\geq(j,1)}$, and secondly create an offspring in levels $A_{\geq(j,1)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. By Lemma 6.5.3, the probability of producing an offspring in levels $A_{\geq(j,1)}$ is at least $(1+\delta)/\alpha_0$.

- Case 3: $j = \lceil m \rceil$ and $\ell \in [d'_j - 2]$. Similarly to Case 1, it is "safe" to increase the mutation rate, since $\chi/n < \eta'$. For a lower bound, we therefore pessimistically only account for offspring where the mutation parameter is increased to $\min(A\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$. The probability of producing an offspring in levels $A'_{\geq(\lceil m \rceil,\ell+1)}$ is at least $1 + \delta/\alpha_0$.

- Case 4: $j = \lceil m \rceil$ and $\ell = d'_j - 1$. By Lemma 6.5.3, the probability of producing an offspring in levels $A'_{\geq(\lceil m \rceil, d'_{\lceil m \rceil})}$ is at least $\frac{1+\delta}{\alpha_0}$.

- Case 5: $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$. We use a stronger assumption that there are at least $\gamma\lambda$ individuals in $A_{\geq(0,1)}$ for any $\gamma \in (0, \gamma_0]$, which is a subset of $A'_{>(\lceil m \rceil, d'_{\lceil m \rceil})}$. The probability of producing an offspring in levels $A_{\geq(0,1)}$ is at least $(1+\delta)/\alpha_0$.

To produce an offspring in levels $A'_{>(j,\ell)}$, it suffices to firstly select a parent $(x, \chi/n)$ from $A'_{>(j,\ell)}$, and then create an offspring $(x', \chi'/n)$ in levels $A'_{>(j,\ell)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. Thus, the probability that the offspring $(x', \chi'/n)$ in levels $A'_{>(j,\ell)}$, is at least $\gamma\alpha_0 \frac{1+\delta}{\alpha_0} = \gamma(1+\delta)$, which (G2) is satisfied.

For condition (G1) of Theorem 2.2.1, we assume that there are $\gamma_0\lambda$ individuals in P_t in $A'_{\geq(j,\ell)}$ where $j \in [0..\lceil m \rceil]$ and $\ell \in [d'_j]$. To verify condition (G1), we assume that the parent $(x, \chi/n)$ is in $A'_{(j,\ell)}$, and we distinguish four cases:

- Case 1: $j \in [0..\lceil m \rceil - 1]$ and $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A'_{\geq(j,\ell+1)}$ is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.

- Case 2: $j \in [0..\lceil m \rceil - 1]$ and $\ell = d'_j$. The probability of the offspring in region A, i.e., $A_{\geq(j,1)}$, is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.

- Case 3: $j = \lceil m \rceil$ and $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A'_{\geq(\lceil m \rceil, \ell+1)}$ is at least $p_{\text{inc}}(1-\chi/n)^{\lceil m \rceil - 1} = \Omega(1)$.

- Case 4: $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$. The probability of the offspring in levels $A_{\geq(0,d_0)}$ is at least $p_{\text{inc2}}(1-(1-1/2)^{\lceil m \rceil - 1}) = \Omega(1/n)$, where the mutation parameter is changed to $\min(\chi, \varepsilon n A^{\lfloor \log_A(\frac{1}{2\varepsilon}) \rfloor})$ and at least one 0-bit of first k bit-position is flipped.

Thus, a lower bound of the probability of producing an offspring in higher levels, i.e., $A'_{>(j,\ell)}$, is $\gamma_0\alpha_0\Omega(1/n) = \Omega(1/n\log(n)) =: z'(\lceil m \rceil, d'_{\lceil m \rceil})$ and $\gamma_0\alpha_0\Omega(1) = \Omega(1/\log(n)) =: z'(j,\ell)$ except the case of $j = \lceil m \rceil$ and $\ell = d'_{\lceil m \rceil}$.

Region A: To verify condition (G2), we distinguish two cases for all $j \in [0..n-1]$ and $\ell \in [d_j]$:

- Case 1: $\ell \in [d_j - 1]$. Firstly, we need to estimate the probability of producing an offspring in levels $A_{\geq(j,\ell+1)}$, assuming that there are at least $\gamma\lambda$ individuals in levels $A_{\geq(j,\ell+1)}$, for any $\gamma \in (0, \gamma_0]$. To produce an offspring in levels $A_{\geq(j,\ell+1)}$, it suffices to first select a parent $(x, \chi/n)$ from $A_{\geq(j,\ell+1)}$, and secondly create an offspring $(x', \chi'/n)$ in levels $A_{\geq(j,\ell+1)}$. The probability of selecting such a parent is at least $\gamma\alpha_0$. Assuming that the parent is in level $A_{(u,v)} \subseteq A_{\geq(j,\ell+1)}$, and applying Lemma 6.5.3 to level $A_{(u,v)}$, the probability that the offspring $(x', \chi'/n)$ is in levels $A_{(u,v)} \subseteq A_{\geq(j,\ell+1)}$ is $(1+\delta)/\alpha_0$ for some $\delta \in (0, 1)$. Thus the probability of selecting a parent in levels $A_{\geq(j,\ell+1)}$, then producing an offspring in levels $A_{\geq(j,\ell+1)}$, is at least $\gamma\alpha_0 (1+\delta) /\alpha_0 = \gamma(1+\delta)$, so condition (G2) is satisfied.

- Case 2: $\ell = d_j$. We assume that there are at least $\gamma\lambda$ individuals in levels $A_{\geq(j+1,1)}$, for $\gamma \in (0, \gamma_0]$. We again apply Lemma 6.5.3 to show the probability of selecting an individual from $A_{\geq(j+1,1)}$ and producing a new individual also in $A_{\geq(j+1,1)}$ is at least $\gamma(1+\delta)$, showing condition (G2) is satisfied.

To verify condition (G1), we assume that there are $\gamma_0\lambda$ individuals in P_t in $A_{\geq(j,\ell)}$ where $j \in [0..n-1]$ and $\ell \in [d_j]$. If the parent $(x, \chi/n)$ is in $A_{\geq(j,\ell)}$, then

- Case 1: $\ell \in [d'_j - 1]$. The probability of the offspring in levels $A_{\geq(j,\ell+1)}$ is at least $(1+\delta)/\alpha_0 = \Omega(1)$.

- Case 2: $\ell = d'_j$. By Lemma 6.5.4, the probability of the offspring in $A_{\geq(j+1,1)}$ is at least $\Omega(1/j)$.

Thus, a lower bound of the probability of producing an offspring in higher levels, i.e., $A_{(j,d_j)}$

is $\gamma_0\alpha_0\Omega(1) = \Omega(1/\log(n)) =: z(j, d_j)$ for $j \in [0..\lceil m \rceil]$, and $\gamma_0\alpha_0\Omega(1/j) = \Omega(1/j\log(n)) =: z(j, \ell)$ for $j \in [0..\lceil m \rceil]$ and $\ell \in [d_j - 1]$.

To verify that $\lambda \geq c\log^2(n)$ is large enough to satisfy condition (G3), we first must calculate the number of total sub-levels m'. The depth of each level j is no more than $d'_j < \lceil \log_A\left(\frac{n}{2\varepsilon}\right)\rceil = O(\log(n))$ for all $j \in [0..\lceil m \rceil]$ and $d_j < \lceil \log_A\left(\frac{n}{2\varepsilon}\right)\rceil = O(\log(n))$ for all $j \in [0..n-1]$. Therefore, by $\gamma_0 = \Omega(1/\log(n))$, $m' = O(n\log(n))$ and $z_{\min} = \Omega(1/n\log(n))$, we know that $\lambda \geq c\log^2(n)$ satisfies condition (G3) for a large enough $c > 1$.

Overall, assuming no failure, the expected time to reach the last level can be calculated by considering regions A' and A separately, which is no more than

$$\begin{aligned}
t_0(n) \leq & \frac{8}{\delta^2}\Bigg(\sum_{j=0}^{\lceil m \rceil - 1}\sum_{\ell=1}^{d'_j}\left(\lambda\log\left(\frac{6\delta\lambda}{4 + z'_{(j,\ell)}\delta\lambda}\right) + \frac{1}{z'_{(j,\ell)}}\right) \\
& + \sum_{\ell=1}^{d'_j - 1}\left(\lambda\log\left(\frac{6\delta\lambda}{4 + z'_{(\lceil m \rceil,\ell)}\delta\lambda}\right) + \frac{1}{z'_{(\lceil m \rceil,\ell)}}\right) \\
& + \lambda\log\left(\frac{6\delta\lambda}{4 + z'_{(\lceil m \rceil,d'_j)}\delta\lambda}\right) + \frac{1}{z'_{(\lceil m \rceil,d'_j)}} \\
& + \sum_{j=0}^{n-1}\sum_{\ell=1}^{d_j - 1} O\left(\lambda\log\left(\frac{1}{z_{(j,\ell)}}\right) + \frac{1}{z_{(j,\ell)}}\right) \\
& + \sum_{j=0}^{n-1} O\left(\lambda\log(\lambda) + \frac{1}{z_{(j,d_j)}}\right)\Bigg) \\
= & O(n\lambda\log(n)\log(\log(n)) + n^2\log(n)).
\end{aligned}$$

Finally, we account for "failed" generations where our assumption that there are less than $(1 - \zeta/2)\mu$ individuals in region B does not hold. We refer to a sequence of $2t_0(n)/\lambda$ generations as a *phase*, and call a phase *good* if for $2t_0(n)/\lambda$ consecutive generations there are fewer than $(1 - \zeta/2)\mu$ individuals in region B. By Lemma 6.5.5 and a union bound, a phase is good with probability $1 - 2t_0(n)/\lambda e^{-\Omega(\mu)} = \Omega(1)$, for $\mu = \Omega(\log(n))$. By Markov's inequality, the probability of reaching a global optimum in a good phase is at least $1/2$.

Hence, the expected number of phases required, each costing $2t_0(n)$ function evaluations, is $O(1)$. □

6.6 Conclusion

In this chapter, we first introduced the MOSA-EA for single-objective optimisation, which treats parameter control from the perspective of multi-objective optimisation: The algorithm simultaneously maximises the fitness and the mutation rates. To achieve this, we proposed the multi-objective sorting partial order mechanism, which sorts the population based on the strict non-dominated fronts. Theoretically, we demonstrated that the MOSA-EA can escape from sparse local optima of $\textsc{PeakedLO}_{m,k}$ for any $k \in \Omega(n)$, where fixed mutation rate EAs may be trapped.

Chapter Seven

Self-adaptation on Complex Combinatorial Optimisation Problems

Authors: Xiaoyu Qin and Per Kristian Lehre

This chapter is based on the following publication:

Self-adaptation via Multi-objectivisation: An Empirical Study (Qin and Lehre, 2022) which is published in Parallel Problem Solving from Nature XVII (PPSN'22).

7.1 Introduction

In Chapter 6, a new self-adaptive EA, the multi-objective self-adaptive EA (MOSA-EA), was proposed to optimise single-objective functions, which treats parameter control from multi-objectivisation. The algorithm maximises the fitness and the mutation rates simultaneously, allowing individuals in dense fitness valleys and on sparse local optima to co-exist on a non-dominated Pareto front. The previous study showed its efficiency in escaping a local optimum with unknown sparsity, where some fixed mutation rate EAs including non-linear selection EAs become trapped. However, it is unclear whether the benefit of the MOSA-EA can also be observed for more complex problems, such as NP-hard combinatorial optimisation problems and noisy fitness functions.

This chapter continues the study of MOSA-EA through an empirical study of its performance on selected combinatorial optimisation problems. We find that the MOSA-EA not only has a comparable performance on unimodal functions, e.g., LeadingOnes and OneMax, but also outperforms eleven randomised search heuristics considered on a bi-modal function with a sparse local optimum, i.e., Funnel (Dang et al., 2021a). For NP-hard combinatorial optimisation problems, the MOSA-EA increasingly outperforms other algorithms for harder NK-Landscape and Max-k-Sat instances. In particular, the MOSA-EA outperforms a problem-specific MaxSat solver on some hard Max-k-Sat instances.

7.2 Parameter Settings in MOSA-EA

One of the aims of self-adaptation is to reduce the number of parameters that must be set by the user. MOSA-EA has three parameters ε, p_{inc} and A, in addition to the population sizes λ and μ (As mentioned in Chapter 6, the MOSA-EA applies Algorithm 17 in all experiments

conducted in this chapter). We will first investigate how sensitive the algorithm is to these parameters. Adding three new parameters to adapt one parameter seems contradictory to the aim of self-adaptation. However, we will show later that these parameters need not to be tuned carefully. We use the same parameter setting of the MOSA-EA for all experiments in this chapter to show that the MOSA-EA does not require problem-specific tuning of the parameters.

The parameter ε is the lower bound of the mutation rate in the MOSA-EA. In fixed mutation rate EAs, we usually set a constant mutation parameter χ. To cover the range of all possible mutation rates χ/n, we recommend to set the lowest mutation rate $\varepsilon = c/(n\ln(n))$, where c is some small constant. In this chapter, we set $\varepsilon = 1/(2n\ln(n))$. As mentioned before, $A > 0$ and $p_{\text{inc}} \in (0,1)$ are two self-adapting mutation rate parameters in Algorithm 17. We use simple functions as a starting point to empirically analyse the effect of setting the parameters of A and p_{inc}. We run the MOSA-EA with different parameters A and p_{inc} on OneMax, LeadingOnes and Funnel (Definition A.1.2) with problem instance size $n = 100$ which represent single-modal and multi-modal functions. For each pair of A and p_{inc}, we execute the algorithm 100 times, with population sizes $\lambda = 10^4 \ln(n)$ and $\mu = \lambda/8$. Figures 7.1 (a), (b) and (c) show the medians of the runtimes of the MOSA-EA for different parameters A and p_{inc} on OneMax, LeadingOnes and Funnel, respectively. The maximal number of fitness evaluations is 10^9.

From Figures 7.1, the algorithm finds the optimum within 10^7 function evaluations for an extensive range of parameter settings. The algorithm is slow when A and p_{inc} are too large. Therefore, we recommend to set $p_{\text{inc}} \in (0.3, 0.5)$ and $A \in (1.01, 1.5)$. For the remainder of the chapter, we will choose $p_{\text{inc}} = 0.4$ and $A = 1.01$. We also recommend to use a sufficiently large population size $\lambda = c\ln(n)$ for some large constant c. We will state λ and μ later.

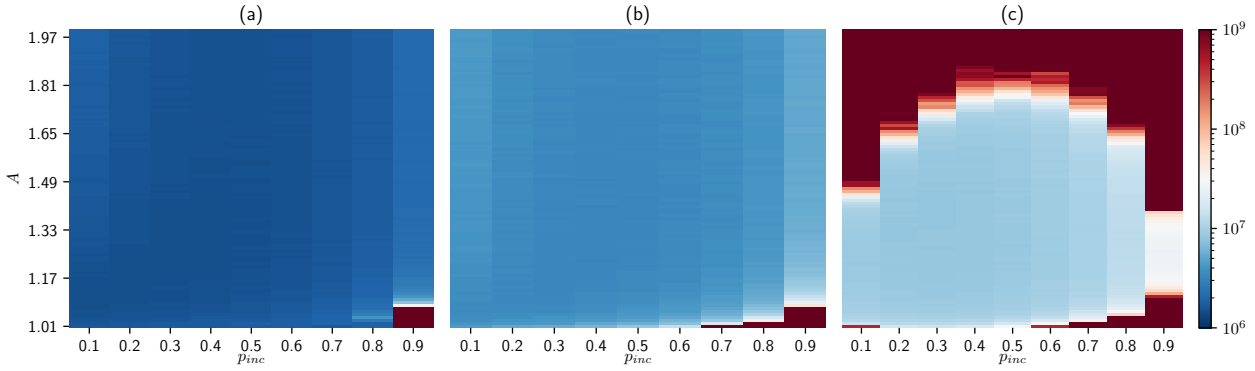

Figure 7.1: Median runtimes of MOSA-EA for given A and p_{inc} on (a) OneMax, (b) LeadingOnes and (c) Funnel over 100 independent runs ($n = 100$).

7.3 Experimental Settings and Methodology

We compare the performance of the MOSA-EA with eleven other heuristic algorithms on three classical pseudo-Boolean functions and two more complex combinatorial optimisation problems. We also empirically study the MOSA-EA in noisy environments. In this section, we will first introduce the other algorithms and their parameter settings. We will then describe the definitions of benchmarking functions and problems. We will also indicate the statistical approach applied in the experiments.

7.3.1 Parameter Settings in Other Algorithms

We consider eleven other heuristic algorithms, including three single-individual elitist algorithms, *random search* (RS), *random local search* (RLS) and (1+1) EA, two population-based elitist algorithms, $(1 + (\lambda, \lambda))$ GA (B. Doerr et al., 2015) and FastGA(B. Doerr et al., 2017), two *estimation of distribution algorithms* (EDAs), cGA(Harik et al., 1999) and UMDA(Mühlenbein and Paaß, 1996), two non-elitist EAs, 3-tournament EA and (μ, λ) EA, and two self-adjusting EAs, (μ, λ) self-adaptive EA (Case and Lehre, 2020) and self-adjusting

population size $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA (we call it SA-$(1, \lambda)$ EA in short)(Hevia Fajardo and Sudholt, 2021a), and a problem-specific algorithm, Open-WBO(Martins et al., 2014). These heuristic algorithms are proved to be efficient in many scenarios, e.g., in multi-modal and noisy optimisation (Antipov et al., 2022; Buzdalov and Doerr, 2017; Case and Lehre, 2020; Dang et al., 2021a,b; B. Doerr et al., 2017; Droste, 2004; Friedrich et al., 2016; Hevia Fajardo and Sudholt, 2021b; Lehre and P. T. H. Nguyen, 2021). Open-WBO is a MaxSat solver that operates differently than randomised search heuristics. It was one of the best MaxSat solvers in *MaxSAT Evaluations* 2014, 2015, 2016 and 2017 (Ansótegui et al., 2017).

It is essential to set proper parameters for each algorithm for a comparative study (Bartz-Beielstein et al., 2020). In the experiments, we use parameter recommendations from the existing theoretical and empirical studies, which are summarised in Table 7.1.

Note that to investigate the effect of self-adaptation via multi-objectivisation, the (μ, λ) self-adaptive EA applied the same self-adapting mutation rate strategy and initialisation method as the MOSA-EA, instead of the strategy used in (Case and Lehre, 2020). The only difference between the (μ, λ) self-adaptive EA and the MOSA-EA in the experiments is the sorting mechanism: The (μ, λ) self-adaptive EA uses the fitness-first sorting mechanism (Case and Lehre, 2020) and the MOSA-EA uses the multi-objective sorting mechanism.

7.3.2 Theoretical Benchmarking Functions

We first consider two well-known unimodal functions, OneMax and LeadingOnes. One would not expect to encounter these functions in real-world optimisation. However, they serve as a good starting point to analyse the algorithms. We cannot expect good performance from an algorithm which performs poorly on these simple functions. We also consider Funnel which was proposed by Dang et al. (2021a), It is a bi-modal function

Table 7.1: Parameter settings of algorithms considered in this chapter

Category	**Algorithm**	**Parameter Settings**
Elitist EAs	RS	-
	RLS	-
	(1+1) EA	Mutation rate $\chi/n = 1/n$
	$(1 + (\lambda, \lambda))$ GA (B. Doerr et al., 2015)	Mutation rate $p = \lambda/n$; Crossover bias $c = 1/\lambda$; Population size $\lambda = 2\ln(n)$(Buzdalov and Doerr, 2017)
	FastGA (B. Doerr et al., 2017)	$\beta = 1.5$ (B. Doerr et al., 2017)
EDAs	cGA (Harik et al., 1999)	$K = 7\sqrt{n}\ln(n)$ (Sudholt and Witt, 2016)
	UMDA (Mühlenbein and Paaß, 1996)	$\mu = \lambda/8$
Non-Elitist EAs	3-tournament EA	Mutation rate $\chi/n = 1.09812/n$ (Dang et al., 2021a,b)
	(μ, λ) EA	Mutation rate $\chi/n = 2.07/n$; Population size $\mu = \lambda/8$ (Lehre, 2010; Lehre and Yao, 2012)
Self-adjusting EAs	SA-$(1, \lambda)$ EA (Hevia Fajardo and Sudholt, 2021a)	Population size $\lambda_{\text{init}} = 1$, $\lambda_{\max} = enF^{1/s}$; $F = 1.5$, $s = 1$ (Hevia Fajardo and Sudholt, 2021a)
	SA-EA	Population size $\mu = \lambda/8$; $A = 1.01$, $p_{\text{inc}} = 0.4$, $\varepsilon = 1/(2\ln(n))$
	MOSA-EA	Population size $\mu = \lambda/8$; $A = 1.01$, $p_{\text{inc}} = 0.4$, $\varepsilon = 1/(2\ln(n))$
MaxSat solver	Open-WBO (Martins et al., 2014)	Default (We use version 2.1: https://github.com/sat-group/open-wbo)

with sparse local optima and a dense fitness valley which belongs to the problem class SparseLocalOpt$_{\alpha,\varepsilon}$ (Dang et al., 2021b). The parameters u, v, w in the Funnel function describe the sparsity of the deceptive region and the density of the fitness valley. Dang et al. (2021a) proved that the $(\mu+\lambda)$ EA and the (μ,λ) EA are inefficient on Funnel if $v-u=\Omega(n)$ and $w-v=\Omega(n)$, while the 3-tournament EA with the mutation rate $\chi/n=1.09812/n$ can find the optimum in polynomial runtime. We consider Funnel with the parameters $u=0.5n$ $v=0.6n$ and $w=0.7n$ which satisfy the restrictions above.

For each problem, we independently run each algorithm 30 times for each problem size $n\in\{100,110,\ldots,200\}$. For fair comparison, we set sufficiently large population sizes $\lambda=10^4\ln(n)$ for the MOSA-EA, the 3-tournament EA, the (μ,λ) EA, the UMDA and the (μ,λ) self-adaptive EA.

7.3.3 Complex Combinatorial Optimisation Problems

We consider two NP-hard problems, the random NK-Landscape problem and the random Max-k-Sat problem, which feature many local optima (Kauffman and Weinberger, 1989; Ochoa and Chicano, 2019). We compare the performance of the MOSA-EA with other popular randomised search heuristics in a fitness evaluation budget. We also compare MOSA-EA with the MaxSat solver Open-WBO (Martins et al., 2014) for a fixed CPU budget.

We use the same population size $\lambda=20000$ for the MOSA-EA, the 3-tournament EA, the (μ,λ) EA, the UMDA and the (μ,λ) self-adaptive EA. The MOSA-EA are compared with the other algorithms using Wilcoxon rank-sum tests (Wilcoxon, 1992).

Random NK-Landscape problems We generate 100 random NK-Landscape instances with $n=100$ for each $k\in\{5,10,15,20,25\}$ by uniformly sampling values between 0 and 1 in the lookup table. We run each algorithm once on each instance and record the

highest fitness value achieved in the fitness evaluation budget of 10^8.

Random Max-k-Sat problems Similarly with the NK-Landscape experiments, we run each algorithm on these random instances and record the smallest number of unsatisfied clauses during runs of 10^8 fitness function evaluations. Additionally, we run Open-WBO and the MOSA-EA on the same machine in one hour CPU time. The MOSA-EA is implemented in OCaml, while OpenWBO is implemented in C++, which generally leads to faster-compiled code than OCaml.

7.4 Results and Discussion

7.4.1 Theoretical Benchmarking Functions

Figures 7.2 (a), (b) and (c) show the runtimes of the MOSA-EA and nine other heuristic algorithms on OneMax, LeadingOnes and Funnel over 30 independent runs, respectively. Based on theoretical results (He and Yao, 2004), the expected runtimes of the (1+1) EA are $O(n\log(n))$ and $O(n^2)$ on OneMax and LeadingOnes, respectively. We thus normalise the y-axis of Figures 7.2 (a) and (b) by $n\ln(n)$ and n^2, respectively. We also use the log-scaled y-axis for Figures 7.2 (a) and 7.2 (b). The runtime of the 3-tournament EA with a mutation rate $\chi/n = 1.09812/n$ and a population size $c\log(n)$ for a sufficiently large constant c on Funnel is $O(n^2\log(n))$ (Dang et al., 2021a). We thus normalise the y-axis of Figure 7.2 (c) by $n^2\ln(n)$. Note that (1+1) EA, RLS, (μ,λ) EA, cGA, FastGA, $(1+(\lambda,\lambda))$ GA, (μ,λ) self-adaptive EA and SA-$(1,\lambda)$ EA cannot achieve the optimum of the Funnel function in 10^9 fitness evaluations. It is known that non-elitist black-box algorithm can optimise Funnel in polynomial time with high probability (Dang et al., 2021a,b).

Although the MOSA-EA is slower than EDAs and elitist EAs on the unimodal functions

OneMax and LeadingOnes, it has comparable performance with the other non-elitist EAs and the (μ, λ) self-adaptive EA. Recall theoretical results on Funnel (Dang et al., 2021a,b), elitist EAs and the (μ, λ) EA fail to find the optimum, while the 3-tournament EA is efficient. The results in Figure 7.2 (c) are consistent with the theoretical results. In this chapter, the (μ, λ) EA, the (μ, λ) self-adaptive EA and the MOSA-EA use the (μ, λ) selection. Compared with the (μ, λ) EA and the (μ, λ) self-adaptive EA, self-adaptation via multi-objectivisation can cope with sparse local optima and even achieve a better performance than the 3-tournament EA.

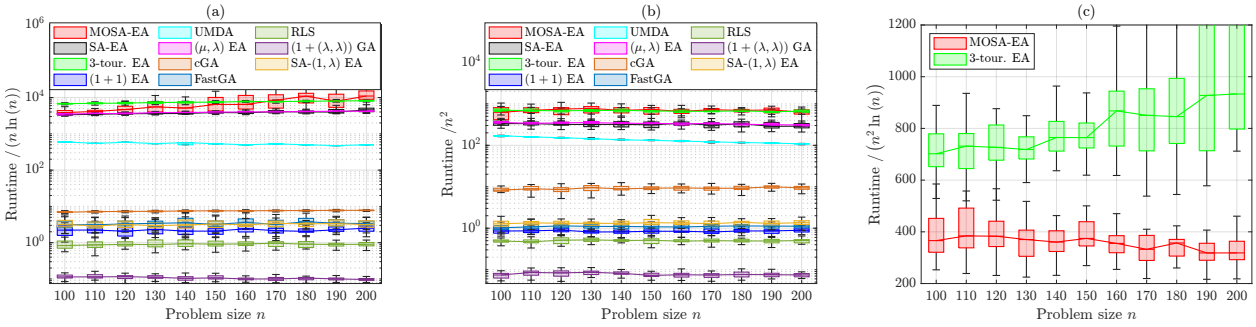

Figure 7.2: Runtimes of nine algorithms on the (a) OneMax, (b) LeadingOnes, (c) Funnel ($u = 0.5n, v = 0.6n, w = 0.7n$) functions over 30 independent runs. The y-axis in sub-figures (a) and (b) are log-scaled. (1+1) EA, RLS, (μ, λ) EA, cGA, FastGA, $(1 + (\lambda, \lambda))$ GA, (μ, λ) self-adaptive EA and SA-$(1, \lambda)$ EA cannot find the optimum of the Funnel function in 10^9 fitness evaluations.

7.4.2 Complex Combinatorial Optimisation Problems

Random NK-Landscape Problems Figure 7.3 illustrates the experimental results of eleven algorithms on random NK-Landscape problems. From Wilcoxon rank-sum tests, the highest fitness values achieved by the MOSA-EA are statistically significantly higher than all other algorithms with significance level $\alpha = 0.05$ for all NK-Landscape with $k \in \{10, 15, 20, 25\}$. Furthermore, the advantage of the MOSA-EA is more significant for

the harder problem instances.

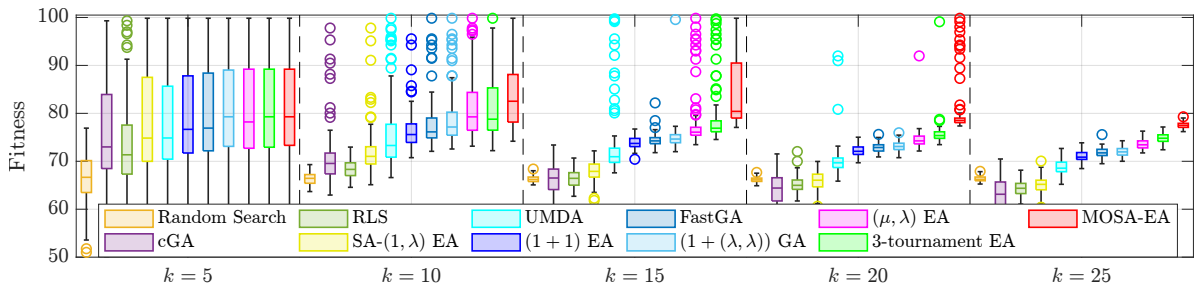

Figure 7.3: The highest fitness values found in the end of runs in 10^8 fitness evaluations on 100 random NK-Landscape instances with different k ($n = 100$).

Figure 7.4 illustrates the highest fitness values found during the optimisation process on one random NK-Landscape instance ($k = 20$, $n = 100$). Note that the non-elitist algorithms, i.e., EDAs, (μ, λ) EA, 3-tournament EA, SA-$(1, \lambda)$ EA and MOSA-EA, do not always keep the best solution found. Therefore, the corresponding lines might fluctuate. In contrast, the elitist EAs, e.g., (1+1) EA, increase the fitness value monotonically during the whole run.

The elitist EAs converge quickly to solutions of medium quality, then stagnate. In contrast, the 3-tournament EAs, the (μ, λ) EA and the MOSA-EA improve the solution steadily. Most noticeably, the MOSA-EA improves the solution even after 10^7 fitness evaluations.

Random Max-k-Sat Problems Figure 7.5 illustrates the medians of the smallest number of unsatisfied clauses found in the 10^8 fitness evaluations budget among eleven algorithms on 100 random Max-k-Sat instances ($k = 5$, $n = 100$) with different total number of clauses m. Coja-Oghlan (2014) proved that the probability of generating a satisfiable instance drops from nearly 1 to nearly 0, if the ratio of the number of clauses m and the problem size n is greater than a threshold, $r_{k-\text{Sat}} = 2^k \ln(2) - \frac{1}{2}(1 + \ln(2)) + o_k(1)$, where $o_k(1)$ signifies a term that tends to 0 in the limit of large k. In this case, $r_{k-\text{Sat}}$ is roughly 2133 if we ignore the $o_k(1)$ term. We therefore call an instance with $m \geq 2133$ hard. The MOSA-EA is

Table 7.2: Statistical results of experiments on random NK-LANDSCAPE problems. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and MOSA-EA.

k	Stat.	**RS**	**cGA**	**UMDA**	**RLS**	**SA-**$(1,\lambda)$**EA**	$(1+1)$**EA**	**FastGA**	$(1+(\lambda,\lambda))$**GA**	(μ,λ)**EA**	**3-tour.EA**	**MOSA-EA**
5	Median	66.6591	72.9964	74.8631	71.3547	74.8418	76.6613	76.9230	79.2846	78.2089	79.2846	**79.2846**
	p-value	2.1e-22	2.3e-04	0.0213	6.5e-08	0.0226	0.2668	0.4215	0.9299	0.7985	0.8805	-
10	Median	66.4442	69.5499	73.2968	68.3100	71.0248	75.5792	76.1340	77.1520	79.2680	78.7832	**82.5270**
	p-value	2.6e-34	1.5e-26	2.0e-15	2.6e-34	3.5e-34	5.0e-18	1.1e-12	2.2e-09	0.0030	0.0063	-
15	Median	66.2055	66.5517	70.9576	66.4446	67.8968	73.7253	74.2253	74.6407	76.0777	76.9053	**80.4417**
	p-value	2.6e-34	2.6e-34	5.5e-22	2.6e-34	2.6e-34	2.6e-34	1.8e-33	5.2e-33	1.3e-20	1.1e-17	-
20	Median	66.1233	64.4191	69.6786	64.9865	66.0533	72.8025	72.8783	73.0882	74.2580	75.3662	**78.5247**
	p-value	2.6e-34	2.6e-34	7.0e-31	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	4.0e-33	1.2e-31	-
25	Median	66.2207	63.1222	68.5683	64.3685	65.1886	70.8648	71.7564	71.9623	73.4398	74.8115	**77.5024**
	p-value	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	2.6e-34	1.4e-33	-

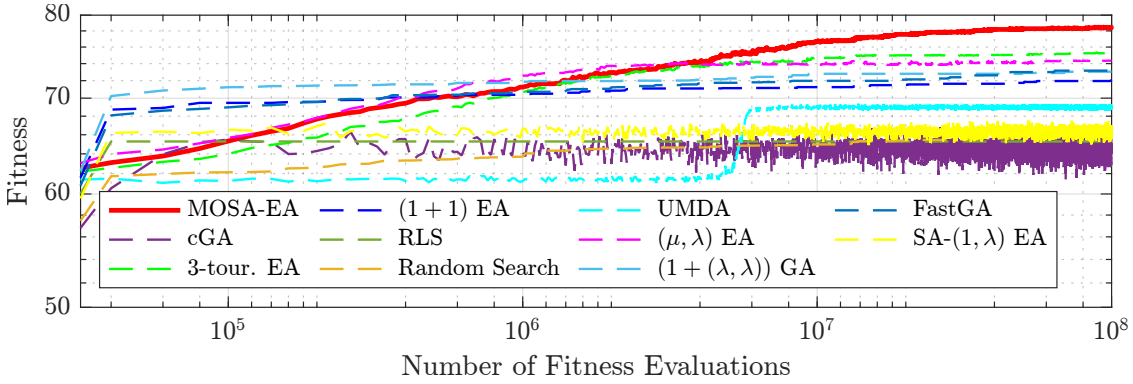

Figure 7.4: The median of the highest fitness values found in every 2×10^4 fitness evaluations over 30 independent runs on one random NK-Landscape instance ($k = 20$, $n = 100$). The x-axis is log-scaled.

statistically significantly better than the other ten algorithms with significance level $\alpha = 0.05$ on hard instances from Wilcoxon rank-sum tests.

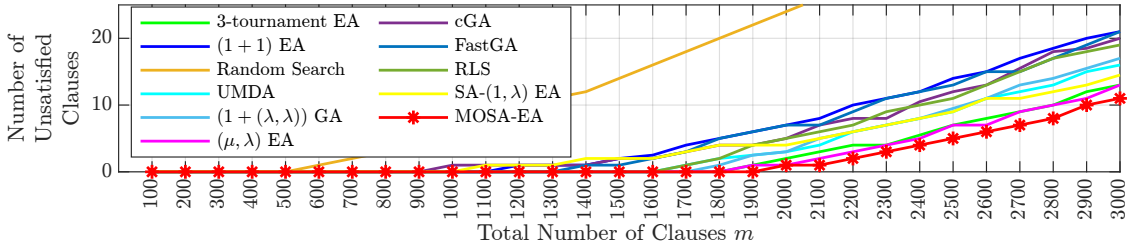

Figure 7.5: The medians of the smallest number of unsatisfied clauses found in 10^8 fitness evaluations on 100 random Max-k-Sat instances with different total numbers of clauses m ($k = 5$, $n = 100$).

Figure 7.6 illustrates the smallest number of unsatisfied clauses of the best solution found during the optimisation process on one random Max-k-Sat instance ($k = 5$, $n = 100$, $m = 2500$). From Figure 7.6, we come to similar conclusions with the experiments on NK-Landscape (Figure 7.4).

Figure 7.8 illustrates the medians of the smallest number of unsatisfied clauses found in one hour CPU-time budget of Open-WBO and the MOSA-EA on 100 random Max-k-Sat

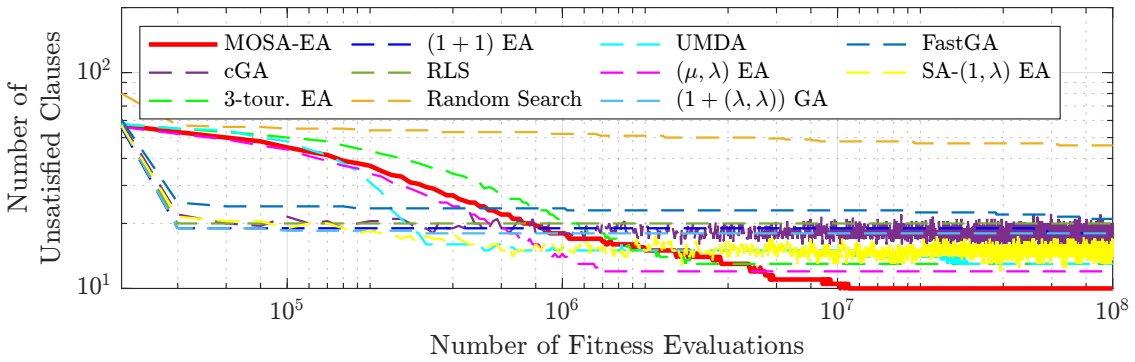

Figure 7.6: Minimum number of unsatisfied clauses over 2×10^4 fitness evaluations over 30 independent runs on one random Max-k-Sat instance ($k = 5$, $n = 100$, $m = 2500$). The axis are log-scaled.

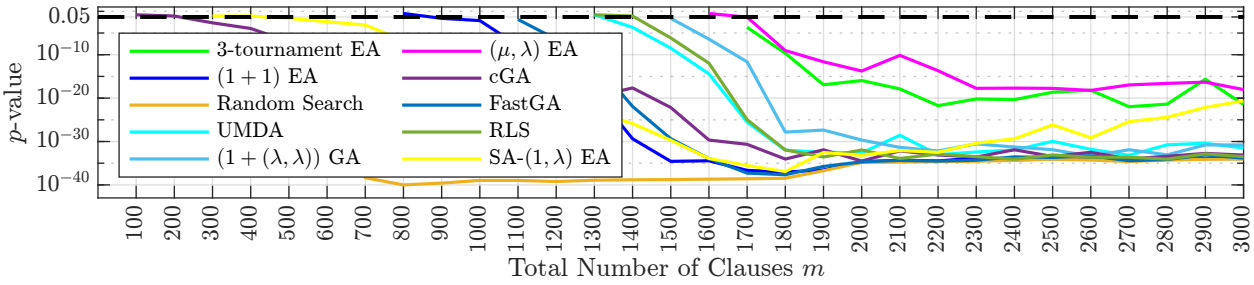

Figure 7.7: The p-values of Wilcoxon rank-sum tests between the algorithms and the MOSA-EA on 100 random Max-k-Sat instances. The y-axis is log-scaled.

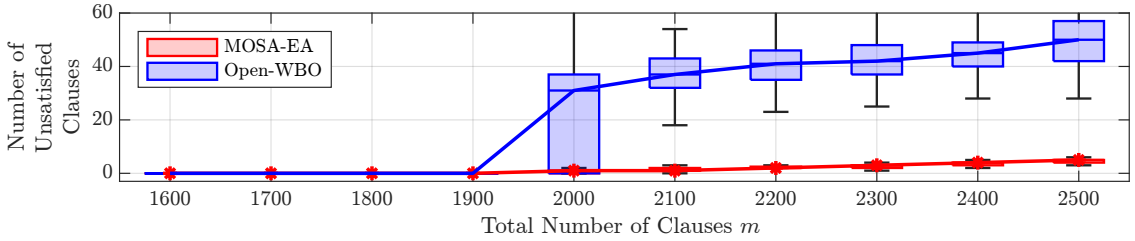

Figure 7.8: Minimum number of unsatisfied clauses found in one hour CPU-time on 100 random Max-k-Sat instances with different total number of clauses m ($k = 5$, $n = 100$).

instances ($k = 5$, $n = 200$) with different total number of clauses m. On instances with few clauses ($m \leq 1900$), Open-WBO returns an optimal solution within a few minutes, while the MOSA-EA takes up to one hour to find an optimal solution. However, the performance

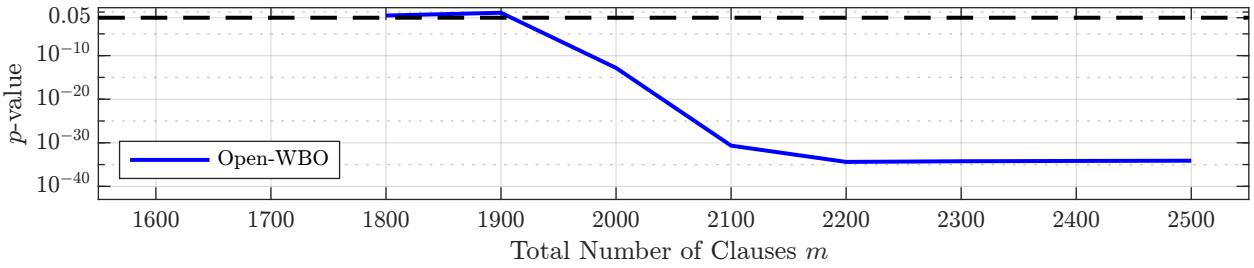

Figure 7.9: The p-value of Wilcoxon rank-sum test between Open-WBO and the MOSA-EA on 100 random Max-k-Sat instances. The y-axis is log-scaled.

of the MOSA-EA is statistically significantly better than Open-WBO on hard instances in one hour of CPU time.

7.5 Conclusion

EAs applied to optimisation problems can benefit from non-elitism. However, it is non-trivial to set the parameters of non-elitist EAs appropriately. Self-adaptation via multi-objectivisation, a parameter control method, is proved to be efficient in escaping local optima with unknown sparsity (in Chapter 6). This chapter continues the study of MOSA-EA through an empirical study of its performance on several combinatorial optimisation problems. We first empirically study the MOSA-EA on theoretical benchmark problems. The performance of the MOSA-EA is comparable with other non-elitist EAs on unimodal functions, i.e., OneMax and LeadingOnes. Self-adaption via multi-objectivisation can also help to cope with sparse local optima. For the NP-hard combinatorial optimisation problems, random NK-Landscape and Max-k-Sat, the MOSA-EA is significantly better than the other nine heuristic algorithms. In particular, the MOSA-EA can beat a state-of-the-art MaxSat solver on some hard random Max-k-Sat instances in a fixed CPU time. In conclusion, the MOSA-EA outperforms a range of optimisation algorithms on several benchmarking function and complex combinatorial optimisation problems.

Chapter Eight

Conclusion

8.1 Summary

EAs are effective solvers for a variety of discrete black-box optimisation problems. However, their performance heavily relies on the proper setting of algorithm parameters, such as mutation rate, which are often problem- and instance-specific. One promising approach to help configure parameters is self-adaptive parameter control, where both parameters and solutions are encoded in individuals and evolved through variation operators. Although there have been some theoretical studies demonstrating the efficiency of self-adaptive EAs on the unknown-structure function, the potential benefits of this approach in other scenarios remain unknown. Moreover, there exists much room for creativity in designing self-adaptive EAs. In this thesis, we have provided significant insights into the self-adaptive parameter control mechanism in EAs, utilising mathematical proofs and empirical analysis.

First, we explored the benefits of self-adaptive parameter control mechanisms for EAs in noisy optimisation. Ideally, we aimed to demonstrate that self-adaptation can identify the "right" parameter settings in noisy optimisation. However, although existing studies emphasise the significance of "right" parameter settings for static EAs in successful noisy optimisation, the exact relationship between parameter settings and noise was unclear. To begin, we conducted a theoretical analysis of the runtime of static non-elitist EAs using two popular selection mechanisms: 2-tournament and (μ, λ) selections, in various uncertain settings. For the 2-tournament EA, we derived a general theorem via the level-based theorem, which provided an upper bound of the expected runtime in uncertain environments. This theorem is then applied to obtain upper bounds of runtimes on OneMax and LeadingOnes in several noise models, including one-bit, bit-wise, Gaussian, and symmetric noise models. In noisy settings, our analysis is more comprehensive and precise compared to the previous study of static non-elitist EAs (Dang and Lehre, 2015). We offer more precise guidance on selecting the mutation rate and population size based on the level of uncertainty. Overall, in multi-

ple noisy settings, we demonstrated that non-elitist EAs surpass the current state-of-the-art results (refer to Tables 2.3-2.10). Specifically, in scenarios optimising the LeadingOnes function in the symmetric noise model, we revealed that a low selective pressure, i.e., a low reproductive rate, and an excessively high mutation rate relative to the noise level, lead to inefficient optimisation, i.e., exponential runtime. This points out the importance of appropriately setting the mutation rate in accordance with the noise level for efficient noisy optimisation and motivates self-adapting the mutation rate in EAs. As another application of the general theorem is that non-elitist EAs optimise the DBV problem within polynomial expected time.

Building on the results of static non-elitist EAs under noise, we conducted, for the first time, a rigorous theoretical analysis of self-adaptive EAs in noisy environments. A detailed runtime analysis of the LeadingOnes function indicates that the 2-tournament EA with self-adaptation from high/low mutation rates guarantees the lowest runtime among fixed high/low mutation rates and a mutation rate uniformly chosen from high/low rates, regardless of the presence of symmetric noise. These results are summarised in Table 4.1. Additionally, we expanded our study to include other types of noise, such as one-bit and bit-wise noise, and examine additional self-adaptive EAs. Both theoretical and empirical results demonstrate that self-adaptive EAs can effectively adjust to noise levels and outperform static EAs in considered noisy settings. Recalling Research Question 1 about the efficiency of self-adaptation in noisy optimisation, our response is that self-adaptive parameter control mechanisms can help configure the parameter setting in EAs according to the noise level.

Subsequently, encouraged by the positive results for noisy optimisation, we asked whether self-adaptation could also be helpful in another form of uncertain optimisation, namely dynamic optimisation (Research Question 2). Our focus is on a problem of tracking dynamic optima with changing structure, specifically the DSM problem, that require adjustable mutation rates. We showed that the EAs with any fixed mutation rate fails to track with

overwhelmingly high probability. In contrast, our theoretical analysis indicates that the (μ, λ) self-adaptive EA can successfully track every optimum in the DSM problem and find the final optimum with an overwhelmingly high probability. In this analysis, to assess the efficacy of the self-adaptive EA in tracking dynamic optima, it is crucial to establish a lower bound for the probability of achieving the current optimum within the specified evaluation budget. Thus, another significant contribution of this work is the level-based theorem with tail bounds (Theorem 5.3.1). In contrast, previous level-based theorems only provide the upper bound of expected runtime. Based on this analysis, we responded to Research Question 2 by concluding that self-adaptive parameter control mechanisms can adjust parameter settings in dynamic optimisation.

Now, we have demonstrated the benefits of existing self-adaptive EAs in uncertain environments. To enhance the performance of self-adaptive EAs on multi-modal landscapes, we proposed a novel self-adaptive EA, named MOSA-EA, that is designed for single-objective optimisation. It addresses parameter control from a multi-objective optimisation perspective, with the goal of simultaneously maximising fitness and mutation rates. A new bimodal function, termed PeakedLO$_{m,k}$, is presented to illustrate the potential benefits of the MOSA-EA. This function is characterised by its adjustable sparsity of local optima. Such sparse local optima can trap elitist EAs. For static non-elitist EAs, it is curial to carefully set the mutation rate according to the sparsity. The runtime analyses show that the MOSA-EA efficiently escapes local optima with unknown sparsity, while elitist EAs, the (μ, λ) EA, and the 2-tournament EA can fail. Table 6.1 demonstrates the theoretical results obtained in this study. An empirical study focusing on complex combinatorial optimisation problems further demonstrates that the MOSA-EA increasingly outperforms other algorithms on more challenging random NK-Landscape and random Max-k-Sat instances. Precisely, the MOSA-EA can find better solutions (higher fitness) for the given evaluation budget. Notably, the MOSA-EA surpasses a problem-specific MaxSat solver, Open-WBO, on several

random Max-k-Sat instances in term of the quality of found solution in the given CPU-time budget. Hence, we responded affirmatively to Research Questions 3 and 4 (self-adaptive EAs in escaping local optima and solving complex combinatorial optimisation problems). According to our research, self-adaptation can efficiently escape a particular type of local optima and contribute significantly on the complex combinatorial optimisation problems.

To summarise, this thesis has provided a comprehensive exploration of self-adaptive parameter control in EAs, shedding light on its advantages and potential applications. Through a combination of comprehensive theoretical proofs, supplemented by empirical analysis, we have demonstrated that self-adaptation of mutation rate can significantly enhance the performance of EAs, notably in noisy and dynamic environments, and also can aid in effectively escaping local optima.

8.2 Future Work

To understand the potential of self-adaptation in EAs, further investigation and clarification are necessary. Potential future research could encompass both theoretical and empirical analysis on self-adaptive EAs to a more diverse range of benchmarking functions and scenarios.

In the present study, Chapter 3 reports one negative result in noisy optimisation, which identifies a mutation rate that fails optimisation under noise, exclusively for static non-elitist EAs in optimising LeadingOnes under a symmetric noise model. For a more comprehensive understanding, additional investigations into the optimisation of OneMax and LeadingOnes under varied noise models, such as one-bit and bit-wise noise models, are recommended. Such exploration could yield fresh insights into the performance of self-adaptive EAs under other more realistic noise models.

Referring to Chapter 4, our theoretical analysis primarily focused on an EA with two self-adapting mutation rates for noisy optimisation. To gain a more rounded understanding, we suggest future research to delve into the performance of other self-adaptive EAs. One algorithm worth considering is the (μ, λ) self-adaptive EA, as proposed by Case and Lehre (2020) and examined in dynamic optimisation (see Chapter 5). Given the multitude of mutation rates involved in this algorithm, we think that it exhibits greater robustness to noise, though posing increased analytical difficulty.

Lastly, extending the use of self-adaptive parameter control mechanisms to other parameters, such as crossover rates, population sizes, and selection parameters, presents a significant research opportunity. As noted in Section 2.3.2, numerous empirical studies have demonstrated the advantages of self-adaptation of parameters beyond just the mutation rate for EAs. However, the understanding, especially from a theoretical perspective, is still lacking.

In conclusion, while substantial progress has been made, there remains a plethora of untapped research opportunities within the field of self-adaptation in EA. Future work can seek to address these unexplored areas to facilitate an even more comprehensive understanding of this fascinating area.

Appendix One

Supplemental Materials

A.1 Definitions

Definition A.1.1 (Asymptotic notations (Cormen et al., 2001)). *For any two functions* $f, g : \mathbb{N}_0 \to \mathbb{R}$*, we define:*

- $f(n) = O(g(n))$ *if and only if there exist constants* $c > 0$ *and* n_0 *such that* $0 \leq f(n) \leq cg(n)$ *for all* $n \geq n_0$*.*
- $f(n) = \Omega(g(n))$ *if and only if* $g(n) = O(f(n))$*.*
- $f(n) = \Theta(g(n))$ *if and only if* $f(n) = O(g(n))$ *and* $f(n) = \Omega(g(n))$*.*
- $f(n) = o(g(n))$ *if and only if* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$*.*
- $f(n) = \omega(g(n))$ *if and only if* $g(n) = o(f(n))$*.*

Definition A.1.2 (FUNNEL function (Dang et al., 2021a)). *Given a bistring* $x \in \{0,1\}^n$ *for*

all $n \in \mathbb{N}$, $0 \leq u \leq v \leq w \leq n$ *and* $u, v, w \in \mathbb{N}_0$ *then*

$$\textsc{Funnel}(x) := \begin{cases} LO(x) & \text{if } w < LO(x) \leq n & (D) \\ LO(x) + u - v & \text{if } v < LO(x) \leq w & (C) \\ LO(x) + w - v & \text{if } u < LO(x) \leq v \text{ and } LO(x) = OM(x) & (B) \\ -n & \text{if } u < LO(x) \leq v \text{ and } LO(x) < OM(x) & (B') \\ LO(x) & \text{if } OM(x) \leq u & (A) \\ -OM(x) & \text{otherwise} & (A') \end{cases} \tag{A.1}$$

where the OneMax *function* $OM(x)$ *and the* LeadingOnes *function* $LO(x)$ *are defined in Definitions 2.2.7 and 2.2.6, respectively.*

A.2 Useful Inequalities and Lemmas

Lemma A.2.1. *For any* $\theta \in (0, 1/2]$ *and any constant* $\zeta \in (0, 1)$, $\theta\zeta < \ln(1 + 2\theta\zeta)$.

Proof. By $\frac{2x}{2+x} \leq \ln(1 + x)$ from Eq. (3) in (Topsoe, 2007), we obtain

$$\ln(1 + 2\theta\zeta) \geq \frac{4\theta\zeta}{2 + 2\theta\zeta} = \frac{2\theta\zeta}{1 + \theta\zeta} > \frac{2\theta\zeta}{2} = \theta\zeta.$$

□

Lemma A.2.2. *For all* $b \in (0, 1)$ *and* $x > 0$, $1 - b^{\frac{1}{x}} \leq -\frac{\ln(b)}{x}$.

Proof. Let $f(x) := 1 - b^{\frac{1}{x}} + \frac{\ln(b)}{x}$. For all $b \in (0, 1)$ and $x > 0$, we know that $f(x)$ is a monotone increasing function by its derivative $f'(x) = \left(b^{\frac{1}{x}} - 1\right) \frac{\ln(b)}{x^2} > 0$. Thus, $f(x) \leq \lim_{x\to\infty} f(x) = \lim_{x\to\infty} \left(1 - b^{\frac{1}{x}} + \frac{\ln(b)}{x}\right) = 0$. □

Lemma A.2.3. *For all* $-1 < x < 0$, $1 + \frac{1}{x} < \frac{1}{\ln(1+x)}$.

Proof. This follows directly from the well-known inequality $\frac{x}{x+1} < \ln(1+x)$, which holds for all $-1 < x < 0$. □

Lemma A.2.4. $\ln(1+x) < \sqrt{x}$ *for* $0 \leq x < \infty$.

Proof. By Eq. (14) in (Topsoe, 2007),

$$\ln(1+x) \leq \frac{x}{\sqrt{1+x}} < \frac{x}{\sqrt{x}} = \sqrt{x}.$$

□

Lemma A.2.5 ((Niculescu and Vernescu, 2004))**.** $\left(1+\frac{x}{n}\right)^n \geq e^x \left(1-\frac{x^2}{n}\right)$ *for* $n \in \mathbb{N}, |x| \leq n$.

Lemma A.2.6. $\left((1-x)^{1/x-1}\right)^y \geq e^{-y}$ *for* $0 < x < 1$ *and* $y > 0$.

Proof. By Lemma A.2.5,

$$(1-x)^{1/x} \geq e^{-1}(1-x) \tag{A.2}$$

$$\implies (1-x)^{1/x-1} \geq e^{-1} \tag{A.3}$$

$$\implies \left((1-x)^{1/x-1}\right)^y \geq e^{-y} \tag{A.4}$$

□

Lemma A.2.7 ((Case and Lehre, 2020))**.** *For all* $c \in (0,1)$ *and* $j \geq 1, 1-c^{1/j} \geq (1-c)/j$.

Lemma A.2.8. $\left(1-\frac{\chi}{n}\right)^i \geq e^{-\chi}(1-o(1))$ *for* $0 < \chi = O(1)$, $n \in \mathbb{N}$ *and* $0 \leq i \leq n$.

Proof.

$$\left(1-\frac{\chi}{n}\right)^i \geq \left(1-\frac{\chi}{n}\right)^n$$

by Lemma A.2.5,

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

□

Lemma A.2.9 ((Dang and Lehre, 2015)). *Let $F(x)$ be the cumulative density function of the normal distribution $\mathcal{N}(0, \sigma^2)$ for $\sigma > 0$ and $x > 0$. we have*

$$F(x) > 1 - \frac{1}{\sqrt{\frac{x\pi}{\sigma\sqrt{2}} + 4}}$$

Lemma A.2.10 ((Dang and Lehre, 2016a)). *Let X and Y be identically distributed independent random variables with integer support, finite expected value μ and finite non-zero variance σ^2, it holds that*

$$\Pr(X = Y) \geq \frac{(1 - 1/d^2)^2}{2d\sigma + 1} \text{ for any } d \geq 1$$

Lemma A.2.11. *If $q \in [0, 1/2)$ and $\psi \in [\psi_0, 1]$ where $\psi_0 \in (0, 1)$, then*

1. *$g(\psi) := \psi\left(2(1-q) - (1-2q)\psi\right)$ is monotone increasing.*
2. *$90(9 - 6q + 12q^2 - 8q^3) < 810$.*
3. *$\frac{1}{10} + \frac{491q}{270} - \frac{728q^2}{135} + \frac{1108q^3}{405} - \frac{8q^4}{135} + \frac{8q^5}{405} \geq -\frac{q}{5} + \frac{1}{10}$.*
4. *$3 - 5q > \frac{1}{2}$.*

Proof.

1. To prove that $g(\psi)$ is a monotone increasing function, we need to show that $g'(\psi) > 0$ for all $\psi \in [\psi_0, 1]$. The derivative of $g(\psi)$ is $g'(\psi) = 2(1-q) - 2(1-2q)\psi$. Since $q \in [0, 1/2)$ and $\psi_0 \in (0, 1)$, we have $2(1-q) - 2(1-2q)\psi_0 > 2((1-q) - (1-2q)) = 2q \geq 0$. Therefore, $g'(\psi) > 0$ for all $\psi \in [\psi_0, 1]$, which means $g(\psi)$ is monotone increasing.

2. We know $90(9-6q+12q^2-8q^3) = 810-90(6q-12q^2+8q^3) = 810-90(1-(1-2q)^3)$. Since $q \in [0, 1/2)$, we know that $0 \leq 1-(1-2q)^3 < 1$ Therefore, $810-90(1-(1-2q)^3) < 810$, which means $90(9 - 6q + 12q^2 - 8q^3) < 810$.

3. The inequality is proved by a non-negative function $\frac{109q}{54} - \frac{728q^2}{135} + \frac{1108q^3}{405} - \frac{8q^4}{135} + \frac{8q^5}{405} \geq 0$ for all $q \in [0, 1/2)$.

4. Since $3 - 5q > 1/2$ for all $q \in [0, 1/2)$, we know that $3 - 5q > \frac{1}{2}$.

□

Lemma A.2.12 is an adaptation of Lemma 4 in (Case and Lehre, 2020), but with some irrelevant items removed from the original. It applies to the values of θ_2, η, and θ_1 as defined in Eq.(5.9)-(5.11).

Lemma A.2.12. *Let $A > 1$, $b < 1$, and $p_{\text{inc}} \in (0, 1)$ be constants satisfying the constraints in Lemma 5.4.1. Then there exists a constant $\delta \in (0, 1/10)$ such that for all $j \in [n]$ and $\chi/n \in [\epsilon, 1/2]$,*

(1) $\theta_2(j) = \Omega(1/j)$,

(2) $\theta_1(j) = O(1/j)$,

(3) $A\eta(j) \leq \theta_2(j)$,

(4) $b\eta(j) \geq \theta_1(j)$,

(5) $b\theta_2(j) < \theta_2(j+1)$,

(6) $A\theta_1(j) \leq \theta_2(j+1)$,

(7) if $\frac{\chi}{n} \leq \eta(j)$, *then* $(1 - A\chi/n)^j \geq \frac{1+\delta}{\alpha_0 p_{\text{inc}}}$,

(8) if $\frac{\chi}{n} \leq \theta_2(j)$, *then* $(1 - b\chi/n)^j \geq \frac{1+\delta}{\alpha_0(1-p_{\text{inc}})}$.

A.3 Useful Theorem

Theorem A.3.1 (Theorem 2 in (Dang and Lehre, 2016b))**.** *For any $x^* \in \{0,1\}^n$, define $T := \min\{t \mid x^* \in P_t\}$, where P_t is the population of the 2-tournament EA at time $t \in \mathbb{N}$. Let $I_t(j)$ denote the j-th sampled index in generation t. Let $R_t(i) := \sum_{j=1}^{\lambda} [I_t(j) = i]$ for $i \in [\lambda]$. Assume that mutation rate χ/n is sampled from the set $\mathcal{M} = \{\chi_1/n, \chi_2/2, \ldots, \chi_m/n\}$ with probabilities $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ one-to-one, every time an individual is mutated. If there exist constant $\alpha_0, c, c', \delta > 0$ such that with probability $1 - e^{-\Omega(n)}$,*

1. the initial population satisfies $H(P_0, x^*) \geq c'n$;

2. for $t \leq e^{cn}$, $i \in [\lambda]$, *if* $H(P_0, x^*) \leq c'n$, *then* $E\left[R_t(i) \mid P_t\right] \leq \alpha_0$,

3. $\sum_{j=1}^{m} p_j e^{-\chi_j} \leq (1-\delta)/\alpha_0$, *and* $\max\{\mathcal{M}\} \leq \chi_{\max}/n$ *for a constant* $\chi_{\max}$,

then $\Pr\left(T \leq e^{c''n}\right) = e^{-\Omega(n)}$ *for a constant* $c'' > 0$.

A.4 Statistical Results of Experiments

Tables A.1-A.20 show statistical results of experiments (Figures 4.3-4.11).

Table A.1: Statistical results of experiments LeadingOnes without noise. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	307948	684124	326388	**283054**
	p-value (to SA-2mr)	0.0008	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
110	Median	361344	866661	372636	**336878**
	p-value (to SA-2mr)	0.0155	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
120	Median	420562	1084456	419604	**388948**
	p-value (to SA-2mr)	0.9649	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
130	Median	480182	1263278	483104	**448040**
	p-value (to SA-2mr)	0.9182	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	530104	1539873	539994	**508346**
	p-value (to SA-2mr)	0.3375	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
150	Median	593776	1882631	613836	**548641**
	p-value (to SA-2mr)	0.2122	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
160	Median	665480	2089912	658368	**620776**
	p-value (to SA-2mr)	0.9183	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
170	Median	736048	2638876	749412	**670256**
	p-value (to SA-2mr)	0.3418	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	800030	2942448	814576	**739768**
	p-value (to SA-2mr)	0.4172	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
190	Median	869400	3369450	886200	**807450**
	p-value (to SA-2mr)	0.5252	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
200	Median	947640	3873240	945520	**877680**
	p-value (to SA-2mr)	0.7947	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.2: Statistical results of experiments LeadingOnes under symmetric noise with noise level $q = 0.1$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	Stat.	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	361424	798452	380786	**354970**
	p-value (to SA-2mr)	0.0093	0.0000	-	0.0000
	p-value (to SA)	0.0647	0.0000	0.0000	-
110	Median	441329	990873	436624	**419686**
	p-value (to SA-2mr)	0.9591	0.0000	-	0.0005
	p-value (to SA)	0.0007	0.0000	0.0005	-
120	Median	499118	1208038	520194	**477084**
	p-value (to SA-2mr)	0.0302	0.0000	-	0.0000
	p-value (to SA)	0.0048	0.0000	0.0000	-
130	Median	561024	1513596	583426	**538622**
	p-value (to SA-2mr)	0.0032	0.0000	-	0.0000
	p-value (to SA)	0.0105	0.0000	0.0000	-
140	Median	625048	1766354	662630	**607246**
	p-value (to SA-2mr)	0.0031	0.0000	-	0.0000
	p-value (to SA)	0.0301	0.0000	0.0000	-
150	Median	729181	2107303	740214	**682040**
	p-value (to SA-2mr)	0.1103	0.0000	-	0.0000
	p-value (to SA)	0.0001	0.0000	0.0000	-
160	Median	814832	2438400	817880	**732536**
	p-value (to SA-2mr)	0.4854	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
170	Median	886136	2884568	908752	**838848**
	p-value (to SA-2mr)	0.0203	0.0000	-	0.0000
	p-value (to SA)	0.0001	0.0000	0.0000	-
180	Median	970426	3425583	974582	**912242**
	p-value (to SA-2mr)	0.9903	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
190	Median	1040550	3891300	1051050	**995400**
	p-value (to SA-2mr)	0.1202	0.0000	-	0.0000
	p-value (to SA)	0.0003	0.0000	0.0000	-
200	Median	1153280	4403240	1167060	**1066360**
	p-value (to SA-2mr)	0.4480	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.3: Statistical results of experiments LeadingOnes under symmetric noise with noise level $q = 0.2$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	2009960	950582	493270	**463766**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0002
	p-value (to SA)	0.0000	0.0000	0.0002	-
110	Median	3126943	1239297	569305	**545780**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0601
	p-value (to SA)	0.0000	0.0000	0.0601	-
120	Median	4400094	1487774	660062	**638986**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0376
	p-value (to SA)	0.0000	0.0000	0.0376	-
130	Median	5707640	1821380	775304	**729526**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0044
	p-value (to SA)	0.0000	0.0000	0.0044	-
140	Median	12454477	2234151	870320	**806035**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0001
	p-value (to SA)	0.0000	0.0000	0.0001	-
150	Median	-	2574701	956862	**894676**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
160	Median	-	2958592	1067816	**1009904**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
170	Median	-	3472584	1148276	**1108184**
	p-value (to SA-2mr)	-	0.0000	-	0.0048
	p-value (to SA)	-	0.0000	0.0048	-
180	Median	-	4041710	1272775	**1228098**
	p-value (to SA-2mr)	-	0.0000	-	0.0031
	p-value (to SA)	-	0.0000	0.0031	-
190	Median	-	4719750	1407000	**1326150**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
200	Median	-	5454760	1516860	**1444780**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

Table A.4: Statistical results of experiments LeadingOnes under symmetric noise with noise level $q = 0.3$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	-	1349808	774480	**709940**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
110	Median	-	1650514	923121	**845018**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
120	Median	-	2042456	1065296	**966622**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
130	Median	-	2471038	1203864	**1112308**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
140	Median	-	2857221	1409325	**1256030**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
150	Median	-	3503479	1560668	**1396176**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
160	Median	-	4056888	1798320	**1559560**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
170	Median	-	4792536	1954228	**1723956**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
180	Median	-	5582547	2145535	**1867083**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
190	Median	-	6087900	2387700	**2071650**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
200	Median	-	7126380	2540820	**2256740**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

Table A.5: Statistical results of experiments OneMax without noise. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	110640	200996	112484	**104186**
	p-value (to SA-2mr)	0.0410	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	172086	356040	177031	**160218**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	238970	536124	241048	**218190**
	p-value (to SA-2mr)	0.0088	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	308594	740194	308594	**276224**
	p-value (to SA-2mr)	0.2689	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	385098	976101	387324	**339465**
	p-value (to SA-2mr)	0.0148	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	454118	1239126	460964	**401632**
	p-value (to SA-2mr)	0.0011	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	530530	1527460	534028	**468732**
	p-value (to SA-2mr)	0.1340	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	608768	1834627	614713	**532672**
	p-value (to SA-2mr)	0.0011	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	686712	2154438	695175	**599664**
	p-value (to SA-2mr)	0.0013	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	768102	2517804	773010	**669942**
	p-value (to SA-2mr)	0.1209	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	847726	2930994	857670	**734613**
	p-value (to SA-2mr)	0.0006	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.6: Statistical results of experiments OneMax under symmetric noise with noise level $q = 0.2$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	129080	234188	132768	**125392**
	p-value (to SA-2mr)	0.0094	0.0000	-	0.0000
	p-value (to SA)	0.0001	0.0000	0.0000	-
140	Median	203734	413402	207690	**191866**
	p-value (to SA-2mr)	0.0006	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	280530	625478	284686	**265984**
	p-value (to SA-2mr)	0.0759	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	364702	864279	369018	**338806**
	p-value (to SA-2mr)	0.0053	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	450765	1149729	454104	**416262**
	p-value (to SA-2mr)	0.0376	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	538552	1451352	546539	**490630**
	p-value (to SA-2mr)	0.0021	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	627308	1768822	631972	**571340**
	p-value (to SA-2mr)	0.0350	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	718156	2109286	726479	**653950**
	p-value (to SA-2mr)	0.0089	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	812448	2508675	819702	**735072**
	p-value (to SA-2mr)	0.0404	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	910434	2914125	916569	**817182**
	p-value (to SA-2mr)	0.0470	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	1003101	3393390	1016774	**904904**
	p-value (to SA-2mr)	0.0047	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.7: Statistical results of experiments OneMax under symmetric noise with noise level $q = 0.3$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	165960	286742	167804	**164116**
	p-value (to SA-2mr)	0.2665	0.0000	-	0.0027
	p-value (to SA)	0.0325	0.0000	0.0027	-
140	Median	259118	507357	262085	**252195**
	p-value (to SA-2mr)	0.1670	0.0000	-	0.0000
	p-value (to SA)	0.0032	0.0000	0.0000	-
180	Median	361572	760548	359494	**344948**
	p-value (to SA-2mr)	0.6812	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	463970	1066052	474760	**441311**
	p-value (to SA-2mr)	0.0007	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	572082	1395702	579873	**542031**
	p-value (to SA-2mr)	0.0465	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	685741	1754858	689164	**641242**
	p-value (to SA-2mr)	0.0475	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	804540	2144274	808038	**753236**
	p-value (to SA-2mr)	0.5338	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	916719	2601532	929798	**860836**
	p-value (to SA-2mr)	0.0049	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	1034904	3051516	1046994	**963573**
	p-value (to SA-2mr)	0.0625	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	1160742	3542349	1170558	**1072398**
	p-value (to SA-2mr)	0.0298	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	1272832	4074554	1287748	**1193280**
	p-value (to SA-2mr)	0.0049	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.8: Statistical results of experiments OneMax under symmetric noise with noise level $q = 0.4$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	8159700	663840	687812	**484972**
	p-value (to SA-2mr)	0.0000	0.0531	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	-	1150207	1146251	**749662**
	p-value (to SA-2mr)	-	0.4214	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
180	Median	-	1738247	1757988	**1056663**
	p-value (to SA-2mr)	-	0.7013	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
220	Median	-	2423434	2392143	**1351987**
	p-value (to SA-2mr)	-	0.9105	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
260	Median	-	3178728	3084123	**1637223**
	p-value (to SA-2mr)	-	0.2991	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
300	Median	-	3974103	3772146	**1964802**
	p-value (to SA-2mr)	-	0.0047	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
340	Median	-	4934512	4805086	**2286526**
	p-value (to SA-2mr)	-	0.2595	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
380	Median	-	5933110	5839179	**2645525**
	p-value (to SA-2mr)	-	0.2878	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
420	Median	-	6925152	7006155	**2951169**
	p-value (to SA-2mr)	-	0.4613	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
460	Median	-	8200041	8006175	**3246642**
	p-value (to SA-2mr)	-	0.4756	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
500	Median	-	9242948	9702858	**3670579**
	p-value (to SA-2mr)	-	0.0186	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

Table A.9: Statistical results of experiments LeadingOnes under one-bit noise with noise level $q = 0.4$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	Stat.	χ_{high}	χ_{low}	SA-2mr	SA
100	Median	467454	1003136	484972	**466532**
	p-value (to SA-2mr)	0.3332	0.0000	-	0.0847
	p-value (to SA)	0.4443	0.0000	0.0847	-
110	Median	554249	1214831	568364	**554249**
	p-value (to SA-2mr)	0.3784	0.0000	-	0.1210
	p-value (to SA)	0.5951	0.0000	0.1210	-
120	Median	643776	1484900	647608	**629406**
	p-value (to SA-2mr)	0.3326	0.0000	-	0.0173
	p-value (to SA)	0.1611	0.0000	0.0173	-
130	Median	722708	1826250	749980	**717838**
	p-value (to SA-2mr)	0.0060	0.0000	-	0.0010
	p-value (to SA)	0.4006	0.0000	0.0010	-
140	Median	810980	2164921	851529	**804057**
	p-value (to SA-2mr)	0.0058	0.0000	-	0.0002
	p-value (to SA)	0.4915	0.0000	0.0002	-
150	Median	942820	2641902	950844	**892670**
	p-value (to SA-2mr)	0.9455	0.0000	-	0.0018
	p-value (to SA)	0.0007	0.0000	0.0018	-
160	Median	1041400	3039872	1041400	**1000760**
	p-value (to SA-2mr)	0.6147	0.0000	-	0.0058
	p-value (to SA)	0.0016	0.0000	0.0058	-
170	Median	1128744	3562020	1142108	**1095848**
	p-value (to SA-2mr)	0.1991	0.0000	-	0.0034
	p-value (to SA)	0.1090	0.0000	0.0034	-
180	Median	1249917	4144571	1267580	**1222903**
	p-value (to SA-2mr)	0.8013	0.0000	-	0.0418
	p-value (to SA)	0.0584	0.0000	0.0418	-
190	Median	1379700	4717650	1397550	**1317750**
	p-value (to SA-2mr)	0.1487	0.0000	-	0.0000
	p-value (to SA)	0.0003	0.0000	0.0000	-
200	Median	1457500	5371020	1502020	**1429940**
	p-value (to SA-2mr)	0.0474	0.0000	-	0.0005
	p-value (to SA)	0.1128	0.0000	0.0005	-

Table A.10: Statistical results of experiments LeadingOnes under one-bit noise with noise level $q = 0.6$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	687812	1294488	813204	**725614**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0247	0.0000	0.0000	-
110	Median	816788	1623225	1036041	**862897**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0143	0.0000	0.0000	-
120	Median	946504	2067364	1184088	**950336**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.6285	0.0000	0.0000	-
130	Median	1135684	2468116	1429832	**1099646**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.4621	0.0000	0.0000	-
140	Median	1295590	2896781	1669432	**1274821**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.9134	0.0000	0.0000	-
150	Median	1470398	3506488	1914727	**1412224**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0096	0.0000	0.0000	-
160	Median	1615440	4017264	2182368	**1568704**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.3236	0.0000	0.0000	-
170	Median	1801056	4786368	2452808	**1751712**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0872	0.0000	0.0000	-
180	Median	2015660	5392410	2855172	**1962671**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.1060	0.0000	0.0000	-
190	Median	2177700	6232800	3099600	**2114700**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0073	0.0000	0.0000	-
200	Median	2417860	7283260	3572200	**2244020**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.11: Statistical results of experiments LeadingOnes under one-bit noise with noise level $q = 0.8$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	2805646	2183296	1660522	**1413426**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
110	Median	4191214	2804180	2123837	**1649573**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
120	Median	6479912	3388446	2557860	**2004136**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
130	Median	16028144	4103462	2965830	**2437922**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	27848262	4930165	3806661	**2781068**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
150	Median	46829067	5873568	4423230	**3160453**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
160	Median	136519920	6896608	5219192	**3762248**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
170	Median	-	8231196	6075480	**4024620**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
180	Median	-	9516201	6661029	**4603809**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
190	Median	-	10533600	7660800	**5009550**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
200	Median	-	12305540	9248500	**5651920**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

Table A.12: Statistical results of experiments OneMax under one-bit noise with noise level $q = 0.85$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	119860	237876	121704	**110640**
	p-value (to SA-2mr)	0.0179	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	183954	423292	187910	**166152**
	p-value (to SA-2mr)	0.0216	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	254555	650414	259750	**229619**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	331253	911755	334490	**295646**
	p-value (to SA-2mr)	0.0068	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	405132	1190910	412923	**358386**
	p-value (to SA-2mr)	0.0039	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	483784	1515248	492912	**429016**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	569008	1860936	578336	**496716**
	p-value (to SA-2mr)	0.0063	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	649194	2246021	655139	**568342**
	p-value (to SA-2mr)	0.1384	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	731445	2696070	742326	**643188**
	p-value (to SA-2mr)	0.0003	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	819636	3082224	822090	**709206**
	p-value (to SA-2mr)	0.0778	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	903661	3574868	912362	**785576**
	p-value (to SA-2mr)	0.0024	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.13: Statistical results of experiments OneMax under one-bit noise with noise level $q = 0.9$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	118016	238798	121704	**111562**
	p-value (to SA-2mr)	0.0202	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	183954	419336	187910	**168130**
	p-value (to SA-2mr)	0.0242	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	254555	649375	257672	**230658**
	p-value (to SA-2mr)	0.0004	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	328016	886938	334490	**293488**
	p-value (to SA-2mr)	0.0002	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	402906	1187571	414036	**362838**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	483784	1495851	490630	**426734**
	p-value (to SA-2mr)	0.0002	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	564344	1829454	573672	**494384**
	p-value (to SA-2mr)	0.0060	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	646816	2221052	661084	**561208**
	p-value (to SA-2mr)	0.0005	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	730236	2609022	742326	**638352**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	812274	3074862	823317	**709206**
	p-value (to SA-2mr)	0.0018	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	901175	3576111	912362	**778118**
	p-value (to SA-2mr)	0.0025	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.14: Statistical results of experiments OneMax under one-bit noise with noise level $q = 0.95$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	118016	236032	121704	**111562**
	p-value (to SA-2mr)	0.0017	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	181976	419336	189888	**166152**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	253516	634829	259750	**230658**
	p-value (to SA-2mr)	0.0143	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	328016	894491	336648	**291330**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	401793	1173102	409584	**358386**
	p-value (to SA-2mr)	0.0071	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	480361	1494710	491771	**424452**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	566676	1824790	567842	**494384**
	p-value (to SA-2mr)	0.1244	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	644438	2192516	649194	**565964**
	p-value (to SA-2mr)	0.0344	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	725400	2602977	735072	**635934**
	p-value (to SA-2mr)	0.0003	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	811047	3049095	823317	**701844**
	p-value (to SA-2mr)	0.0015	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	897446	3484129	904904	**778118**
	p-value (to SA-2mr)	0.0539	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.15: Statistical results of experiments LeadingOnes under bit-wise noise with noise level $p = 1.0/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	1033562	1639316	1168174	**935830**
	p-value (to SA-2mr)	0.0007	0.0000	-	0.0000
	p-value (to SA)	0.0002	0.0000	0.0000	-
110	Median	1274114	2043852	1357863	**1115085**
	p-value (to SA-2mr)	0.0311	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
120	Median	1544296	2528162	1661172	**1279888**
	p-value (to SA-2mr)	0.0051	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
130	Median	1817484	3040828	2015206	**1513596**
	p-value (to SA-2mr)	0.0009	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	2105581	3735453	2379534	**1663498**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
150	Median	2345014	4250714	2775301	**1936793**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
160	Median	2650744	5082032	3143504	**2093976**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
170	Median	3097364	5748576	3461276	**2348980**
	p-value (to SA-2mr)	0.0001	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	3458831	6874024	4010540	**2622436**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
190	Median	3860850	7747950	4544400	**2844450**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
200	Median	4276040	8651720	5010620	**3131240**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.16: Statistical results of experiments LeadingOnes under bit-wise noise with noise level $p = 0.8/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	729302	1335056	833488	**716394**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.6805	0.0000	0.0000	-
110	Median	834667	1640163	1017221	**848782**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.3308	0.0000	0.0000	-
120	Median	1006858	2007968	1226240	**996320**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0396	0.0000	0.0000	-
130	Median	1117178	2420390	1343146	**1116204**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.2900	0.0000	0.0000	-
140	Median	1281744	2898759	1675366	**1272843**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.3418	0.0000	0.0000	-
150	Median	1462374	3467371	1814427	**1357059**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
160	Median	1645920	4069080	2116328	**1557528**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0004	0.0000	0.0000	-
170	Median	1809280	4708240	2329448	**1732180**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0063	0.0000	0.0000	-
180	Median	2021894	5446438	2576720	**1899292**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0001	0.0000	0.0000	-
190	Median	2158800	6226500	3066000	**2088450**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0085	0.0000	0.0000	-
200	Median	2439060	7042640	3425920	**2233420**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.17: Statistical results of experiments LeadingOnes under bit-wise noise with noise level $p = 1.2/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	2842526	2033010	1597826	**1229948**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
110	Median	5901952	2497414	1857534	**1470783**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
120	Median	8872996	3082844	2258006	**1759846**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
130	Median	14521366	3772302	2822652	**2042478**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	24964338	4463357	3229085	**2343930**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
150	Median	68122757	5477383	3774289	**2708100**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
160	Median	-	6089904	4410456	**2995168**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
170	Median	-	7127124	4927204	**3354364**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
180	Median	-	8542658	5852687	**3808974**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
190	Median	-	9383850	6473250	**4179000**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
200	Median	-	10680560	7635180	**4413840**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

Table A.18: Statistical results of experiments OneMax under bit-wise noise with noise level $p = 5\ln(n)/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	617740	911858	595612	**469298**
	p-value (to SA-2mr)	0.2177	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	848562	1578444	843617	**721970**
	p-value (to SA-2mr)	0.3147	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	1181343	2374115	1151212	**1022376**
	p-value (to SA-2mr)	0.0568	0.0000	-	0.0004
	p-value (to SA)	0.0001	0.0000	0.0004	-
220	Median	1419964	3326557	1528943	**1354145**
	p-value (to SA-2mr)	0.1543	0.0000	-	0.0000
	p-value (to SA)	0.1068	0.0000	0.0000	-
260	Median	2067954	4413045	2020095	**1737393**
	p-value (to SA-2mr)	0.5445	0.0000	-	0.0000
	p-value (to SA)	0.0004	0.0000	0.0000	-
300	Median	2218104	5727820	2405228	**2043531**
	p-value (to SA-2mr)	0.2621	0.0000	-	0.0000
	p-value (to SA)	0.0144	0.0000	0.0000	-
340	Median	2616504	7145248	3031600	**2415952**
	p-value (to SA-2mr)	0.0178	0.0000	-	0.0000
	p-value (to SA)	0.0676	0.0000	0.0000	-
380	Median	3408863	8636896	3587213	**2834576**
	p-value (to SA-2mr)	0.3718	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	4271397	10396191	4057404	**3304197**
	p-value (to SA-2mr)	0.5716	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	4885914	12067545	4796343	**3747258**
	p-value (to SA-2mr)	0.8441	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	5132347	14379024	5484116	**4125517**
	p-value (to SA-2mr)	0.7619	0.0000	-	0.0000
	p-value (to SA)	0.0002	0.0000	0.0000	-

Table A.19: Statistical results of experiments OneMax under bit-wise noise with noise level $p = 6\ln(n)/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	1395908	1072286	771714	**624194**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
140	Median	1517126	1754486	1050318	**908891**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
180	Median	2463469	2673347	1440054	**1197967**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
220	Median	2590679	3693417	1825668	**1581814**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
260	Median	3500385	4908330	2356221	**1954428**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
300	Median	5500761	6353088	2936934	**2446304**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
340	Median	5604962	7828524	3580786	**2852036**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
380	Median	9554804	9375265	4132964	**3361303**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
420	Median	12013833	11556831	4866225	**3876054**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
460	Median	16521555	13214790	5755857	**4436832**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-
500	Median	0	15415686	6677396	**4970757**
	p-value (to SA-2mr)	0.0000	0.0000	-	0.0000
	p-value (to SA)	0.0000	0.0000	0.0000	-

Table A.20: Statistical results of experiments OneMax under bit-wise noise with noise level $p = 7\ln(n)/n$. The p-values of each algorithm come from Wilcoxon rank-sum tests between the algorithm and 2-tournament EAs with SA-2mr and SA (100 runs).

n	**Stat.**	χ_{high}	χ_{low}	**SA-2mr**	**SA**
100	Median	-	1368248	1052924	**874056**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
140	Median	-	1981956	1351963	**1103724**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
180	Median	-	2964267	1712272	**1408884**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
220	Median	-	4072146	2124551	**1858038**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
260	Median	-	5353530	2714607	**2274972**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
300	Median	-	6882512	3260978	**2711016**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
340	Median	-	8545614	4049518	**3185512**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
380	Median	-	10453688	4827340	**3842848**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
420	Median	-	12409176	5868486	**4462419**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
460	Median	-	14476146	6646659	**5123952**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-
500	Median	-	16773042	7477888	**5720286**
	p-value (to SA-2mr)	-	0.0000	-	0.0000
	p-value (to SA)	-	0.0000	0.0000	-

References

Achlioptas, Dimitris and Cristopher Moore (2006). "Random k-SAT: Two Moments Suffice to Cross a Sharp Threshold". In: *SIAM Journal on Computing* 36.3, pp. 740–762.

Achterberg, Tobias and Roland Wunderling (2013). "Mixed Integer Programming: Analyzing 12 Years of Progress". In: *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*. Ed. by Michael Jünger and Gerhard Reinelt. Springer Berlin Heidelberg, pp. 449–481.

Adenso-Diaz, Belarmino and Manuel Laguna (2006). "Fine-tuning of algorithms using fractional experimental designs and local search". In: *Operations research* 54.1, pp. 99–114.

Aishwaryaprajna and Jonathan E. Rowe (2023). "The Voting algorithm is robust to various noise models". In: *Theoretical Computer Science* 957, p. 113844.

Ansotegui, Carlos, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney (2015). "Model-based Genetic Algorithms for Algorithm Configuration". In: *Proceedings of International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, pp. 733–739.

Ansótegui, Carlos, Fahiem Bacchus, Matti Järvisalo, and Ruben Martins (2017). "MaxSAT Evaluation 2017". In: *SAT*.

Antipov, Denis, Benjamin Doerr, and Vitalii Karavaev (2022). "A Rigorous Runtime Analysis of the $(1 + (\lambda, \lambda))$GA on Jump Functions". In: *Algorithmica* 84.6, pp. 1573–1602.

Auger, Anne and Benjamin Doerr (2011). *Theory of Randomized Search Heuristics*. World Scientific.

Ausiello, Giorgio, Alberto Marchetti-Spaccamela, Pierluigi Crescenzi, Giorgio Gambosi, Marco Protasi, and Viggo Kann (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer Berlin, Heidelberg.

Bäck, Thomas (1992). "Self-Adaptation in Genetic Algorithms". In: *Self Adaptation in Genetic Algorithms*. MIT Press, pp. 263–271.

Bäck, Thomas, A. E. Eiben, and N. A. L. van der Vaart (2000). "An Emperical Study on GAs "Without Parameters"". In: *Parallel Problem Solving from Nature PPSN VI*. Springer Berlin Heidelberg, pp. 315–324.

Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz (1997). *Handbook of evolutionary computation*. Taylor & Francis.

Bäck, Thomas and Franz Anton Hoffmeister (1991). "Extended Selection Mechanisms in Genetic Algorithms". In: *International Conference on Genetic Algorithms*.

Bäck, Thomas and Martin Schütz (1996). "Intelligent mutation rate control in canonical genetic algorithms". In: *Foundations of Intelligent Systems*. Ed. by Zbigniew W. Raś and Maciek Michalewicz. Springer Berlin Heidelberg, pp. 158–167.

Baker, James Edward (1989). "An analysis of the effects of selection in genetic algorithms". PhD Thesis. Vanderbilt University.

Bambury, Henry, Antoine Bultel, and Benjamin Doerr (2021). "Generalized jump functions". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 1124–1132.

Bartz-Beielstein, Thomas, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel Lopez-Ibanez, Katherine M. Malan, Jason H. Moore, Boris Naujoks, Patryk Orzechowski, Vanessa Volz, Markus Wagner, and Thomas Weise (2020). "Benchmarking in Optimization: Best Practice and Open Issues". In: *arXiv:2007.03488 [cs, math, stat]*.

Boros, Endre and Peter L. Hammer (2002). "Pseudo-Boolean optimization". In: *Discrete Applied Mathematics* 123.1, pp. 155–225.

Böttcher, Süntje, Benjamin Doerr, and Frank Neumann (2010). "Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem". In: *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, pp. 1–10.

Buzdalov, Maxim and Benjamin Doerr (2017). "Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 1343–1350.

Case, Brendan and Per Kristian Lehre (2020). "Self-adaptation in non-Elitist Evolutionary Algorithms on Discrete Problems with Unknown Structure". In: *IEEE Transactions on Evolutionary Computation* 24.4, pp. 650–663.

Cathabard, Stephan, Per Kristian Lehre, and Xin Yao (2011). "Non-Uniform Mutation Rates for Problems with Unknown Solution Lengths". In: *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*. FOGA '11. Association for Computing Machinery, pp. 173–180.

Chen, Shu-Heng, ed. (2002). *Evolutionary Computation in Economics and Finance*. 1st ed. Studies in Fuzziness and Soft Computing. Physica Heidelberg.

Coja-Oghlan, Amin (2014). "The asymptotic k-SAT threshold". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. ACM, pp. 804–813.

Cook, Stephen A. (1971). "The Complexity of Theorem-Proving Procedures". In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Association for Computing Machinery, pp. 151–158.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2001). *Introduction to Algorithms*. 2nd. The MIT Press.

Corus, Dogan, Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre (2018). "Level-Based Analysis of Genetic Algorithms and Other Search Processes". In: *IEEE Transactions on Evolutionary Computation* 22.5, pp. 707–719.

Črepinšek, Matej, Shih-Hsi Liu, and Marjan Mernik (2013). "Exploration and Exploitation in Evolutionary Algorithms: A Survey". In: *ACM Comput. Surv.* 45.3.

Cruz, Carlos, Juan R. González, and David A. Pelta (2011). "Optimization in dynamic environments: a survey on problems, methods and measures". In: *Soft Computing* 15.7, pp. 1427–1448.

Dang, Duc-Cuong, Anton Eremeev, and Per Kristian Lehre (2021a). "Escaping Local Optima with Non-Elitist Evolutionary Algorithms". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35, pp. 12275–12283.

– (2021b). "Non-Elitist Evolutionary Algorithms Excel in Fitness Landscapes with Sparse Deceptive Regions and Dense Valleys". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '21. Association for Computing Machinery, pp. 1133–1141.

Dang, Duc-Cuong, Tobias Friedrich, Timo Kotzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton (2018). "Escaping Local Optima Using Crossover With Emergent Diversity". In: *IEEE Transactions on Evolutionary Computation* 22.3, pp. 484–497.

Dang, Duc-Cuong, Thomas Jansen, and Per Kristian Lehre (2015). "Populations can be Essential in Dynamic Optimisation". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, pp. 1407–1414.

– (2017). "Populations Can Be Essential in Tracking Dynamic Optima". In: *Algorithmica* 78.2, pp. 660–680.

Dang, Duc-Cuong and Per Kristian Lehre (2015). "Efficient Optimisation of Noisy Fitness Functions with Population-based Evolutionary Algorithms". In: *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII - FOGA '15*. ACM Press, pp. 62–68.

– (2016a). "Runtime Analysis of Non-elitist Populations: From Classical Optimisation to Partial Information". In: *Algorithmica* 75.3, pp. 428–461.

– (2016b). "Self-adaptation of Mutation Rates in Non-elitist Populations". In: *Parallel Problem Solving from Nature – PPSN XIV*. Vol. 9921. Springer International Publishing, pp. 803–813.

Dang, Duc-Cuong, Per Kristian Lehre, and Phan Trung Hai Nguyen (2019). "Level-Based Analysis of the Univariate Marginal Distribution Algorithm". In: *Algorithmica* 81.2, pp. 668–702.

Darwin, Charles (1859). *On the Origin of Species*.

De Jong, Kenneth A. (1975). "An analysis of the behavior of a class of genetic adaptive systems." PhD thesis. University of Michigan.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2, pp. 182–197.

Doerr, Benjamin (2022). "Does Comma Selection Help to Cope with Local Optima?" In: *Algorithmica* 84.6, pp. 1659–1693.

Doerr, Benjamin and Carola Doerr (2018). "Optimal Static and Self-Adjusting Parameter Choices for the $(1 + (\lambda, \lambda))$ Genetic Algorithm". In: *Algorithmica* 80.5, pp. 1658–1709.

– (2020). "Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices". In: *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Ed. by Benjamin Doerr and Frank Neumann. Springer International Publishing, pp. 271–321.

Doerr, Benjamin, Carola Doerr, and Franziska Ebel (2015). "From black-box complexity to designing new genetic algorithms". In: *Theoretical Computer Science* 567, pp. 87–104.

Doerr, Benjamin, Carola Doerr, and Johannes Lengler (2019). "Self-Adjusting Mutation Rates with Provably Optimal Success Rules". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '19. Association for Computing Machinery, pp. 1479–1487.

Doerr, Benjamin, Carola Doerr, and Johannes Lengler (2021). "Self-Adjusting Mutation Rates with Provably Optimal Success Rules". In: *Algorithmica* 83.10, pp. 3108–3147.

Doerr, Benjamin, Christian Gießen, Carsten Witt, and Jing Yang (2019). "The $(1 + \lambda)$ Evolutionary Algorithm with Self-Adjusting Mutation Rate". In: *Algorithmica* 81.2, pp. 593–631.

Doerr, Benjamin and Leslie Ann Goldberg (2010). "Drift Analysis with Tail Bounds". In: *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, pp. 174–183.

– (2013). "Adaptive Drift Analysis". In: *Algorithmica* 65.1, pp. 224–250.

Doerr, Benjamin, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges (2013). "Mutation Rate Matters Even When Optimizing Monotonic Functions". In: *Evolutionary Computation* 21.1, pp. 1–27.

Doerr, Benjamin, Daniel Johannsen, and Carola Winzen (2012). "Multiplicative Drift Analysis". In: *Algorithmica* 64.4, pp. 673–697.

Doerr, Benjamin and Timo Kötzing (2021). "Multiplicative Up-Drift". In: *Algorithmica* 83.10, pp. 3017–3058.

Doerr, Benjamin, Huu Phuoc Le, R{́e}gis Makhmara, and Ta Duy Nguyen (2017). "Fast genetic algorithms". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 777–784.

Doerr, Benjamin and Frank Neumann, eds. (2019). *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Natural Computing Series. Springer Cham.

Doerr, Benjamin and Amirhossein Rajabi (2023). "Stagnation detection meets fast mutation". In: *Theoretical Computer Science* 946, p. 113670.

Doerr, Benjamin, Carsten Witt, and Jing Yang (2021). "Runtime Analysis for Self-adaptive Mutation Rates". In: *Algorithmica* 83.4, pp. 1012–1053.

Droste, Stefan (2002). "Analysis of the (1+1) EA for a dynamically changing OneMax-variant". In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02.* Vol. 1, pp. 55–60.

– (2003). "Analysis of the (1+1) EA for a Dynamically Bitwise Changing OneMax". In: *Genetic and Evolutionary Computation — GECCO 2003.* Vol. 2723. Springer Berlin Heidelberg, pp. 909–921.

– (2004). "Analysis of the (1+1) EA for a Noisy OneMax". In: *Genetic and Evolutionary Computation – GECCO 2004.* Vol. 3102. Springer Berlin Heidelberg, pp. 1088–1099.

Droste, Stefan, Thomas Jansen, and Ingo Wegener (2002). "On the analysis of the (1+1) evolutionary algorithm". In: *Theoretical Computer Science* 276.1, pp. 51–81.

– (2006). "Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization". In: *Theory of Computing Systems* 39.4, pp. 525–544.

Eiben, A. E., R. Hinterding, and Z. Michalewicz (1999). "Parameter control in evolutionary algorithms". In: *IEEE Transactions on Evolutionary Computation* 3.2, pp. 124–141.

Eiben, A. E., M. C. Schut, and A. R. de Wilde (2006). "Is Self-adaptation of Selection Pressure and Population Size Possible? – A Case Study". In: *Parallel Problem Solving from Nature - PPSN IX.* Vol. 4193. Springer Berlin Heidelberg, pp. 900–909.

Fleming, P.J and R.C Purshouse (2002). "Evolutionary algorithms in control systems engineering: a survey". In: *Control Engineering Practice* 10.11, pp. 1223–1241.

Fogel, David B. (1998). *Evolutionary Computation: The Fossil Record.* IEEE Press, New York.

– (2000). "What is evolutionary computation?" In: *IEEE Spectrum* 37.2, pp. 26–32.

Friedrich, Tobias, Timo Kotzing, Martin S. Krejca, and Andrew M. Sutton (2016). "The Compact Genetic Algorithm is Efficient under Extreme Gaussian Noise". In: *IEEE Transactions on Evolutionary Computation*, pp. 1–1.

Friedrich, Tobias, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton (2015). "The benefit of recombination in noisy evolutionary search". In: *Algorithms and Computation -*

26th International Symposium, ISAAC 2015, Nagoya, Japan, December 9-11, 2015, Proceedings. Vol. 9472. Springer, pp. 140–150.

Friedrich, Tobias, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton (2016). "Robustness of Ant Colony Optimization to Noise". In: *Evolutionary Computation* 24.2, pp. 237–254.

Galaviz, J. and A. Xuri (1996). "A self-adaptive genetic algorithm for function optimization". In: *Proceedings Mexico-USA Collaboration in Intelligent Systems Technologies*. Pp. 156–161.

Gen, Mitsuo and Runwei Cheng (1999). *Genetic algorithms and engineering optimization*. Vol. 7. John Wiley & Sons.

Gießen, Christian and Timo Kötzing (2016). "Robustness of Populations in Stochastic Environments". In: *Algorithmica* 75.3, pp. 462–489.

Goldberg, David E. and Kalyanmoy Deb (1991). "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms". In: *Foundations of Genetic Algorithms*. Vol. 1. Elsevier, pp. 69–93.

Gottlieb, Jens, Elena Marchiori, and Claudio Rossi (2002). "Evolutionary Algorithms for the Satisfiability Problem". In: *Evolutionary Computation* 10.1, pp. 35–50.

Hajek, Bruce (1982). "Hitting-time and occupation-time bounds implied by drift analysis with applications". In: *Advances in Applied Probability* 14.3, pp. 502–525.

Harik, G.R., F.G. Lobo, and D.E. Goldberg (1999). "The compact genetic algorithm". In: *IEEE Transactions on Evolutionary Computation* 3.4, pp. 287–297.

Harman, Mark, S. Afshin Mansouri, and Yuanyuan Zhang (2012). "Search-Based Software Engineering: Trends, Techniques and Applications". In: *ACM Comput. Surv.* 45.1.

He, Jun and Xin Yao (2004). "A study of drift analysis for estimating computation time of evolutionary algorithms". In: *Natural Computing* 3.1, pp. 21–35.

Hevia Fajardo, Mario Alejandro (2019). "An Empirical Evaluation of Success-Based Parameter Control Mechanisms for Evolutionary Algorithms". In: *Proceedings of the Genetic and*

Evolutionary Computation Conference. GECCO '19. Association for Computing Machinery, pp. 787–795.

– (2023). "Runtime Analysis of Success-Based Parameter Control Mechanisms for Evolutionary Algorithms on Multimodal Problems". PhD Thesis. University of Sheffield.

Hevia Fajardo, Mario Alejandro and Dirk Sudholt (2021a). "Self-Adjusting Offspring Population Sizes Outperform Fixed Parameters on the Cliff Function". In: *Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. FOGA '21. Association for Computing Machinery.

– (2021b). "Self-Adjusting Population Sizes for Non-Elitist Evolutionary Algorithms: Why Success Rates Matter". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '21. Association for Computing Machinery, pp. 1151–1159.

– (2022). "Hard Problems Are Easier for Success-Based Parameter Control". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Association for Computing Machinery, pp. 796–804.

– (2023). "Theoretical and Empirical Analysis of Parameter Control Mechanisms in the (1 + (λ, λ)) Genetic Algorithm". In: *ACM Trans. Evol. Learn. Optim.* 2.4.

Holland, John (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

Huang, Changwu, Yuanxiang Li, and Xin Yao (2020). "A Survey of Automatic Parameter Tuning Methods for Metaheuristics". In: *IEEE Transactions on Evolutionary Computation* 24.2, pp. 201–216.

Hutter, Frank, Holger H Hoos, and Kevin Leyton-Brown (2009). "ParamILS: An Automatic Algorithm Con guration Framework". In: *Journal of Artificial Intelligence Research*, pp. 267–306.

– (2011). "Sequential Model-Based Optimization for General Algorithm Configuration". In: *Learning and Intelligent Optimization*. Springer Berlin Heidelberg, pp. 507–523.

Intriligator, Michael D. (2002). *Mathematical optimization and economic theory*. SIAM.

Jagerskupper, Jens and Tobias Storch (2007). "When the Plus Strategy Outperforms the Comma Strategyand When Not". In: *2007 IEEE Symposium on Foundations of Computational Intelligence*. IEEE, pp. 25–32.

Jansen, Thomas (2013). *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. 1st ed. Natural Computing Series. Springer Berlin Heidelberg.

Jansen, Thomas, Kenneth A. De Jong, and Ingo Wegener (2005). "On the Choice of the Offspring Population Size in Evolutionary Algorithms". In: *Evol. Comput.* 13.4, pp. 413–440.

Jansen, Thomas and Ulf Schellbach (2005). "Theoretical Analysis of a Mutation-Based Evolutionary Algorithm for a Tracking Problem in the Lattice". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. GECCO '05. Association for Computing Machinery, pp. 841–848.

Jansen, Thomas and Ingo Wegener (2002). "The Analysis of Evolutionary Algorithms—A Proof That Crossover Really Can Help". In: *Algorithmica* 34.1, pp. 47–66.

– (2006). "On the analysis of a dynamic evolutionary algorithm". In: *Journal of Discrete Algorithms* 4.1, pp. 181–199.

Jin, Yaochu and Jürgen Branke (2005). "Evolutionary optimization in uncertain environments-a survey". In: *IEEE Transactions on Evolutionary Computation* 9.3, pp. 303–317.

Jones, Terry (1995). "Evolutionary algorithms, fitness landscapes and search". PhD Thesis. University of New Mexico.

Kauffman, Stuart A. and Edward D. Weinberger (1989). "The NK model of rugged fitness landscapes and its application to maturation of the immune response". In: *Journal of Theoretical Biology* 141.2, pp. 211–245.

Kaufmann, Marc, Maxime Larcher, Johannes Lengler, and Xun Zou (2023). "OneMax Is Not the Easiest Function for Fitness Improvements". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by Leslie Pérez Cáceres and Thomas Stützle. Springer Nature Switzerland, pp. 162–178.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). "Optimization by Simulated Annealing". In: *Science* 220.4598, pp. 671–680.

Kötzing, Timo, Andrei Lissovoi, and Carsten Witt (2015). "(1+1) EA on Generalized Dynamic OneMax". In: *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. FOGA '15. Association for Computing Machinery, pp. 40–51.

Kötzing, Timo and Hendrik Molter (2012). "ACO Beats EA on a Dynamic Pseudo-Boolean Function". In: *Parallel Problem Solving from Nature - PPSN XII*. Springer Berlin Heidelberg, pp. 113–122.

Koza, John R. (1989). "Hierarchical Genetic Algorithms Operating on Populations of Computer Programs". In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'89. Morgan Kaufmann Publishers Inc., pp. 768–774.

Lehre, Per Kristian (2010). "Negative Drift in Populations". In: *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, pp. 244–253.

– (2011). "Fitness-levels for non-elitist populations". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*. ACM Press, p. 2075.

Lehre, Per Kristian and Phan Trung Hai Nguyen (2019). "Runtime analysis of the univariate marginal distribution algorithm under low selective pressure and prior noise". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 1497–1505.

– (2021). "Runtime Analyses of the Population-Based Univariate Estimation of Distribution Algorithms on LeadingOnes". In: *Algorithmica* 83.10, pp. 3238–3280.

Lehre, Per Kristian and Xiaoyu Qin (2021). "More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 1160–1168.

– (2022a). "More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments". In: *Algorithmica*.

Lehre, Per Kristian and Xiaoyu Qin (2022b). "Self-Adaptation via Multi-Objectivisation: A Theoretical Study". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Association for Computing Machinery, pp. 1417–1425.

– (2023a). "Self-Adaptation Can Help Evolutionary Algorithms Track Dynamic Optima". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. Association for Computing Machinery, pp. 1619–1627.

– (2023b). "Self-Adaptation Can Improve the Noise-Tolerance of Evolutionary Algorithms". In: *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. FOGA '23. Association for Computing Machinery, pp. 105–116.

Lehre, Per Kristian and Dirk Sudholt (2020). "Parallel Black-Box Complexity With Tail Bounds". In: *IEEE Transactions on Evolutionary Computation* 24.6, pp. 1010–1024.

Lehre, Per Kristian and Carsten Witt (2012). "Black-Box Search by Unbiased Variation". In: *Algorithmica* 64.4, pp. 623–642.

– (2021). "Tail bounds on hitting times of randomized search heuristics using variable drift analysis". In: *Combinatorics, Probability and Computing* 30.4, pp. 550–569.

Lehre, Per Kristian and Xin Yao (2012). "On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 16.2, pp. 225–241.

Lengler, Johannes (2020). "Drift Analysis". In: *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Ed. by Benjamin Doerr and Frank Neumann. Springer International Publishing, pp. 89–131.

Lengler, Johannes and Jonas Meier (2020). "Large Population Sizes and Crossover Help in Dynamic Environments". In: *Parallel Problem Solving from Nature – PPSN XVI*. Springer International Publishing, pp. 610–622.

– (2022). "Large population sizes and crossover help in dynamic environments". In: *Natural Computing*.

Lengler, Johannes and Simone Riedi (2022). "Runtime Analysis of the $(\mu + 1)$-EA on the Dynamic BinVal Function". In: *SN Computer Science* 3.4, p. 324.

Lengler, Johannes and Ulysse Schaller (2018). "The (1+1)-EA on Noisy Linear Functions with Random Positive Weights". In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 712–719.

Lissovoi, Andrei (2016). "Analysis of Ant Colony Optimization and Population-Based Evolutionary Algorithms on Dynamic Problems". PhD Thesis. Technical University of Denmark.

Lissovoi, Andrei and Carsten Witt (2015). "Runtime analysis of ant colony optimization on dynamic shortest path problems". In: *Theoretical Computer Science* 561, pp. 73–85.

– (2017). "A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization". In: *Algorithmica* 78.2, pp. 641–659.

Lobo, Fernando G., Cláudio F. Lima, and Zbigniew Michalewicz, eds. (2007). *Parameter Setting in Evolutionary Algorithms*. Springer Science & Business Media.

López-Ibáñez, Manuel, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle (2016). "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3, pp. 43–58.

Malan, Katherine Mary (2021). "A Survey of Advances in Landscape Analysis for Optimisation". In: *Algorithms* 14.2.

Martins, Ruben, Vasco Manquinho, and Inês Lynce (2014). "Open-WBO: A Modular MaxSAT Solver". In: *Theory and Applications of Satisfiability Testing – SAT 2014*. Vol. 8561. Springer International Publishing, pp. 438–445.

Meyer-Nieberg, Silja (2007). "Self-adaptation in evolution strategies". PhD Thesis. Dortmund University of Technology.

Mühlenbein, Heinz and Gerhard Paaß (1996). "From recombination of genes to the estimation of distributions I. Binary parameters". In: *International conference on parallel problem solving from nature*. Springer, pp. 178–187.

Neumann, Frank and Carsten Witt (2010). *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Natural Computing Series. Springer Berlin, Heidelberg.

Nguyen, Phan Trung Hai (2021). "Runtime Analysis of Univariate Estimation of Distribution Algorithms under Linearity, Epistasis and Deception". PhD Thesis. Univerisity of Birmingham.

Nguyen, Trung Thanh, Shengxiang Yang, and Juergen Branke (2012). "Evolutionary dynamic optimization: A survey of the state of the art". In: *Swarm and Evolutionary Computation* 6, pp. 1–24.

Niculescu, Constantin P. and Andrei Vernescu (2004). "A two sided estimate of $e^x - (1 + x/n)^n$". In: *Journal of Inequalities in Pure and Applied Mathematics* 5.3.

Ochoa, Gabriela (2006). "Error Thresholds in Genetic Algorithms". In: *Evolutionary Computation* 14.2, pp. 157–182.

Ochoa, Gabriela and Francisco Chicano (2019). "Local optima network analysis for MAX-SAT". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, pp. 1430–1437.

Ochoa, Gabriela, Marco Tomassini, Sebástien Vérel, and Christian Darabos (2008). "A study of NK landscapes' basins and local optima networks". In: *Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08*. ACM Press, p. 555.

Oliveto, Pietro S. and Carsten Witt (2011). "Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation". In: *Algorithmica* 59.3, pp. 369–386.

Oliveto, Pietro S. and Christine Zarges (2013). "Analysis of Diversity Mechanisms for Optimisation in Dynamic Environments with Low Frequencies of Change". In: *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. GECCO '13. Association for Computing Machinery, pp. 837–844.

Papadimitriou, Christos H. and Kenneth Steiglitz (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Pintér, János D. (2006). *Global Optimization: Scientific and Engineering Case Studies*. Nonconvex Optimization and Its Applications. Springer New York, NY.

Qian, Chao, Chao Bian, Wu Jiang, and Ke Tang (2019). "Running Time Analysis of the (1+1)-EA for OneMax and LeadingOnes Under Bit-Wise Noise". In: *Algorithmica* 81.2, pp. 749–795.

Qian, Chao, Chao Bian, Yang Yu, Ke Tang, and Xin Yao (2021). "Analysis of Noisy Evolutionary Optimization When Sampling Fails". In: *Algorithmica* 83.4, pp. 940–975.

Qian, Chao, Yang Yu, Ke Tang, Yaochu Jin, Xin Yao, and Zhi-Hua Zhou (2018). "On the Effectiveness of Sampling for Evolutionary Optimization in Noisy Environments". In: *Evolutionary Computation* 26.2, pp. 237–267.

Qin, Xiaoyu and Per Kristian Lehre (2022). "Self-adaptation via Multi-objectivisation: An Empirical Study". In: *Parallel Problem Solving from Nature – PPSN XVII*. Springer International Publishing, pp. 308–323.

Rajabi, Amirhossein (2022). "Single-trajectory Search Heuristics on Discrete Multimodal Optimization Problems". PhD Thesis. Technical University of Denmark.

Rajabi, Amirhossein and Carsten Witt (2021). "Stagnation detection in highly multimodal fitness landscapes". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, pp. 1178–1186.

– (2022). "Self-Adjusting Evolutionary Algorithms for Multimodal Optimization". In: *Algorithmica* 84.6, pp. 1694–1723.

Rechenberg, Ingo (1978). "Evolutionsstrategien". In: *Simulationsmethoden in der Medizin und Biologie*. Springer Berlin Heidelberg, pp. 83–114.

Renegar, James (2001). *A mathematical view of interior-point methods in convex optimization*. SIAM.

Rohlfshagen, Philipp, Per Kristian Lehre, and Xin Yao (2009). "Dynamic Evolutionary Optimisation: An Analysis of Frequency and Magnitude of Change". In: *Proceedings of the 11th*

Annual Conference on Genetic and Evolutionary Computation. GECCO '09. Association for Computing Machinery, pp. 1713–1720.

Rowe, Jonathan E. and Aishwaryaprajna (2019). "The benefits and limitations of voting mechanisms in evolutionary optimisation". In: *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms - FOGA '19*. ACM Press, pp. 34–42.

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

Ruder, Sebastian (2017). *An overview of gradient descent optimization algorithms*.

Schwefel, Hans-Paul (1981). *Numerical optimization of computer models*. John Wiley & Sons, Inc.

Serpell, Martin and James E. Smith (2010). "Self-Adaptation of Mutation Operator and Probability for Permutation Representations in Genetic Algorithms". In: *Evolutionary Computation* 18.3, pp. 491–514.

Slowik, Adam and Halina Kwasnicka (2020). "Evolutionary algorithms and their applications to engineering problems". In: *Neural Computing and Applications* 32.16, pp. 12363–12379.

Smith, Jim (2001). "Modelling Gas with Self Adaptive Mutation Rates". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. GECCO'01. Morgan Kaufmann Publishers Inc., pp. 599–606.

Smith, Jim and T. C. Fogarty (1996). "Self adaptation of mutation rates in a steady state genetic algorithm". In: *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE, pp. 318–323.

Srinivas, N. and Kalyanmoy Deb (1994). "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms". In: *Evolutionary Computation* 2.3, pp. 221–248.

Stadler, Peter F. (2002). "Fitness landscapes". In: *Biological Evolution and Statistical Physics*. Ed. by Michael Lässig and Angelo Valleriani. Springer Berlin Heidelberg, pp. 183–204.

Sudholt, Dirk (2013). "A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation* 17.3, pp. 418–435.

– (2021). "Analysing the Robustness of Evolutionary Algorithms to Noise: Refined Runtime Bounds and an Example Where Noise is Beneficial". In: *Algorithmica* 83.4, pp. 976–1011.

Sudholt, Dirk and Carsten Witt (2016). "Update Strength in EDAs and ACO: How to Avoid Genetic Drift". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, pp. 61–68.

Tong, Hao, Leandro L. Minku, Stefan Menzel, Bernhard Sendhoff, and Xin Yao (2022). "A Novel Generalized Metaheuristic Framework for Dynamic Capacitated Arc Routing Problems". In: *IEEE Transactions on Evolutionary Computation* 26.6, pp. 1486–1500.

Topsoe, Flemming (2007). "Some bounds for the logarithmic function". In: *Inequality Theory and Applications* 4, p. 137.

Vanderbei, Robert J. (2020). *Linear Programming: Foundations and Extensions*. 5th ed. International Series in Operations Research & Management Science. Springer Cham.

Weicker, Karsten (2005). "Analysis of local operators applied to discrete tracking problems". In: *Soft Computing* 9.11, pp. 778–792.

Wilcoxon, Frank (1992). "Individual comparisons by ranking methods". In: *Breakthroughs in statistics*. Springer, pp. 196–202.

Witt, Carsten (2013). "Tight Bounds on the Optimization Time of a Randomized Search Heuristic on Linear Functions". In: *Combinatorics, Probability and Computing* 22.2, pp. 294–318.

Wright, Sewall (1932). "The roles of mutation, inbreeding, crossbreeding and selection in evolution". In: *Proceedings of the sixth international congress of Genetics*. Vol. 1, pp. 356–366.

Zhou, Zhi-Hua, Yang Yu, and Chao Qian (2019). *Evolutionary Learning: Advances in Theories and Algorithms*. Springer Singapore.