Learning in Games via Competitive Coevolution: A Rigorous Runtime Perspective

By

Shishen Lin

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
September 2025

ABSTRACT

Learning in games is a fundamental challenge in machine learning and artificial intelligence, with wide-ranging applications from board games to robust optimisation and adversarial training (Schrittwieser et al., 2020; Silver et al., 2016). In this context, games broadly refer to strategic interactions among players, which may be competitive or cooperative. A particularly rich setting emerges in *adversarial optimisation*, where the goal is to succeed against adaptive or strategic opponents. Such problems are well captured by the framework of *test-based optimisation* (De Jong and Pollack, 2004; Jaśkowski, 2011), where candidate solutions are evaluated based on their performance across evolving sets of test cases, such as in self-play or generative adversarial learning.

Co-evolutionary algorithms (CoEAs) are a class of evolutionary methods well-suited for such adversarial and black-box settings, particularly when gradient information is unavailable (Popovici et al., 2012). In *Competitive CoEAs*, candidate solutions (designs) are evaluated against a population of tests. A classic example is Hillis' sorting network problem (Hillis, 1990), where both the design of sorting networks and their test cases are evolved simultaneously. Despite their promise, these algorithms can suffer from cycling and other failure modes (Wiegand, 2004). Previous theoretical analysis of coevolutionary dynamics relies on modelling coevolutionary computation as a dynamical system under the assumption of the 'infinite population model' (Popovici et al., 2012). The main focus is to analyse the stability of coevolution. Runtime analysis of evolutionary algorithms considers the time complexity of a given randomised algorithm. It provides either lower or upper bounds for the number of fitness function evaluations (called runtime) to understand the performance of given algorithms (Doerr and Neumann, 2020). To the best of our knowledge, the previous runtime analysis of coevolutionary algorithms focuses on a cooperative coevolutionary algorithm, namely CC-(1+1) EA (Jansen and Wiegand, 2004). Runtime analysis of competitive coevolutionary algorithms remains largely underexplored. Our research is driven by the following questions:

1. Under what conditions can competitive coevolutionary algorithms (CoEAs) solve adversarial optimisation problems in expected polynomial time?

2. What are the performance limitations of competitive CoEAs in solving adversarial optimisation problems, in terms of runtime or regret?

To understand the questions above, we proceed by using runtime analysis (Doerr and Neumann, 2020) and regret analysis (Bubeck, Cesa-Bianchi, 2012). In this thesis, we make several theoretical contributions. First, we develop suitable theoretical tools to analyse the runtime of CoEAs. Next, we show that there are no universal good algorithms in adversarial optimisation, also known as Adversarial No Free Lunch. Then, we analyse competitive co-evolutionary algorithms on a case-by-case basis. We compare the traditional evolutionary algorithm $(1, \lambda)$-EA with a competitive co-evolutionary variant $(1, \lambda)$-CoEA. More precisely, we show that the traditional evolutionary algorithm $(1, \lambda)$-EA need exponential time to find the Nash Equilibrium (NE) of DIAGONAL. At the same time, a CoEA can help to find the NE in polynomial time, which shows the promising potential of competitive coevolution. Next, we examine different types of competitive co-evolutionary algorithms. In particular, we analyse the single-pair CoEA against population-based CoEA on the given game benchmark, DIAGONAL. Our analysis reveals that a large population size and mutation rate are key for the polynomial runtime of CoEA on DIAGONAL. These findings advance the theoretical foundation of coevolutionary algorithms and offer deep insights for researchers designing competitive coevolutionary algorithms in adversarial environments, ranging from robust optimisation to game-playing agents. By uncovering when and why coevolution is effective, our work provides a principled basis for developing more efficient algorithms in domains where adaptability and strategic interaction are key.

ACKNOWLEDGMENTS

I would like to express my gratitude to the many individuals who supported me throughout my PhD.

First and foremost, I owe my deepest thanks to my supervisor, Prof. Per Kristian Lehre. His patience, rigour, and sense of duty have been a constant source of motivation. His regular supervision, especially at critical stages, and his comprehensive guidance shaped my approach and taste to research. Under his mentorship, I published several peer-reviewed papers during my PhD; an accomplishment that would not have been possible without him.

I am grateful to the Theory of Evolutionary Computation Group at the University of Birmingham, especially Dr Alistair Benford, Dr Mario Alejandro Hevia Fajardo, and Dr Xiaoyu Qin, for their generous help, detailed comments, and suggestions. From them I learned how to conduct research that is interesting, serious, and rigorous.

I also thank the members of my Thesis Group, Dr Rajesh Chitnis, Prof. Peter Tiňo, and Prof. Jonathan Rowe. Across six intensive meetings, their insights and suggestions improved this thesis and enriched my PhD experience.

Thanks to Prof. Chao Qian for hosting my visit to LAMDA Lab, Nanjing University and Prof. Miqing Li for discussions on evolutionary computation and for their kind advice on my research career.

I am indebted to the wider Evolutionary Computation community. Its collaborative spirit and support for early-career researchers broadened my perspective through many constructive discussions and helpful feedback.

I acknowledge BlueBEAR, the University of Birmingham's high-performance computing service, which was essential to my PhD research.

I am grateful to the Research Committee at the School of Computer Science, University of Birmingham, for supporting my attendance at prestigious international AI conferences, including IJCAI, AAAI, NeurIPS, GECCO and PPSN.

I have been fortunate to be surrounded by excellent colleagues and friends. Their camaraderie, insight, and steady support sustained me throughout my PhD. While I cannot list everyone here, special thanks go to the following individuals, arranged by surname in alphabetical order:

Ms Nanfang Bin; Mr Zijian Cai; Dr Mingjie Chen; Mr Wenbin Chen; Dr Xinyue Chen; Mr Yixin Chen; Dr Zitai Chen; Mr Xiaocheng Cheng; Mr Xinxing Cheng; Ms Shanshan He; Mr Rui Huang; Mr Mingxi Ji; Dr Xi Jia; Mr Chao Jiang; Ms Wen Jiang; Mr Tianshun Li; Dr Wei Li; Mr Zimin Liang; Mr Kuinan Lin; Dr Jiajie Luo; Ms Tong Lv; Mr Ziyang Ma; Mr Jianqiao Mao; Ms Shanshan Mao; Mr Yuchen Peng; Mr Shengjie Ren; Mr Haopu Shang; Ms Xin Tan; Dr Hao Tong; Mr Renjian Wang; Ms Ziwei Wang; Dr Chenguang Xiao; Mr Erxiong Xu; Ms Zihan Xu; Dr Xinyue Xue; Ms Yixuan Yang; Mr Yulong Ye; Dr Yang Yue; Dr Tianyang Zhang; Mr Yulun Zhang; Dr Zhongqu gn Zhang; Ms Feifei Zheng; Mr Ruiquan Zhong; Dr Sijia Zhou.

To everyone, both mentioned and unmentioned, I offer my sincere thanks.

Lastly, and most importantly, my deepest gratitude goes to my parents, my brothers, and my entire family. Their unending love, encouragement, and unwavering faith in me have been the foundation of my academic pursuits.

Contents

Chapter One

Introduction

1.1 General Introduction

Learning in games refers to the process by which individuals[1] adapt and refine their strategies over time through repeated interactions with their opponents, often in response to feedback or observations from them (Fudenberg and Levine, 1998; Rezek et al., 2008). Learning in games is a fundamental challenge in machine learning and artificial intelligence, with numerous applications ranging from board games to robust optimisation tasks (Schrittwieser et al., 2020; Silver et al., 2016). Learning in games lies at the intersection of various disciplines, including (evolutionary) game theory (Fudenberg and Levine, 1998; Weibull, 1997), reinforcement learning (Sutton, Barto, 1998), deep learning (Goodfellow et al., 2014), algorithmic game theory (Roughgarden, 2010) and optimisation (Dempe, 2002; Popovici et al., 2012).

Different domains focus on different problem settings and research interests. For example, classical game theorists (e.g. non-cooperative games) might be interested in what strategies rational players adopt when their outcomes depend on the actions of others. It analyses

[1]These refer to AI algorithms, including evolutionary search heuristic or learning algorithms in the context of this thesis.

players' behaviours in strategic settings, often under assumptions of perfect rationality and complete information (Nash, 1951b; Neumann, 1928; Von Neumann, Morgenstern, 1953). Evolutionary game theorists ask: How do strategies evolve and stabilise over time in large populations? They model adaptation in infinitely large populations as dynamical systems and focus on the stability and robustness of equilibria (Smith, 1982). In reinforcement learning, researchers ask: How can an agent learn to make better decisions through trial and error? They focus on algorithm design for learning optimal policies, often evaluated by regret, with prominent methods including Q-learning (Hu and Wellman, 2003), Proximal Policy Optimisation (PPO) (Schulman et al., 2017), and Monte Carlo Tree Search (MCTS: a core part in Alpha-Zero (Silver et al., 2018)). In deep learning, one of the central questions is: How can we train models that can generate or discriminate complex data distributions? A recent notable success is the Generative Adversarial Network (GAN), where a generator and discriminator are trained in competition (Goodfellow et al., 2014). In algorithmic game theory (Nisan et al., 2007), researchers are interested in how hard it is to compute or approximate equilibria in strategic settings. They aim to understand the computational complexity or computational hardness of computing Nash Equilibrium or approximating Nash Equilibrium, including introducing the well-known PPAD ("Polynomial Parity Arguments on Directed graphs") class and proving that computing a Nash Equilibrium in general-sum games is PPAD-complete (Daskalakis et al., 2009). This assumes white-box access to the entire payoff matrix and analyses how hard it is to solve the problem in terms of time-complexity, measured by the number of elementary operations (such as comparisons or arithmetic steps) the algorithm performs, assuming each takes constant time. Finally, some subfields of optimisation are interested in: How can we efficiently find (or approximate) optimal strategies in adversarial settings? This view treats equilibrium (i.e. Nash Equilibrium or other optimal solution concepts defined in Chapter 2) computation as an optimisation problem and develops efficient algorithms for solving it (Dempe, 2002; Popovici et al., 2012). For example, for two-player zero-sum games, we can solve this optimisation

problem by using linear programming, which guarantees polynomial time[2] (Dantzig, 1951; Maiti et al., 2023) or solve it by bilevel programming (Dempe, 2002). Another approach is evolutionary computation (Popovici et al., 2012), which co-evolves both solutions and problem instances, as seen in the co-design of sorting networks and test cases (Hillis, 1990). In contrast to computational complexity, from the optimisation perspective, especially in black-box settings, the notion of query complexity is more relevant. Here, the algorithm does not have access to the full game description but can query specific payoff entries. Researchers are interested in how many such queries are required to find or approximate a solution. This thesis focuses on this viewpoint, namely query complexity (or runtime defined in Chapter 2), studying adversarial optimisation under limited information, where the cost of information gathering becomes a bottleneck.

A key instance of learning in games arises in settings where an individual's performance is evaluated against one or more opponents, a scenario known as *adversarial optimisation*, where the goal is to succeed despite strategic resistance from adversaries, such as Density Classification Task, Symbolic Regression and Optimisation of Algorithm Parameters (Jaśkowski, 2011). These problems are naturally modelled by *test-based optimisation*, a framework in which the quality of a candidate solution is determined by its interaction with a (typically large and evolving) set of test cases (Bucci et al., 2004; De Jong, 2005; De Jong, 2004a; De Jong and Pollack, 2004; Jaśkowski, 2011; Jaśkowski and Krawiec, 2010; Yo and De Jong, 2007). Unlike classical optimisation, where the objective function is a well-defined and often real-valued mapping from a domain to a scalar performance measure, test-based optimisation[3] does not assume such a function exists in an explicit or efficiently computable form. This paradigm is particularly suited to domains where a solution's quality is inherently relational, emerging only through competitive or cooperative evaluation. For instance, in games such as tic-tac-toe, Go or Chess, evaluating a strategy requires it

[2]Polynomial time means that the time-complexity is polynomial in the size of the input matrix.

[3]We will define test-based optimisation formally in Chapter 2.

to be tested against numerous opposing strategies. Computing a scalar-valued objective function over all possible opponents is infeasible due to a very large number of possible strategies. For tic-tac-toe, this number is estimated to around 3.47×10^{162} (Jaśkowski, 2011), rendering exact evaluation intractable even with modern computing. This motivates how we can design more efficient algorithms for finding optima instead of evaluating a strategy against all possible opposing strategies, especially under the circumstances where players in games have exponentially many strategies.

Test-based optimisation differs fundamentally from classical optimisation in that it treats optimisation as a *process of interaction* rather than scalar function evaluation (Bucci, 2007; Jaśkowski, 2011; Popovici et al., 2012). The objective is no longer to maximise a static function, but rather to identify candidates that perform well across a dynamic or representative subset of tests. Examples include *coevolutionary training*, where model parameters and training data evolve simultaneously (Hastie et al., 2009), and *self-play* in multi-agent systems, such as AlphaGo (Silver et al., 2016) or Stratego (Perolat et al., 2022), where agents iteratively improve through mutual competition. Because the "fitness" of a solution is only meaningful when considered in relation to other entities, test-based optimisation often relies on relative performance metrics, coevolutionary dynamics, and solution concepts borrowed from game theory (e.g., Nash equilibrium, minimax). In this thesis, we adopt the test-based formulation as a unifying framework for studying such adversarial optimisation scenarios.

Many real-world optimisation problems are challenging because the objective function is black-box, noisy, non-differentiable, or shaped by the behaviour of adversaries. In such black-box settings, traditional optimisation techniques, especially those relying on gradients (i.e. gradient descent), often fail or cannot be applied (Bäck et al., 1997). This makes it necessary to use alternative methods that are robust, flexible, and require minimal assumptions about the problem structure. For these reasons, this thesis focuses on evolutionary

algorithms (EAs): a class of bio-inspired, randomised search methods that are particularly well suited to black-box and adversarial optimisation (Bäck et al., 1997; Popovici et al., 2012). EAs, inspired by Darwinian principles of natural selection (Darwin, 1859), operate by maintaining a population of candidate solutions, which are improved over time. In the context of biological metaphor, EAs maintains a population of individuals, where each individual corresponds to a search point of the problem instance. EAs usually contain two key components, *selection* and *variation*. To understand how EAs explore and exploit the search space, it is essential to clarify what these two components entail. The *fitness* value of an individual is often defined by the objective value at the corresponding search point. In each generation, EAs produce *offspring* based on the current population via variation and then select new individuals into the maintaining population based on their fitness values. EAs are powerful tools for discrete black-box optimisation and have shown promise in such settings, especially when gradients are unavailable (Eiben and Smith, 2015; Ruder, 2017). Traditional EAs rely on the selection of the fittest individuals, where individuals are directly ranked by their fitness values and the fittest are more likely to reproduce. More details about EAs can be found in Chapter 2.

The broader field that explores EAs and other bio-inspired randomised heuristic methods is known as evolutionary computation (EC). Originating in the 1950s, EC has a rich history shaped by various independent contributions on different search domains (Eiben and Smith, 2015; Fogel, 1998). EC encompasses a wide range of algorithms, including evolution strategies (ESs) for continuous search spaces (i.e. $\mathbb{R}^n$), initially developed by Ingo Rechenberg and Hans-Paul Schwefel, with other researchers also contributing to their early development (Beyer and Schwefel, 2002; Rechenberg, 1978) and one of notable evolution strategies: CMA-ES developed by Hansen and Ostermeier (2001); genetic algorithms (GAs) for discrete search space (i.e. $\{0, 1\}^n$), introduced by Holland (1992) and significantly advanced by many other researchers, including De Jong (1975); and genetic programming (GP) for symbolic optimisation, developed by many researchers including Banzhaf et al.

(1998), Forsyth (1981) and Koza (1989).

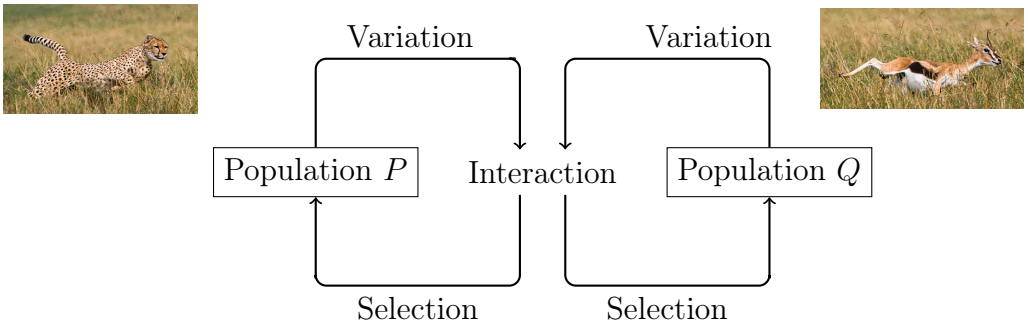


Figure 1.1: A general framework of competitive coevolution (Picture from the slides of the paper (Lehre and Lin, 2025)).

However, in their standard forms, evolutionary algorithms are not directly applicable to learning in games, since adversarial optimisation involves interactive payoff functions that depend on multiple solutions and test cases, rather than simple fitness functions based on individual performance. To address this, another important class of evolutionary algorithms, known as *coevolutionary algorithms* (CoEAs), has been developed (Popovici et al., 2012). Unlike traditional evolutionary algorithms (EAs), which typically involve a single population evolving in a determined fitness landscape, CoEAs involve multiple interacting populations whose fitness landscapes are shaped dynamically by one another. This mirrors the concept of coevolution in nature, where species evolve in response to each other, such as predators and prey, rather than solely in response to their environment. Such a coevolutionary loop may promote genetic polymorphisms in nature (Anderson and May, 1982). It is easy to formulate game learning problems in a way where CoEAs are applicable in strategic and adversarial environments (Popovici et al., 2012). Competitive CoEAs, in particular, model interactions between populations that resemble prey and predator dynamics, and have found success in applications such as Generative Adversarial Networks (GANs) (Al-Dujaili et al., 2018; Goodfellow et al., 2020; Toutouh et al., 2019) and co-learning strategies in games (Mitchell, 2006). As discussed by Dawkins and Krebs (1979), an evolutionary arms race occurs when coevolving competing populations, such as preda-

tors and prey, continuously adapt in response to each other's changes. As one side gains an advantage, it creates new challenges and selective pressure for the other to evolve counter-adaptations, potentially leading to increasing levels of complexity over time. As shown in Figure 1.1, competitive coevolution can mimic this biological arms race: as prey evolve better defences, predators adapt to overcome them, and *vice versa*, which is exactly the goal of competitive coevolutionary algorithms. The interaction between the prey and predator can be naturally viewed as a two-player game, with a *payoff* function capturing the outcome of their encounters.

One of the most influential works in this area was by Hillis (1990), who demonstrated the success of competitive coevolution in optimising sorting networks. In this work, Hillis (1990) employed a coevolutionary genetic algorithm where two populations of sorting networks (prey) and input sequences (predators) evolved simultaneously[4]. The prey aimed to correctly sort the test sequences, while the predators evolved to generate input cases that exposed weaknesses in the networks. This setup created an adversarial dynamic: as one population improved, it directly altered the selection pressures[5] faced by the other. Crucially, this continual adaptation helps to overcome two major limitations of standard evolutionary approaches: premature convergence[6] and inefficient evaluation. As is claimed in (Hillis, 1990), without coevolution, the evolutionary search tended to get stuck in local optima, as most individuals converged on suboptimal solutions and the static test sets failed to challenge them meaningfully. In contrast, Hillis (1990) empirically showed that the coevolutionary system maintained diversity and selection pressure through a form of evolutionary arms race, where each new innovation in the prey population was met with

[4]Different papers about coevolution use different metaphors, including prey/predators, teachers/students, hosts/parasites. In this thesis, we use prey/predators to align with (Lehre, 2022).

[5]Selection pressure refers to the intensity of selection, whether it strongly (high selection pressure) or moderately (low selection pressure) favours the best individuals (Back, 1994).

[6]Premature convergence means that an algorithm has converged too early, resulting in suboptimal solutions (De Jong, 2017).

adaptive challenges from the predator population. This led to both more robust evaluations and more effective exploration of the search space. Hillis's approach notably yielded 16-input sorting networks with only 61 comparison-exchange operations, an improvement over prior evolutionary results and approaching the best known human-designed constructions (i.e. 60 comparison-exchange operations). His work thus illustrated how coevolutionary dynamics can serve not only as a metaphor for biological adaptation, but also as a practical mechanism to enhance the performance of optimisation algorithms. This work, along with earlier contributions such as Axelrod's tournaments (Axelrod, 1987) and Lindgren's work on the evolution of strategies (Lindgren, 1992), brought increased attention to CoEAs. In CoEAs, the mutual adaptation between prey and predators can drive exploration and lead to optimal strategies in challenging settings such as MAXIMIN-optimisation problems, where one seeks to solve $\max_x \min_y f(x, y)$ for a given payoff function f. However, CoEAs can suffer from several pathological behaviours that hinder their broader practical use (Popovici et al., 2012; Rosin, 1997; Wiegand, 2004). One prominent issue is cycling, where the population revisits previously explored areas of the search space without achieving global progress. This is especially common in intransitive domains, where no dominant strategy exists, for instance, A defeats B, B defeats C, but C defeats A. Such cycles can trap the algorithm in inefficient search loops. Another related phenomenon is evolutionary forgetting, where effective past strategies are discarded because they are no longer advantageous against the current opponents (Rosin, 1997). Another pathology is disengagement, which arises when the opponents are either too weak or too strong, resulting in diminished selection pressure and a lack of informative gradient for progress. In such cases, evolution stagnates or regresses. Additionally, overspecialisation can occur, where individuals evolve narrowly targeted strategies that exploit specific opponent behaviours while failing to generalise. These issues compromise the generality and robustness of evolved solutions. To mitigate such pathologies, it is critical to construct a good evaluation environment, namely, a diverse and representative set of opponents, to provide consistent and informative feed-

back during the evolutionary process (Krawiec and Heywood, 2020; Miconi, 2009; Rosin and Belew, 1995a).

Empirical success has made CoEAs increasingly popular, yet still not fully understood due to unexpected pathological behaviour. Can we develop a more rigorous understanding of the mechanism of CoEAs? When assessing the performance of EAs, we aim to measure its use of computational resources. Adopting the standard computer science perspective, we focus on how computational resources required by an algorithm to find a solution depend on the problem size (Jansen, 2013b). In classical complexity theory, computational complexity is generally defined as the number of basic operations executed before the algorithm returns a solution. We distinguish between query complexity (or black-box complexity, bandit feedback) and scenarios where the algorithm can see the entire input. In this thesis, we often refer to complexity as query complexity/black box complexity/bandit feedback, where only limited information is provided at each iteration (i.e. black-box setting). Moreover, instead of considering the number of fitness function evaluations (called the runtime of EAs) in traditional EAs, it is more sensible to consider the payoff function evaluations (called the runtime of CoEAs) in CoEAs since CoEAs only rely on payoff function to capture the interactive outcome of two populations instead of one specify static fitness function of one population in traditional EAs as mentioned earlier. Runtime analysis of traditional evolutionary algorithms considers the time complexity of a given randomised algorithm. It provides either lower or upper bounds for the runtime to understand the performance of the given algorithms (Doerr and Neumann, 2020). To understand the questions above, we extend runtime analysis to coevolutionary algorithms. Runtime analysis can identify relationships between algorithmic parameters and problem characteristics that determine the efficiency of evolutionary algorithms. We expect runtime analysis of CoEAs to bring us similar insights into CoEAs. There is limited literature about runtime analysis of CoEAs apart from (Fajardo et al., 2023; Jansen and Wiegand, 2004; Lehre, 2022). We would like to develop runtime analysis for competitive coevolutionary algorithms, as a deeper theoretical

understanding of CoEAs can reveal how various algorithmic parameters influence their performance and guide the design of more effective variants. We expect that insights from runtime analysis of coevolution (Popovici et al., 2012) will eventually improve the design of coevolutionary algorithms. More related works about CoEAs can be found in Chapter 2.

In this thesis, we aim to develop a further rigorous runtime analysis of competitive CoEAs on simplified adversarial benchmarks. We study under what conditions these algorithms can find optimal solutions efficiently and how key parameters influence their performance. We summarise the following research questions we aim to address.

1.2 Research Questions

Competitive coevolutionary algorithms evolve competing populations, where each population adapts in response to the other, and are often used to model adversarial scenarios. Given a 'payoff function" $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ determining the outcome of interactions, where $\mathcal{X}$ is the set of strategies to the predator, $\mathcal{Y}$ is the set of strategies available to the prey and $g(x, y)$ is the payoff to the predator[7], competitive coevolution often leads to complicated dynamics, where traditional runtime analysis tools cannot be directly applied. Thus, we are interested in the following research questions:

Research Question 1: *Can we develop suitable theoretical tools to analyse the runtime of competitive CoEAs?*

Traditional drift analysis requires the process to remain consistent with a positive drift (an expected tendency reaching the optimum) towards the optimum. In the subsequent discussion detailed in Chapter 3, competitive CoEA on the bilinear problem (a simple pseudo-Boolean maximin benchmark) induces a dynamic with negative drift (an expected

[7]We assume the predator chooses action x and the prey chooses action y.

tendency away from the optimum) when the search point moves towards the optimum. Thus, it is hard to employ the previous tools directly. Consequently, the primary research question emerges: Can we develop a new theoretical tool to overcome this challenge?

Research Question 2: *For any solution concept that we define as a good solution, is there always an algorithm that performs better than others in adversarial optimisation with respect to that comparison? If we focus on Nash Equilibria, what is the computational hardness of solving such problems using general, possibly randomised, search heuristics?*

Wolpert and Macready (1997) show that all traditional black-box optimisation algorithms perform equally on average across all problem instances. Adversarial optimisation scenarios pose a more challenging case to explore, sometimes presenting a counter-intuitive result. In adversarial optimisation scenarios, solution concepts define what a good solution is. Previous studies by Wolpert and Macready (2005) and Service and Tauritz (2008) have demonstrated the existence of 'free lunches' in adversarial optimisation with respect to various solution concepts, including 'maximin' and 'maximisation over all test cases'. However, Nash Equilibrium, as a widely studied solution concept in adversarial learning and maximin optimisation, is less explored in this context. This concept is central to various applications, including adversarial learning models such as GANs (Goodfellow et al., 2014; Heusel et al., 2017) and spatial games (Hemberg et al., 2021; Lehre, 2022; Lehre et al., 2023). Thus, it is natural to ask the following key questions:

(1) Does adversarial optimisation exhibit a 'free lunch' for other widely-used solution concepts in game theory, including Nash Equilibrium?

(2) If we use Nash Equilibrium as the solution concept, how can we characterise the difficulty of black-box adversarial optimisation problems for problem-independent, possibly randomised, search heuristics?

To apply traditional EAs to adversarial optimisation, a common method is to evaluate

individuals by pairing candidate solutions from the two competing roles (e.g., x and y) and defining a fitness function over these fixed pairs $z = (x, y)$. We refer to this methodology as the **fixed-pairing EA** approach. In this setup, the EA evolves one or both populations by treating fitness as a function of performance in a static match-up, typically without modelling an evolving adversary. This contrasts with competitive coevolution, where the adversary population co-adapts over time.

Referencing our discussion in Chapter 5, one may wonder why we cannot directly employ traditional evolutionary algorithms for adversarial optimisation using a fixed pairing approach. Is it really necessary to use competitive coevolutionary approaches? To the best of our knowledge, although previous empirical studies (Hemberg et al., 2021; Hillis, 1990; Lehre et al., 2023) demonstrate the promising potential of competitive coevolution, the necessity of competitive coevolution is barely touched. In particular, there exists no literature on theoretical guarantees or impossibility results for this question. An underlying essential question behind these questions is: what is the key distinction between CoEA and EA? Thus, the third research question is posed as follows.

Research Question 3: *Are there adversarial optimisation problems where competitive coevolutionary algorithms (CoEAs) are more efficient than traditional evolutionary algorithms (EAs) using the fixed-pairing approach? (EA vs CoEA)*

This research question investigates whether there exist adversarial settings in which the co-adaptive dynamics of CoEAs yield a provable or empirical advantage over the fixed-pairing EA baseline.

Research Question 4: *How does the design of coevolution, including population size and mutation rate, impact the performance of coevolution (time-complexity)? (CoEA vs CoEA)*

As resolved by Question 3, we illustrate the necessity of competitive coevolution compared with traditional evolutionary algorithms. How about competitive coevolution itself? Re-

ferring to Chapter 6, although a few empirical studies suggest that CoEAs can effectively solve adversarial optimisation problems, how do parameters in CoEAs, like mutation rate or population size, affect the performance of CoEA? Can every CoEA help to solve adversarial optimisation problems efficiently? It would be intriguing to understand how different CoEAs perform on theoretical benchmarking functions.

1.3 Contributions and Outline

This thesis contributes to the theoretical understanding of coevolutionary algorithms (CoEAs) in adversarial optimisation through rigorous runtime analysis and complexity-theoretic studies. Each chapter tackles a specific aspect of the central theme: analysing the performance and design of CoEAs in solving challenging black-box adversarial optimisation problems. The contributions are as follows:

In **Chapter 2**, we provide essential background knowledge, including definitions of evolutionary algorithms (EAs), benchmarking functions, traditional runtime analysis techniques, and key theoretical tools such as the drift theorem and the level-based theorem. We also review existing work on coevolutionary dynamics and adversarial optimisation.

In **Chapter 3**, we develop new theoretical tools tailored for analysing CoEAs. We present a new drift theorem capable of handling negative drift, introduce an upper-tail bound for runtime, and propose a refined level-based theorem suitable for population-based coevolution. These tools form the analytical foundation for the rest of the thesis and address the challenge of modelling stochastic dynamics in a competitive setting.

In **Chapter 4**, we explore the fundamental hardness of adversarial optimisation. We prove a No Free Lunch theorem for black-box adversarial settings and analyse the query complexity of finding approximate equilibria. This characterises the theoretical limits of

algorithmic performance under general adversarial conditions.

In **Chapter 5**, we compare traditional EAs with CoEAs in the context of sparse binary zero-sum games. We prove that a certain kind of CoEA, $(1, \lambda)$-CoEA, can overcome difficulties that cause traditional EA, $(1, \lambda)$-EA, to stagnate, particularly in escaping from the flat and binary payoff landscape. This demonstrates the advantage of a competitive coevolutionary approach and interaction in adversarial scenarios.

In **Chapter 6**, we compare single-pair coevolution with population-based coevolution. We show that population-based methods are more robust and better at maintaining diverse strategies, resulting in improved performance in challenging zero-sum game settings. These findings highlight the importance of population structure in CoEA design. Moreover, we demonstrate a low mutation rate is also essential for effective performance of CoEAs on the discussed benchmark.

In **Chapter 7**, we summarise the main findings and contributions of the thesis. We reflect on the broader implications of our work for the theory of evolutionary computation and suggest future directions, including tighter runtime bounds and applications to modern adversarial learning frameworks in game-theoretical scenarios.

Each chapter is based on published or accepted papers, co-authored with Prof. Per Kristian Lehre and Dr Mario Alejandro Hevia Fajardo, and the relevant publication is cited at the beginning of each chapter.

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 is listed in alphabetical order.

Table 1.1: Peer-reviewed publications during the PhD study

#	**Title**	**Authors**	**Jour./Conf.**	**Status**
1.	*Randomised Optimism via Competitive coevolution for Matrix Games with Bandit Feedback.*	Lin	IJCAI 2025	Accepted/In Press
2.	*Towards Runtime Analysis of Population-Based co-evolutionary Algorithms on Sparse Binary Zero-Sum Game.*	Lehre, Lin	AAAI 2025	Published (Oral)
3.	*No Free Lunch Theorem and Black-Box Complexity Analysis for Adversarial Optimisation.*	Lehre, Lin	NeurIPS 2024	Published
4.	*Overcoming Binary Adversarial Optimisation with Competitive Coevolution.*	Lehre, Lin	PPSN 2024	Published
5.	*Concentration Tail-Bound Analysis of Coevolutionary and Bandit Learning Algorithms.*	Lehre, Lin	IJCAI 2024	Published (Oral)
6.	*Runtime Analysis of a coevolutionary Algorithm: Overcoming Negative Drift in Maximin-Optimisation.*	Hevia Fajardo, Lehre, Lin	FOGA 2023	Published
7.	*Is CC-(1+1) EA more efficient than (1+1) EA on either separable or inseparable problems?*	Lehre, Lin	CEC 2023	Published

Chapter Two

Background

2.1 Introduction

This chapter provides an overview of the background necessary for understanding the theoretical and empirical study of competitive coevolution. Section 2.2 introduces basic notations and concepts. Section 2.3 outlines solution concepts commonly used in game-theoretic contexts. Section 2.4 discusses the test-based optimisation problems, which are an important class of black-box adversarial optimisation problems. Section 2.5 presents benchmarking functions such as DIAGONAL and BILINEAR games, frequently used in our theoretical analyses in this thesis. Section 2.6 provides a brief introduction to evolutionary algorithms. Section 2.7 offers a comprehensive discussion on coevolutionary algorithms, including both cooperative and competitive variants, classical methods for addressing pathological behaviours, the free lunch theorem in coevolution, and recent runtime analysis results. Finally, Section 2.8 reviews foundational tools in runtime analysis, such as drift theorems and level-based analysis, which are essential in studying the theoretical performance of coevolutionary algorithms.

2.2 Preliminaries

This section introduces the algorithms, runtime analysis and traditional tools, coevolutionary models, and benchmarking functions studied in this thesis. Note that this section will not cover the proposed algorithms and newly developed analysis tools, which are discussed in later chapters. 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. We define the following sets of integers: $[n] := \{i \in \mathbb{N} \mid i \leq n\}$ for $n \in \mathbb{N}$. 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 the binary hypercube, i.e., $\{0,1\}^n$. We call elements of $\{0,1\}^n$ bitstrings of length n with bit values $x = (x_1, \ldots, x_n)$ where $x_i \in \{0,1\}$ for $i \in [n]$. We use 1^n and 0^n to denote the all 1-bit bitstring and the all 0-bit bitstring of length n, respectively. We define $\oplus$ as bitwise exclusive or. We use $\oplus$ to mean that for any two binary bitstrings $x = x_1 \cdots x_n$ and $u = u_1 \cdots u_n$, $x \oplus u := (x_1 \oplus u_1, \cdots, x_n \oplus u_n)$ where $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 1 = 1, 1 \oplus 1 = 0$. Table 2.1 presents a summary of some frequently used notations in this thesis.

When analysing optimisation algorithms, it is common to employ some metrics that measure the closeness between two points in a search space. For instance, assessing how "close" a current solution is to an optimal one may be necessary. In binary or discrete search spaces, the *Hamming distance* and *1-norm distance* are two frequently used methods for evaluating the distance between two bitstrings. We provide their definitions as follows.

Definition 2.2.1 (Hamming, 1986)**.** *Given two bitstrings $x, y \in \{0,1\}^n$ where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ for any $n \in \mathbb{N}$, the Hamming distance between x and y is*

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

Table 2.1: Notation frequently used in this thesis.

Notation	**Definition**
$\mathcal{F}_t$	Natural filtration (see Definition A.1.9 in Appendix)
$\Pr(\cdot)$	Probability measure
$\Pr_t = \Pr(\cdot \mid \mathcal{F}_t)$	Conditional probability with respect to natural filtration
$\mathrm{E}(\cdot)$	Expectation
$\mathrm{E}_t(\cdot) := \mathrm{E}(\cdot \mid \mathcal{F}_t)$	Conditional expectation with respect to natural filtration
$\mathrm{Unif}(A)$	Uniform distribution over a finite 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
$\lvert x \rvert_1$	$\sum_{i=1}^{n} x_i$ where bitstring $x \in \{0,1\}^n$ for $n \in \mathbb{N}$
$\lvert A \rvert$	Cardinality of a set A
$x \oplus u$	Bitwise exclusive or operator between x and u.

Definition 2.2.2. *Given two bitstrings $x, y \in \{0,1\}^n$ where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ for any $n \in \mathbb{N}$, the absolute distance between x and y is*

$$\mathrm{N}(x, y) := ||x|_1 - |y|_1|. \tag{2.2}$$

These two distance functions are widely used in drift analysis (see Section 2.8.1) by setting y to be the optimal binary bitstring in unimodal optimisation problems (i.e. there exists a unique optimum). Equipped with the Hamming distance above, $(\{0,1\}^n, H)$ forms a well-defined metric space (i.e. a non-empty set with a metric satisfying zero-proprieties, non-negativity, symmetry and triangle inequality) (Shirali and Vasudeva, 2005).

Let us begin by introducing two fundamental theoretical functions, OneMax and LeadingOnes, used as benchmark functions in the theory of evolutionary computation. These functions not only provide a basis for evaluating optimisation algorithms but also serve as building blocks for designing more complex problems.

In the theoretical study of EAs, OneMax and LeadingOnes play a crucial role, frequently serving as standard benchmarks to compare and analyse algorithmic performance.

Definition 2.2.3. *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textsc{OneMax}(x) :=: OM(x) := \sum_{i=1}^{n} x_i. \tag{2.3}$$

Definition 2.2.4 (Rudolph, 1997)**.** *Given a bistring $x \in \{0,1\}^n$ for all $n \in \mathbb{N}$, then*

$$\textsc{LeadingOnes}(x) :=: LO(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j. \tag{2.4}$$

2.3 Solution Concepts

In classical optimisation, the goal is typically to find an input that minimises or maximises a given objective function. In such settings, the notion of an *optimum* is well-defined,

either a global or local maximum or minimum. For example, in single-objective function optimisation $f : \mathcal{X} \to \mathbb{R}$, a common solution concept is the point $x^* \in \mathcal{X}$ such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{X}$. More broadly, a *solution concept* refers to a subset of the strategy space that represents acceptable or optimal outcomes under some criteria (Ficici, 2004).

However, in adversarial optimisation tasks, such as those modelled by games or maximin formulations, the objective is not defined over a single decision-maker, but over interacting agents or players whose goals may conflict, i.e., $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. In such settings, classical solution concepts like global optima are no longer appropriate or sufficient. Instead, the outcome depends on the strategic interaction among players, where each player's payoff is influenced by the strategies of others.

To formalise optimality in these settings, we turn to game-theoretic solution concepts. A well-known example is the *maximin* strategy, in which a player seeks to maximise the minimum payoff they can guarantee regardless of the opponent's actions (Wolpert and Macready, 2005). Another common concept in empirical black-box coevolutionary settings is the 'maximisation over all test cases' criterion (Service and Tauritz, 2008), where a candidate solution is evaluated against the entire population of test cases to assess its worst-case or average-case performance. More generally, and particularly in game-theoretic formulations, the *Nash Equilibrium* (NE) is a foundational solution concept (Nash, 1951a; Nisan et al., 2007; Osborne and Rubinstein, 1994). A (pure strategy) Nash Equilibrium is a strategy profile in which no player can improve their payoff by unilaterally deviating from their current strategy. We will define it formally in Definition 2.3.1.

In this work, we adopt the Pure Strategy Nash Equilibrium as our primary solution concept. Our goal is to understand whether and how black-box coevolutionary algorithms can efficiently find NE in games. The choice of NE is motivated by its wide acceptance in both economic theory and algorithmic game theory, and its relevance to adversarial learning

tasks (Goodfellow et al., 2014; Heusel et al., 2017; Nisan et al., 2007; Von Neumann, Morgenstern, 1953; Von Stengel, 2008).

Defining an appropriate solution concept is crucial in coevolutionary settings, where candidate solutions evolve through competitive interactions rather than objective-based search (Ficici, 2004). Ficici's dissertation further formalises solution concepts in coevolution and discusses how coevolutionary dynamics may or may not lead to convergence toward them. Thus, this question motivates our rigorous runtime analysis of coevolutionary algorithms with respect to NE for deeper understanding of the mechanism of coevolution.

Definition 2.3.1 (Nash Equilibrium (Nash, 1951a; Nisan et al., 2007))**.** *Given two finite sets $\mathcal{X}$ and $\mathcal{Y}$ as strategy spaces, and payoff functions $g, h : \mathcal{X} \times \mathcal{Y} \to O$, where O is an ordered set, (x^*, y^*) is a Nash equilibrium if for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $g(x^*, y^*) \geq g(x, y^*)$ and $h(x^*, y^*) \geq h(x^*, y)$. In particular, if the two-player game is zero-sum (that is, if $g(x, y) + h(x, y) = 0$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$), then (x^*, y^*) is a Nash equilibrium if for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y)$. The solution concept is defined by*

$$S = \{(x^*, y^*) \in \mathcal{X} \times \mathcal{Y} \mid \ \forall (x, y) \in \mathcal{X} \times \mathcal{Y}, g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y)\}.$$

Adversarial optimisation tasks are more complex and often counter-intuitive compared to traditional optimisation problems. In such settings, it is crucial to define what constitutes a valid solution or what is meant by optimality. There are other commonly studied solution concepts, including 'maximisation over all test cases' (Service and Tauritz, 2008), and 'maximin' (Wolpert and Macready, 2005).

2.4 Test-Based Optimisation

In many optimisation problems, the quality of a solution cannot be evaluated by a single objective function alone, but must instead be assessed through its performance on a

range of test cases. A classical example arises in the design of *sorting networks*, fixed sequences of comparison-swap operations that must correctly sort any input array (Hillis, 1990; Jaśkowski, 2011). Here, a candidate sorting network (solution) is not evaluated by a scalar cost, but by its ability to sort various input sequences (test cases) correctly and efficiently. A network that fails on even one test input is considered incorrect. Hillis (1990) pioneered the use of coevolution to tackle this challenge, simultaneously evolving both the sorting networks (solutions) and the input sequences (tests), thus introducing a test-based perspective to optimisation.

This idea generalises to a wide class of problems where solutions are evaluated by interacting with a diverse set of test cases or opponents. In such *test-based optimisation* settings, performance is inherently related: a solution is not good in isolation but only relative to how well it performs across the set of tests. Following the definition by Jaśkowski (2011), test-based optimisation is formally defined by a tuple $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, g, P, P^+)$, where $\mathcal{X}$ denotes a set of candidate solutions, $\mathcal{Y}$ is a set of tests, and $G : \mathcal{X} \times \mathcal{Y} \to O$ is an interaction function mapping a solution-test pair to an outcome in a totally ordered set O. The set P contains composite potential solutions (e.g., populations or distributions), and $P^+ \subseteq P$ consists of acceptable or optimal solutions according to some solution concept. We adjust the original definition into a more precise and rigorous definition as follows.

Definition 2.4.1 (Test-based Problem adapted from (Jaśkowski, 2011))**.** *A test-based problem is a tuple*[1] $\mathcal{G}_g = (\mathcal{X}, \mathcal{Y}, g, P^+)$ *that consists of:*

- *a set* $\mathcal{X}$ *of candidate solutions*[2]*,*
- *a set* $\mathcal{Y}$ *of tests*[3]*,*

[1] In some chapters of this dissertation, we also call this tuple a two-player zero-sum game.

[2] It is also known as candidates (Bucci et al., 2004), learners (De Jong, 2004a), prey (Rosin and Belew, 1995b), or components (Popovici et al., 2012).

[3] It is also known as teachers or predators.

- *an interaction function* $G : \mathcal{X} \times \mathcal{Y} \to O$*, where* O *is a totally ordered set,*
- *a solution concept, defining the optimal solutions* $P^+ \subseteq \mathcal{X} \times \mathcal{Y}$*.*

Moreover, given $\mathcal{F}$ *as a subset of all the payoff functions* $g : \mathcal{X} \times \mathcal{Y} \to O$*, we define a class of test-based problems or two-player zero-sum games by* $\mathcal{G} := \{\mathcal{G}_g \mid g \in \mathcal{F}\}$*.*

We revisit the simple example of sorting networks and test cases to illustrate this definition. Consider the application of coevolutionary algorithms in coevolving sorting networks and test cases (Hillis, 1990). We assume the solution concept here is to seek the best sorting network design, which can pass as many test cases as possible. So the set $\mathcal{X}$ contains all possible sorting network designs, and $\mathcal{Y}$ contains all possible test cases. The interaction function g can be viewed as whether the design passes the test cases, which is a totally ordered set $\{\text{True} > \text{False}\}$. In this case, the set of potential solutions P may be simply the set of candidate solutions $\mathcal{X}$, and one could search for a solution which maximises the games' outcomes over all tests from set $\mathcal{Y}$. If such a design exists with respect to the set of test cases $\mathcal{Y}$, then P^+ is non-empty. More details about the empirical studies of test-based optimisation can be found in (Jaśkowski, 2011).

2.5 Benchmarking Functions

To evaluate and understand the performance of coevolutionary algorithms, it is essential to use benchmark functions that expose the specific challenges inherent to adversarial optimisation. Unlike classical single-objective benchmarks, adversarial tasks involve interactions between competing agents, making solution quality dependent on how strategies perform against others rather than in isolation. Benchmark functions for these settings must therefore capture features such as strategic dominance, robustness, and the potential for cyclic dynamics or arms races. We start with simple and structural benchmarks since Chapter 4

will show that the algorithmic performance is independent of the choice of algorithms if the problem has no structure. Moreover, it is reasonable to consider simple and structural benchmarks as starting points and then move to more challenging and general benchmarks.

Well-designed theoretical benchmarks help study such behaviours in a controlled environment. They serve as a building block for our further understanding of complicated dynamics from adversarial optimisation and how competitive coevolution handles complex payoff structures efficiently in this thesis. In this section, we introduce two canonical benchmark functions that are widely used in theoretical studies of coevolution: the DIAGONAL and the BILINEAR functions.

2.5.1 DIAGONAL Games

In order to model the binary test-based optimisation problem inspired by Hillis's method of sorting networks (Hillis, 1990), a payoff function with two strategy spaces $\mathcal{X}, \mathcal{Y}$ as input and $\{0, 1\}$ as output is introduced as follows. In a real-world scenario, a strategy of each player can consist of n binary decisions and this results in an exponentially large strategy space with a binary-valued payoff function (Jaśkowski, 2011; Lehre, 2022; Popovici et al., 2012). More specifically, $\mathcal{X} = \{0, 1\}^n$ is the solution space of a set of designs for sorting networks and $\mathcal{Y} = \{0, 1\}^n$ is the solution space of a set of test cases. We continue to consider a function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. We define $g(x, y) = 1$ if and only if design x passes test case y. Our optimisation problem can be defined in terms of MAXIMIN optimisation (also see Definition 2.3.1). *Find* $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ *such that*

$$\text{for all } (x, y) \in \mathcal{X} \times \mathcal{Y},\ g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y).$$

We want to start our analysis with simple problems with clear structures so we introduce a constraint function c as follows, which splits the search space into several parts.

Definition 2.5.1. *(Generalised boundary test-based problem) Given a constraint function*

$c(z) : \mathbb{R} \to \mathbb{R}$, a generalised boundary function is called GENERAL-BOUNDARY *$g_c : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, where $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^n$, if*

$$g_c(x,y) = \begin{cases} 1 & \text{if } \textsc{OneMax}(y) \leq c\left(\textsc{OneMax}(x)\right) \\ 0 & \text{otherwise.} \end{cases}$$

In our case, we start with a linear constraint function $c(|x|_1) = |x|_1$.

Definition 2.5.2. *For $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^n$, the payoff function* DIAGONAL *: $\mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is*

$$\textsc{Diagonal}(x,y) := \begin{cases} 1 & \text{if } \textsc{OneMax}(y) \leq \textsc{OneMax}(x) \\ 0 & \text{otherwise.} \end{cases}$$

To make it into a binary zero-sum game, we can adjust the payoff from $\{0,1\}$ to $\{-1,1\}$. The analysis on each setting is identical with slight modifications.

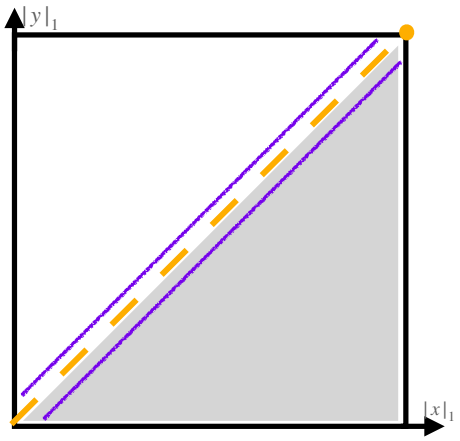


Figure 2.1: Example of DIAGONAL problem. The horizontal axis represents the number of 1-bits in the designs x, and the vertical axis represents the number of 1-bits in the test cases y. The grey area represents search points of payoff 1, and the rest represents search points with payoff 0.

We also use g to denote DIAGONAL. Notice that 1 means the design x passes the test cases $y \in \mathcal{Y}$. In this simple DIAGONAL game, there are exponentially many distinct Maximin

optima, namely all points of the form $(1^n, y)$. In particular, y_n with $|y_n|_1 = n$ represents the most difficult test case, and x_n with $|x_n|_1 = n$ is one of the solutions that can pass y_n (i.e. $g(x_n, y_n) = 1$). Thus, $(1^n, 1^n)$ is the MAXIMIN optimum in this case, satisfying Definition 2.3.1. This thesis aims to explore whether the CoEAs can find such an optimal solution efficiently.

2.5.2 BILINEAR Games

In this section, we consider a simple class of discrete maximin benchmark called BILINEAR, which was first proposed by Lehre, 2022. In this work, we use a variation of BILINEAR as Fajardo et al., 2023 to simplify our calculation and illustrate applications of our main theorem.

Definition 2.5.3 (Fajardo et al., 2023)**.** *The BILINEAR function is defined for two parameters $\alpha, \beta \in (0, 1)$ by*

$$\textsc{Bilinear}_{\alpha,\beta}(x, y) := \textsc{OneMax}(y)\,(\textsc{OneMax}(x) - \beta n) - \alpha n \textsc{OneMax}(x) + E_1 + E_2$$

with the error terms $E_1 := \max\{(\alpha n - \textsc{OneMax}(y))^2, 1\}/n^3$ and $E_2 := -\max\{(\beta n - \textsc{OneMax}(x))^2, 1\}/n^3$. We also denote the set of Nash equilibria as OPT*, where*

$$\text{OPT} := \{(x, y) \mid \textsc{OneMax}(x) = \beta n \wedge \textsc{OneMax}(y) = \alpha n\}.$$

2.6 Evolutionary Algorithms

EAs are a class of bio-inspired optimisation algorithms rooted in Darwinian evolutionary theory (Bäck et al., 1997). As discussed in Chapter 1, EAs maintain a population of individuals, each representing a potential solution to an optimisation problem. These individuals evolve over time through the application of variation and selection operators.

EAs have been successfully applied to a wide range of optimisation problems across diverse domains. For example, they are widely used in operations research and management science (OR/MS) (Sarker et al., 2002), and have been employed to tackle classical combinatorial problems such as the minimum spanning tree, sorting, travelling salesman, and scheduling problems (Hillis, 1990; Neumann and Wegener, 2007; Neumann and Witt, 2010a). We summarise EAs as follows in Algorithm 1.

Algorithm 1 A General Evolutionary Algorithm Adapted from (Eiben and Smith, 2015)

Require: Objective function $f : S \to V$

Require: Initial population $P_0 \in S^\lambda$ randomly

1: **for** each generation number $t \in \mathbb{N}_0$ **do**
2: **for** each $i \in [\lambda]$ **do**
3: Evaluate each candidate $x \in P_t$ by querying $f(x)$
4: Select parents from P_t
5: Mutate the resulting offspring
6: Evaluate new candidates by querying f
7: Select individuals for the next generation

A fundamental concept in EAs is the representation of individuals, which typically distinguishes between the *genotype* (the encoded form) and the *phenotype* (the actual solution). In pseudo-Boolean optimisation, the genotype and phenotype often coincide and are represented as bitstrings (Doerr and Neumann, 2020).

EAs contain two primary processes: variation and selection (Bäck et al., 1997; Eiben and Smith, 2015). Variation generates new individuals by applying random transformations to existing ones. The two principal variation operators are *mutation* and *recombination* (or *crossover*). Mutation modifies parts of the genotype to introduce new individuals. There are two commonly used mutation operators. The first commonly used mutation operator for bistrings is the *1-bit mutation*, which chooses one of the bits in a bitstring $x \in \{0, 1\}^n$

uniformly at random to flip. Another commonly used mutation operator for bitstrings is the *bit-wise mutation*, which independently flips each bit of a bitstring $x \in \{0,1\}^n$ with a small probability χ/n, where $\chi \in (0, n/2]$. These operators are formally described as a random mapping $\text{Mut} : \{0,1\}^n \to \{0,1\}^n$, where $\text{Mut}(x)$ is the bitstring resulting from mutating x. Crossover combines the genetic material of two or more parent individuals to produce offspring and is a defining feature of *Genetic Algorithms (GAs)* (Bäck et al., 1997; Doerr and Neumann, 2020; Eiben and Smith, 2015). A commonly used crossover operator is 1-point crossover, which uniformly at random chooses a point $i \in [n]$ in the parent bitstrings and then swaps the bits to the right of that point between the two parents (i.e., set $z_j = x_j$ for $j \leq i$, and $z_j = y_j$ for $j > i$). There are many other crossover operators, including k-point (i.e. $k \in \mathbb{N}$) and uniform crossover. For a comprehensive review of crossover operators, we refer the reader to (Bäck et al., 1997).

Selection determines which individuals survive and reproduce based on their solution quality. Effective selection mechanisms not only promote individuals with high fitness values but also those with more diverse solutions, such as individuals close in Hamming distance to the global optimum, and sometimes help escape from the local optimum to the global optimum (Bäck et al., 1997; Doerr and Neumann, 2020; Eiben and Smith, 2015). Generally speaking, exploration vs exploitation in evolutionary algorithms is controlled by adjusting selective pressure with respect to the strength of mutation (Lehre, 2010; Lehre and Yao, 2012). Selection strategies can be broadly categorised as *elitist* (preserving the best individuals) or *non-elitist* (allowing the best individuals to be kept or not). Although elitist strategies may appear intuitively superior and have better runtime on a certain class of problems, recent theoretical studies have also demonstrated that non-elitist approaches can help to escape from local optima (Dang et al., 2021a,c). However, the success of such strategies depends heavily on appropriate parameter tuning, particularly of the mutation rate (Lehre, 2010). There are other theoretical analyses of elitism and non-elitism in EAs (Dang and Lehre, 2024; Doerr, 2020; Jagerskupper and Storch, 2007). Note that

in this paper, we do not make a significant distinction between elitism and non-elitism in CoEAs to simplify the analysis at this stage.

One of the simplest evolutionary algorithms, and a widely studied baseline in theoretical analysis, is the (1+1) Evolutionary Algorithm ((1+1) EA), shown in Algorithm 2. It presents a minimal evolutionary process involving bit-wise mutation and elitist selection, using only a single individual throughout the process. In each generation, the current solution generates one offspring via independent bit flips with probability χ/n, as previously described in the bit-wise mutation operator. The offspring replaces the parent if it is at least as fit; otherwise, the parent is retained. This selection strategy is elitist, guaranteeing non-decreasing fitness over time. Despite its simplicity, the (1+1) EA provides a fundamental basis for understanding the interplay between variation and selection in EAs, especially when analysing the algorithmic dynamics in pseudo-Boolean optimisation (Doerr and Goldberg, 2013; Doerr and Neumann, 2020; Droste et al., 2002a; Witt, 2013). Importantly, the mutation rate χ/n is the sole parameter of the algorithm and has a critical influence on performance. To analyse this algorithm, we use a theoretical tool called drift theorem, which will be introduced in Section 2.8.1. Moreover, in this thesis, we will also start our analysis with the simplest (1+1) coevolutionary variant.

Another important class of evolutionary algorithms is the family of *population-based* EAs (Corus et al., 2018; Dang et al., 2021a; Doerr, 2020), which maintain and evolve a set of individuals rather than a single solution. These algorithms apply variation and selection across the population to enhance exploration and preserve genetic diversity. In each generation, a new population is generated by sampling offspring according to a distribution $\mathcal{D}$ that depends on the current population. This general structure is formalised in Algorithm 3, where $\mathcal{D} : \mathcal{X}^\lambda \to (\Omega \to \mathcal{X})$ defines a mapping from the current population to a probability distribution over new individuals, and λ denotes the population size. The population-based framework covers a broad range of classical and modern EAs, including

Algorithm 2 (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_τ with probability χ/n

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

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

5: **else**

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

(μ, λ) EA and so on.

Algorithm 3 Population-based Algorithm (Corus et al., 2018)

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

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

Require: Map $\mathcal{D} : \mathcal{X}^\lambda \to (\Omega \to \mathcal{X})$, 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 $P_{\tau+1}(i) \sim \mathcal{D}(P_\tau)$.

Compared to single-individual algorithms such as the $(1 + 1)$ EA, population-based EAs exhibit more complex stochastic behaviour since a population of individuals scales up the difficulty of the analysis. The underlying stochastic process is less tractable when using only the original drift analysis. To overcome these challenges, general analytical tools like the *Level-Based Theorem* (Corus et al., 2018) have been developed. This theorem provides a flexible and powerful framework for analysing the expected runtime of a wide class of population-based algorithms under mild and natural conditions, which will be introduced

in Section 2.8.2.

2.7 Coevolutionary Algorithms

Traditional evolutionary algorithms (EAs) evaluate individuals using a fitness function that measures their performance independently within a single population (Eiben and Smith, 2015). However, in Coevolutionary Algorithms (CoEAs), particularly those addressing adversarial optimisation problems requiring interaction evaluation, such direct fitness evaluation cannot be straightforwardly applied. Instead, CoEAs employ interactive fitness evaluation, whereby the fitness of an individual is assessed relative to its performance against individuals from other populations. This interdependent evaluation process is a defining characteristic of CoEAs [4].

CoEAs are typically categorised into cooperative and competitive forms. In cooperative CoEAs, multiple populations evolve to jointly optimise a common objective, often through collaboration and decomposition of the problem into subcomponents (Jansen and Wiegand, 2004; Popovici et al., 2012). By contrast, competitive CoEAs model adversarial scenarios where populations represent opposing agents or strategies, each aiming to outperform the others in a game-theoretic setting (Popovici et al., 2012).

2.7.1 Cooperative Coevolutionary Algorithms

Cooperative Coevolutionary Algorithms (Cooperative CoEAs) are a class of population-based search heuristics that aim to solve optimisation problems by decomposing them into smaller, interacting subcomponents. For example, in pseudo-Boolean optimisation

[4] In other words, "Two individuals are compared on the basis of their outcomes from interactions with other individuals" (Popovici et al., 2012).

$f(x) = f(x_1, \ldots, x_{1_k-1}, x_{1_k}, \ldots, x_{2k-1}, , \ldots)$ where population $i \in [n]$ is responsible for the bits $x_{i_k}...x_{(i+1)k-1}$, Jansen and Wiegand (2004) consider the division of a bitstring $x \in \{0,1\}^n$ into k separate subcomponents of equal size n/k, i.e. $x' \in \{0,1\}^{n/k}$. Each subcomponent is handled by a distinct subpopulation, and individuals are evaluated based on their contribution to composite solutions assembled from representatives of all populations. To analyse the behaviour and performance of cooperative coevolution, researchers often begin with abstract models, i.e., pseudo-Boolean functions of the form $f : \{0,1\}^n \to \mathbb{R}$, which serve as models for some real-world tasks such as neural architecture search or combinatorial problems. Additionally, these functions are widely used as benchmarks in the theory of evolutionary computation. Runtime analysis on specific classes of pseudo-Boolean functions, such as Linear Function, OneMax, and Trap has yielded important insights into the fundamental behaviour of cooperative CoEAs (Jansen and Wiegand, 2004; Wiegand, 2004).

In most theoretical literature on runtime analysis of evolutionary algorithms on pseudo-Boolean problems, the optimisation task is framed as follows:

Definition 2.7.1 (Single-objective, static maximization problem (Boros and Hammer, 2002; Droste et al., 2006; Wiegand, 2004))**.** *Let $f : \{0,1\}^n \to \mathbb{R}$ be a function. The goal is to find $x_* \in \{0,1\}^n$ such that*

$$x_* \in \operatorname*{argmax}_{x \in \{0,1\}^n} f(x).$$

Early empirical successes led several authors to suggest that cooperative coevolution could greatly enhance scalability by decomposing large problems into smaller sub-tasks (Potter, 1997; Wiegand, 2004). Additionally, Potter (1997) also analysed the impact of several structural problem features on the effectiveness of cooperative coevolution. These include: the amount and structure of interdependence between subcomponents (i.e., contradictory interactions); the scale or dimensionality of the problem, and the degree of noise present in the fitness evaluation process. Potter (1997) showed that as interdependencies increased,

the advantage of cooperative over standard EAs became more pronounced. For large-scale functions like Rosenbrock (Potter, 1997; Rosenbrock, 1960), increasing dimensionality appeared to amplify the benefit of cooperative decomposition. However, on noisy landscapes (e.g., stochastic De Jong functions (De Jong, 1975; Potter, 1997)), cooperative CoEAs suffered due to misleading evaluations. Building on this, Potter and De Jong (2000) proposed general frameworks for coevolutionary architectures targeting compositional problem domains. Wiegand and Potter (2006) later formalised the concept of robustness in this setting (robustness is informally described as a component's ability to perform well across a variety of partner behaviours, i.e., being effective even when other sub-populations change), introducing criteria to evaluate stability under noise or perturbed representations.

Along with the work above, large-scale optimisation via cooperative coevolution has been one of the primary motivations for cooperative CoEAs. Yang et al. (2008) proposed the DECC-G framework, a cooperative coevolution approach designed for high-dimensional, non-separable problems. Unlike earlier models that assume separability, DECC-G attempted to overcome interdependencies through adaptive decomposition strategies (i.e., combining self-adaptive neighbourhood search with differential evolution). Their empirical results demonstrated that cooperative CoEAs could successfully address certain forms of non-separable problems that are otherwise difficult for standard EAs. However, the theoretical understanding of these behaviours remains limited.

A key consequence of these cross-population evaluations is that the rankings of pairs of individuals are dynamic and can fluctuate over time, potentially leading to cyclic behaviours. This makes CoEAs both more intricate and more analytically intriguing than standard EAs. Neumann et al. (2022) investigated the role of diversity in cooperative CoEAs. While diversity has been widely recognised as important in evolutionary computation, its specific effect on cooperative dynamics has only recently begun to be analysed, mostly from an empirical viewpoint. Theoretical tools for studying this phenomenon in

cooperative coevolution remain underdeveloped and represent an open area of research.

The simplest cooperative evolutionary algorithm is the Cooperative Coevolutionary (1+1) EA (CC-(1+1) EA). The CC-(1+1) EA first partition a bit-string into k blocks $I_1, \ldots, I_k$. Then, the algorithm mutates each block in one iteration and selects the best individual based on fitness. The algorithm repeats this process until all blocks are updated and the termination criteria are met. A first rigorous mathematical analysis of cooperative coevolution was provided by Jansen and Wiegand (2004), who analysed CC-(1+1) EA on both separable and inseparable pseudo-Boolean functions.

Theorem 2.7.1 (Lower Bound for Runtime of CC-(1+1) EA on Linear Functions (Jansen and Wiegand, 2004))**.** *The expected optimisation time of the CC-(1+1) EA on a linear function with only non-zero weights is* $\Omega(n \log n)$.

Theorem 2.7.2 (Upper Bound for Runtime of CC-(1+1) EA on Linear Functions (Jansen and Wiegand, 2004))**.** *The expected optimisation time of the CC-(1+1) EA on a linear function is* $O(n \log^2 n)$.

Their first conclusion is that separability alone is insufficient for speed-up: for perfectly separable benchmarks such as OneMax and LeadingOnes, the cooperative algorithm exhibits the same asymptotic runtime as a standard $(1+1)$ EA, which is $\Theta(n \log n)$ (Droste et al., 2002a). For others, notably functions including Trap engineered to contain deceptive interdependencies, the algorithm is provably misled and fails to reach a global optimum with overwhelmingly high probability, regardless of how long it runs (Jansen and Wiegand, 2004). Thus, while separability is not sufficient to guarantee success, neither is inseparability sufficient to guarantee failure (e.g. the runtime of CC-(1+1) EA on LeadingOnes is also $\Theta(n^2)$, the same runtime as (1+1)-EA (Witt, 2006)). What matters is the precise pattern of cross-component interactions. Note that we tackle an open conjecture left in (Jansen and Wiegand, 2004) by showing that the expected optimisation time of the CC-(1+1) EA

on a linear function is $O(n \log n)$, a matching upper bound as Theorem 2.7.1 in (Lehre and Lin, 2023).

Cooperative coevolutionary algorithms have shown promising results across a range of applications, including modular neural network training, manufacturing scheduling, function optimisation, and artificial life simulations (Wiegand, 2004). These successes are often credited to their modular design and parallel search capability, which allows the algorithm to adapt flexibly to problem structure. However, both empirical and theoretical studies offer seemingly conflicting insights; some demonstrate clear acceleration (Potter, 1997), while others show no improvement or even degradation (Jansen and Wiegand, 2004). This difference arises because such conclusions are often drawn from specific benchmark settings. In practice, cooperative coevolution can lead to genuine speedups, but only when decomposition yields meaningful sub-tasks and sufficient cross-component diversity is preserved. Without these conditions, the cooperative framework may provide limited benefit or harm performance. Consequently, the question of under what conditions separability universally accelerates cooperative coevolution remains open, particularly for broader and more complex problem classes.

2.7.2 Overview on Competitive CoEAs

Competitive Coevolutionary Algorithms (CoEAs) extend traditional evolutionary computation by introducing adversarial dynamics between two or more populations. Rather than evolving solutions in isolation, individuals are evaluated through interactions with opponents, typically drawn from other evolving populations, and such interactions are determined by a payoff function. This framework replaces absolute fitness with relative performance. In competitive coevolution, we introduce several game-theoretical concepts to describe the interactions in the optimisation problems studied. These interactions often occur in multiplayer games where each player competes with others. Several paradigms

have been proposed to model such interactions, including the prey–predator model (Gomes et al., 2014b; Hevia Fajardo et al., 2023; Wiegand, 2004), the learner–evaluator model (De Jong et al., 2007; Hillis, 1990; Wiegand, 2004), and the host–parasite model (Rosin and Belew, 1997; Rosin, 1997). In the learner–evaluator setting, evaluators may act as test cases, and changes in the set of evaluators can affect the apparent ranking of learners (De Jong and Pollack, 2004).

One of the earliest and most influential demonstrations of competitive coevolution was presented by Hillis (1990), who coevolved sorting networks with test cases. A sorting network is a fixed sequence of comparison and swap operations that correctly sorts any input sequence of numbers, regardless of their initial order (Batcher, 1968). The correctness of a sorting network is typically tested against a set of test cases, each being a different input permutation (Batcher, 1968). The problem is particularly challenging because the number of possible test cases grows exponentially with the input size, making exhaustive evaluation computationally infeasible. Many real-world adversarial optimisation problems share this feature: The number of test-cases grows exponentially with the problem size. Moreover, an exhaustive evaluation against a smaller, randomly sampled subset of test cases could still be inefficient because designs are unlikely to be evaluated on rare but critical test cases. Hillis (1990) addressed these challenges by coevolving two populations: one of candidate sorting networks and another of test cases. This setup allowed the test cases to adaptively focus selection pressure on the weaknesses of the sorting networks.

In this framework, the fitness of an individual is not static but depends on its performance against others. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a payoff function, where $\mathcal{X}$ and $\mathcal{Y}$ are the strategy spaces of two interacting populations. The goal in zero-sum settings is often to find the maximin strategy:

$$(x^*, y^*) \in \arg\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} f(x, y),$$

which formalises many adversarial objectives, including those studied in game theory and

GAN training as mentioned in Chapter 1.

It has been conjectured that the success of competitive coevolutionary algorithms (CoEAs) stems from the occurrence of an "arms race". Wiegand (2004) describes an arms race as: as behaviour in a coevolutionary system in which changes in one group or population result in reciprocal, co-adaptive changes in the other groups or populations, and by iterative application of this process, the system produces increasingly better performing individuals by an external measure." Under the competitive framework, such an arms race may emerge during the evolutionary process (Ficici, Pollack, 1998; Rosin, 1997), potentially facilitating the discovery of optimal solutions in adversarial optimisation problems. Additionally, the notion of "arms races" was formalised and extended in several key works (Axelrod, 1987; Ficici, 2004; Rosin, 1997), which positioned competitive coevolution as a promising approach for open-ended learning and adversarial problem solving.

However, CoEAs may also suffer from pathological behaviours resulting from the same dynamics, including disengagement, over-specialisation, and cyclic behaviour[5] (Rosin, 1997; Wiegand, 2004). Although several empirical studies have investigated these phenomena, rigorous theoretical analysis, such as runtime analysis of competitive coevolution, remains largely underdeveloped due to the complexity of the underlying dynamics. These dynamics often arise in practical tasks, such as density classification in cellular automata (Pagie and Hogeweg, 2000), and are frequently attributed to intransitivity in the dominance relation (Watson and Pollack, 2001). We will later provide a simple example illustrating how intransitivity can lead to cyclic or pathological behaviours.

Ficici and Pollack (1998a) studied the dynamics of arms races in three-player competitive games. Their empirical results showed that arms races can facilitate the discovery of optimal strategies, though such dynamics often collapse when one participant becomes significantly stronger than the others. Key theoretical questions remain open, such as how arms races

[5]We will discuss these pathological behaviours in Section 2.7.3.

emerge and how they affect the expected runtime of competitive CoEAs. We address these questions in Chapter 5.

Recent developments in generative adversarial networks (GANs) (Goodfellow et al., 2020) have renewed interest in competitive coevolution. A Generative Adversarial Network (GAN) is a type of machine learning model consisting of two neural networks, a generator net and a discriminator net, that are trained simultaneously in a game-theoretic setting. The generator aims to create realistic data (e.g., images), while the discriminator tries to distinguish between real and generated data, pushing both models to improve over time in a coevolutionary manner. Successful applications of CoEAs to GAN training have been reported (Al-Dujaili et al., 2018; Mitchell, 2006; Toutouh et al., 2019). CoEAs share several characteristics with adversarial models like GANs, including vulnerability to pathological issues such as focusing, relativism, gradient vanishing, and mode collapse (Al-Dujaili et al., 2018; Goodfellow, 2017; Popovici et al., 2012).

To better understand these gradient-based algorithms, researchers have investigated foundational adversarial scenarios such as bilinear zero-sum games (Liang and Stokes, 2019; Mokhtari et al., 2019; Zhang and Yu, 2020). These studies demonstrate that certain gradient-based algorithms achieve linear convergence in differentiable adversarial problems[6]. Apart from gradient-based algorithms, researchers are also interested in less informative algorithms, including black-box continuous optimisation without a gradient. Miyagi et al. (2022) investigated the problem of black-box min-max continuous optimisation and proposed an adaptation of the covariance matrix adaptation evolution strategy (CMA-ES) using a novel worst-case ranking approximation (WRA) mechanism. Their goal was to address two key limitations in existing methods: the degradation in convergence rate due to strong interaction terms (e.g., $\mathbf{x}^\top B\mathbf{y}$ where $\mathbf{x} \in \Delta_{m-1}$[7] is strategy for x-player, $\mathbf{y} \in \Delta_{n-1}$

[6] A bound of $|f(x_k) - f(x^*)| \leq \varepsilon$ can be achieved using only $O(\log(1/\varepsilon))$ iterations, where for $\varepsilon > 0$ and x^* is the optimum. This is called "linear convergence" with rate $O(c^k)$ for $0 < c < 1$.

[7] $\Delta_k := \{(t_0, \ldots, t_k) \in \mathbb{R}^{k+1} \mid \sum_{i=0}^{k} t_i = 1; t_i \geq 0 \text{ for } i \in [k] \cup \{0\}\}$.

is strategy for y-player and $B \in \mathbb{R}^{m \times n}$ is the payoff matrix) and the failure to converge when the objective function lacks Lipschitz smoothness and strong convex-concavity near the optimum. Their empirical results showed that the proposed method was more efficient in terms of function evaluations under strong interaction terms and could converge on functions where existing approaches failed. However, as noted in their work, the method lacks theoretical analysis, particularly regarding its convergence guarantees, and does not provide a thorough empirical study of how function evaluations scale with dimensionality across a broader class of benchmark functions. Furthermore, the extent to which the proposed method consistently converges remains unclear. Finally, discrete black-box optimisation algorithms are less analysed, including CoEAs in discrete search spaces, such as $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, where gradients and continuity are not available. This absence of gradient and continuity information significantly complicates the theoretical analysis. Although Jensen (Jensen, 2001) made early efforts to analyse coevolution, the work focused on cooperative rather than competitive settings. This thesis helps address this gap by providing a theoretical framework (runtime analysis) for competitive coevolution in discrete adversarial optimisation as a reasonable starting point.

2.7.3 Classical Methods to Alleviate Pathological Behaviour

While the idea of arms races in evolution is intuitively appealing, it often results in non-trivial dynamics, especially when caused by pathological behaviours. Rosin and Belew (1996) identified several failure modes in competitive CoEAs, such as: *Disengagement*, where one population becomes too weak to challenge the other; *Over-specialisation*, where individuals exploit narrow strategies that do not generalise well; and *Cyclic behaviour*, where improvements are forgotten because strategies go back to earlier states.

Empirical studies have reported a range of complex dynamics in coevolutionary systems (Ficici, Pollack, 1998; Nolfi and Floreano, 1998; Stanley and Miikkulainen, 2002),

including continual adaptation without clear progress towards the optimum. One explanation for such behaviour is the presence of intransitive dominance relations in the strategy space (Watson and Pollack, 2001), where no globally optimal solution exists—e.g., strategy A may beat B, B may beat C, and yet C may beat A, yielding endless cycles in population rankings. Additionally, some continual adaptations have been described as "Red Queen" dynamics (Cliff and Miller, 1995), where species must continuously evolve just to maintain their relative performance (Solé, 2022; Van Valen, 1973). Importantly, such dynamics are not inherently pathological: they may simply reflect the ongoing competitive pressures in an evolving environment, especially in settings like bilinear games where constant improvement is necessary to stay competitive (Hevia Fajardo et al., 2023; Lehre, 2024).

To overcome the pathological behaviours described above, several methods have been proposed. Notably, Rosin and Belew (1996) introduced a series of techniques for competitive coevolution in the context of game learning:

- *Each Defeats the Last*: The algorithm constructs a chain of strategies by alternately evolving opponents that defeat the previous strategy in the sequence. According to (Rosin and Belew, 1996), this can help escape local optima by explicitly training against increasingly stronger adversaries, albeit potentially reinforcing cyclic behaviours.

- *Single Counterexamples*: This method evaluates a candidate strategy against a challenger, and if defeated, only the specific counterexample is used for further training. While efficient in isolating weaknesses, it does not completely address cyclic behaviour unless counterexamples span the full spectrum of potential opponents.

- *Covering Competitive Algorithms*: This approach evolves strategies that are robust against all previously encountered opponents, encouraging generalisation. In the presence of intransitivity, this method aims to build strategies that can break cycles

by outperforming entire sets of challengers rather than exploiting niche weaknesses.

In follow-up work, Rosin and Belew (1997) proposed mechanisms for promoting effective competition between distinct populations while preserving population diversity. Their techniques which were tested on 3D Tic-Tac-Toe and Nim, include:

- *Competitive Fitness Sharing*: Competitive Fitness Sharing is a method for assigning fitness in coevolutionary settings that rewards individuals (prey, Rosin and Belew (1997) refers to them as hosts) for defeating opponents (predators, Rosin and Belew (1997) refers to them as parasites) that few others can defeat. Unlike simple fitness, which just counts the number of victories, competitive fitness sharing divides the fitness contribution of each defeated predator by the number of prey that also defeated it. According to (Rosin and Belew, 1997), this encourages diversity and discourages convergence on the same solutions by giving higher fitness to prey that beat rare or difficult predators. Essentially, each predator is treated as a limited resource, and fitness is shared among its conquerors. Rosin and Belew (1997) argued that this is particularly useful when no single prey can yet defeat all predators, as it promotes broader exploration of the strategy space.

- *Shared Sampling*: Shared Sampling is a method designed to reduce computational cost in coevolution by evaluating each prey against only a carefully selected subset of predators, rather than the entire opposing population. The selection process uses competitive fitness sharing to prioritise predators that uniquely challenge prey. Predators are chosen such that each new one adds maximum diversity in the challenges posed, i.e., it defeats prey that others in the sample do not. Rosin and Belew (1997) argued that this ensures a representative and diverse subset across all niches in the parasite population, making evaluations both efficient and effective for evolutionary progress.

- *Archive-Based Method and Hall of Fame*: Archive-Based Method is another classical method. An archive consists of individuals used to evaluate new candidates and store promising solutions. It is incrementally updated from the current population (Popovici et al., 2012). According to (De Jong, 2004b), the value of an archive depends on: Generality: The breadth of coevolutionary scenarios in which the archive guarantees progress; Sensitivity: The ability to detect fine-grained improvements in learner quality; Efficiency: Resource use, including computational cost and memory. In particular, Hall of Fame is a memory mechanism in competitive coevolution used to address the limitations of finite populations, which can easily forget past successful individuals (Rosin and Belew, 1997). While an unbounded population size naturally retains strong genotypes through ongoing competition with all previous opponents, finite populations risk losing individuals that could later be critical for evaluating progress or resisting regressions. The Hall of Fame aims to resolve this by storing the best individual from each generation, not for reproduction but for testing. These elite individuals serve as a persistent benchmark: new candidates must not only defeat current opponents but also perform well against the historical best. Rosin and Belew (1997) argued that this discourages overspecialisation and ensures that evolutionary progress is measured against a broader and more challenging set of opponents. Although performance-based sampling of the Hall of Fame was considered, it was found to be computationally expensive. Instead, random sampling from the Hall of Fame is used in practice to maintain a balance between robustness and efficiency.

- *Phantom Parasite*: Phantom Parasite is a mechanism used to prevent stagnation in competitive coevolution when an undefeatable opponent dominates, leaving the other population with no useful fitness feedback. It introduces an artificial opponent, not actually played against, to reward partially capable individuals. In this setup, prey that lose to any current parasite are treated as defeating the phantom parasite, while those that beat all current predators are treated as losing to it. Rosin and Belew

(1997) argued that this design encourages the survival of individuals that can defeat easier, pedagogically useful predators, guiding selective pressure toward overcoming harder opponents. By reintroducing fitness diversity and maintaining pressure on the dominant population, the phantom parasite helps sustain evolutionary progress when coevolution risks collapsing.

These methods were empirically shown to accelerate the performance of CoEAs on Nim and 3D-Tic-Tac-Toe by preserving high-quality individuals and avoiding premature convergence. As claimed in (Rosin and Belew, 1997), the combination of multiple mechanisms, such as Fitness Sharing and Hall of Fame, was essential; omitting any of them led to significantly increased runtimes. This suggests a potential avenue for runtime analysis of individual components.

Additionally, Rosin and Belew (1997) introduced the infinite population perspective. Assuming a continuous population distribution (e.g., a fraction $p \in [0, 1]$ playing strategy s_1, and $1 - p$ playing s_2), they applied the replicator equation from Evolutionary Game Theory to analyse algorithmic dynamics (Rosin and Belew, 1997). However, this model assumes no genotype extinction due to sampling error, which is a significant limitation. Tino et al. (2013) further criticised the infinite population approximation by showing that real evolutionary trajectories cannot be reliably approximated within infinite and simulated population models. To retain key properties of coevolution under finite populations, Rosin and Belew (1996) proposed several mechanisms outlined above.

De Jong (2004b) proposed the *Incremental Pareto-Coevolution Archive (IPCA)* Method, which is designed to ensure convergence in general CoEAs, albeit with a higher computational cost. Consider two populations: predators (learners in (De Jong, 2004b)) ($P \in \mathcal{X}^\lambda$) and prey (tests in (De Jong, 2004b)) ($Q \in \mathcal{Y}^\lambda$), evaluated by an outcome function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ assigning performance scores $f(x, y)$, the performance of x when playing against y. This approach guarantees the progress with respect to certain solution

concepts in (De Jong, 2004b).

Another critical approach in competitive CoEAs is the *Ideal Evaluation Method*. Ideal Evaluation in the context of coevolution refers to a method of evaluating individuals based on all underlying objectives of a test-based problem, rather than using potentially biased or incomplete approximations (De Jong and Pollack, 2004). De Jong and Pollack (2004) argued that precise evaluation improves coevolutionary performance, as seen in earlier works (Hillis, 1990; Juillé and Pollack, 1996; Pollack, 1998; Sims, 1994). However, due to population size limitations, exact evaluations are often impractical. One of the possible solutions is to approximate its exact evaluation. Two benchmark problems are considered:

Definition 2.7.2 (Compare-on-all)**.**

$$g_{all}(x,y) = \begin{cases} 1 & \text{if } \forall i \in \mathbb{N} : x_i \geq y_i \\ -1 & \text{otherwise} \end{cases}$$

Here, x is a prey (learner) and y a predator (evaluator), with x_i representing performance in dimension i.

Definition 2.7.3 (Compare-on-one)**.** *Let* $m = \arg\max_i y_i$*. Then:*

$$g_{one}(x,y) = \begin{cases} 1 & \text{if } x_m \geq y_m \\ -1 & \text{otherwise} \end{cases}$$

De Jong and Pollack (2004) showed that a limited test set can achieve the same effect as using all opponents. Based on this insight and competitive fitness sharing (Rosin and Belew, 1997), the DELPHI algorithm was proposed. Empirical results show that it outperforms other methods such as Competitive Fitness Sharing and its variants on both Compare-on-one and Compare-on-all problems.

2.7.4 Free Lunch Theorem in CoEAs

Wolpert and Macready (1997) revealed fundamental insights into the limitations of traditional black-box optimisation algorithms, including various randomised search heuristics (such as evolutionary algorithms and simulated annealing) and machine learning algorithms (such as supervised learning). They showed that the performance of all black-box optimisation algorithms, when averaged over all possible problem instances, is identical for both maximisation and minimisation tasks. Droste et al. (2002b) extended the original No Free Lunch (NFL) theorem to more realistic scenarios by relaxing the assumption that it holds for all problem instances to only those closed under permutation. Later, Igel and Toussaint (2005) showed another NFL theorem for non-uniform distributions of target functions.

Recall from Section 2.3, Nash Equilibrium is a widely studied solution concept in adversarial learning and maximin optimisation. It plays a central role in various applications, including adversarial models such as GANs (Goodfellow et al., 2014; Heusel et al., 2017) and spatial games (Hemberg et al., 2021; Lehre, 2022; Lehre et al., 2023).

In the context of adversarial optimisation, a 'free lunch' refers to the existence of algorithms that outperform others[8], when averaged across all possible instances, under a given solution concept. Wolpert and Macready (2005) demonstrated such a phenomenon for the maximin solution concept, while Service and Tauritz (2008) established a similar result for the 'maximisation over all test cases' concept. Although the above studies confirmed the existence of 'free lunches' under certain solution concepts, two important questions remain:

(1) Does adversarial optimisation exhibit a 'free lunch' for all possible solution concepts?

(2) If Nash Equilibrium is adopted as the solution concept, how can we characterise the difficulty of black-box adversarial optimisation problems for problem-independent, possibly randomised, search heuristics?

[8]In the context of NFL, one compares average case expected runtime/number of queries.

These will be covered in depth in Chapter 4.

2.7.5 Runtime Analysis of Competitive CoEAs

We argue that runtime analysis offers valuable insights into the behaviour of coevolutionary algorithms (CoEAs). As a foundational tool in the analysis of traditional evolutionary algorithms (EAs), runtime analysis provides theoretical bounds on the number of fitness function evaluations, commonly referred to as the runtime (Doerr and Neumann, 2020). This approach has proven instrumental in identifying relationships between algorithmic parameters and problem characteristics that influence the performance of EAs. In this thesis, we add to the existing runtime analysis of CoEAs.

Despite its importance, rigorous runtime analysis of CoEAs remains under-explored (Popovici et al., 2012). Jansen and Wiegand (2004) analysed the runtime of a cooperative CoEA and a (1+1) EA on *separable* functions. Their results demonstrated that problem separability does not reliably predict whether a cooperative CoEA will outperform the (1+1) EA.

The first rigorous runtime analysis of a competitive, population-based CoEA was conducted by Lehre (2022). Lehre (2022) introduced a novel level-based theorem for general coevolutionary processes (see Section 2.8.2.2) and theoretically analysed a population-based algorithm called PDCoEA, demonstrating that with suitable parameter settings, it can efficiently compute an ε-approximation of a Nash equilibrium for certain pseudo-Boolean Bilinear problems. Moreover, the analysis showed that inappropriate parameter settings render the algorithm inefficient for all problems (i.e. error threshold[9] for mutation rate). Besides these, Fajardo and Lehre (2024), Hevia Fajardo and Lehre (2023) and Hevia Fajardo et al. (2023) explored furthermore in the runtime analysis of competitive CoEAs, including

[9]Error threshold is a tolerance limit of mutation rate. If the mutation rate is higher than the value of the error threshold, then the algorithm might take exponential time to reach the optimum (Lehre, 2010).

RLS-PD (a type of random local search CoEAs), different fitness aggregation methods and ranking diversity CoEAs on BILINEAR. Moreover, Benford and Lehre (2024a) investigated competitive CoEAs on symmetric zero-sum games and impartial combinatorial games.

As the first runtime analysis for CoEAs on these games, these results serve as a critical step towards a comprehensive theoretical framework for coevolution. A crucial gap in the runtime analysis of CoEAs is the lack of understanding when the payoff in two-player zero-sum games is binary, i.e., either a win or a loss. This thesis will mainly focus on binary games.

2.8 Runtime Analysis and Traditional Tools

Runtime is an important performance measure for black-box optimisation algorithms, including evolutionary algorithms. The runtime of an EA on a problem instance is usually defined in terms of the number of objective function evaluations required by the algorithm to find an optimal solution for the first time (Doerr and Neumann, 2020; Droste et al., 2006; Lehre, 2012). Runtime analysis is a popular theoretical approach for mathematically evaluating the performance of EAs and other randomised heuristic algorithms (Doerr and Neumann, 2020). Many powerful mathematical techniques for runtime analysis of EAs have been developed. This section introduces some basic techniques, including drift theorems and level-based theorems.

In optimisation, runtime analysis helps us understand how quickly an algorithm can find an optimal or satisfactory solution. It provides insights into the efficiency, robustness, and scalability of algorithms. Moreover, it enables us to compare different algorithms theoretically and determine how specific hyperparameters or design choices influence the performance. Runtime analysis not only provides precise performance guarantees and guidelines for practitioners to design randomised algorithms but also enhances the ex-

plainability of heuristic search and black-box optimisation algorithms, including NSGA-II (Non-dominated Sorting Genetic Algorithm II), MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition) and EDAs (Estimation-of-Distribution Algorithms) (Dang et al., 2019b; Do et al., 2023; Doerr and Qu, 2023; Witt, 2019; Zheng et al., 2022). For readers interested in a broader theoretical foundation of evolutionary computation beyond the scope of this thesis, we recommend the works authored or edited by Doerr and Neumann (2020), Jansen (2013b) and Neumann and Witt (2010a).

We first provide formal definitions of the runtime of black-box algorithms for traditional optimisation and adversarial optimisation. We present an unrestricted black-box model of traditional optimisation. The following definition assumes maximisation of functions. Analogous definitions can be made for minimisation.

Definition 2.8.1 (Runtime for single-objective black-box optimisation (Babichenko, 2020; Droste et al., 2006))**.** *The runtime T of a black-box optimisation algorithm A on fitness function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a finite search space and $x_t \in \mathcal{X}$ is the search point at the t-th evaluation for $t \in \mathbb{N}$, is*

$$T_{A,f} := \min \{t \in \mathbb{N} \mid \forall y \in \mathcal{X}, f(x_t) \geq f(y)\}. \tag{2.5}$$

Next, we construct an unrestricted black-box model of adversarial optimisation.

Algorithm 9 defines a class of algorithms subject to various probability distributions and samples the new strategy pair based on previous pairs and their payoffs. The initial search point (x_0, y_0) is independent of the problem, so we can choose any probability distribution $P_\emptyset$ to initialise the algorithm. Subsequent strategy pairs are obtained by sampling subject to the probability distribution $P_{I(t)}$. By specifying sample probability distribution $P_{I(t)}$, Algorithm 9 represents various black-box optimisation algorithms, including several adversarial search (also called competitive coevolutionary) algorithms (Lehre et al., 2023; Wolpert and Macready, 2005) and randomised algorithms such as FindPsne (designed to

Algorithm 4 An Unrestricted Black-Box Model with Unique Query History adapted from (Babichenko, 2016; Lehre and Lin, 2024b)

Require: Search spaces $\mathcal{X}, \mathcal{Y}$.

Require: Payoff functions $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$;

1: Initialise (x_0, y_0); Initialise $H_0 = \emptyset$ and $C_0 = 0$.

2: **for** $t = 1, 2, \cdots$ until the termination criterion met **do**

3: Choose some probability distribution $P_{I(t)}$, depending only on $I(t)$ where $I(t) := \Pi_{j=1}^{t-1}(x_j, y_j, g(x_j, y_j), h(x_j, y_j)) \in (\mathcal{X} \times \mathcal{Y} \times \mathbb{R} \times \mathbb{R})^{t-1}$

4: Sample a random search point (x_t, y_t) according to distribution $P_{I(t)}$.

5: Query the payoffs $g(x_t, y_t), h(x_t, y_t)$

6: **if** $(x_t, y_t) \notin H_{t-1}$ **then** $C_t = C_{t-1} + 1$; $H_t = H_{t-1} \cup \{(x_t, y_t)\}$.

7: **else** $C_t = C_{t-1}$; $H_t = H_{t-1}$.

learn the NE in bimatrix games) (Maiti et al., 2023). Note that the model only considers the cost of unique queries made by the algorithm C_t (i.e. we check the search point in the previous query history H_t in line 6). We assume that an algorithm which makes the same query twice is only charged for the cost of one of the queries.

Definition 2.8.2 (Runtime for adversarial optimisation)**.** *Given a test-based optimisation problem $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, g, P^+)$ where $P^+ \subseteq \mathcal{X} \times \mathcal{Y}$ and $P^+ \neq \emptyset$, the runtime T of a black-box adversarial optimisation algorithm A on an interaction function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $\mathcal{X} \times \mathcal{Y}$ is a finite search space and $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ is the search point at the t-th evaluation for $t \in \mathbb{N}$, is*

$$T_{A,\mathcal{G}} := \min \left\{ t \in \mathbb{N} \mid (x_t, y_t) \in P^+ \right\}. \tag{2.6}$$

As we define runtime, according to (Lehre and Oliveto, 2023), runtime analysis means that we show how $\mathrm{E}(T)$ - expected runtime for first time reaching some target set; and $\Pr(T \leq t)$ - the probability that the optimum is firstly found within t steps depend on algorithms and problems.

2.8.1 Drift Theorems

In the theoretical analysis of evolutionary computation and other randomised search heuristics, most runtime results are derived using a small number of powerful techniques (Doerr and Neumann, 2020; Lengler, 2018). Among these, drift analysis is one of the most widely used. Drift theorems provide a method for estimating the expected runtime, as well as tail bounds on the runtime distribution.

Drift theory is a general term for a collection of theorems that, under certain conditions, yield runtime guarantees of algorithms. The additive drift theorem was first introduced by He and Yao (2001), though the result traces back to Hajek (1982). As Krejca (2019) and Lengler (2018) note, similar results had been proved even earlier in other domains. Several variants have been developed. A stochastic process is a sequence of random variables[10]. Generally speaking, we consider an underlying stochastic process Y_t induced by the dynamics of algorithms over the search space S, and a potential function $X_t := V(Y_t)$, which describes how far the current search point Y_t is from any optima x^*. In the additive drift setting (He and Yao, 2001), the expected drift of current potential (i.e., $\mathrm{E}(X_t - X_{t+1}) \mid \mathcal{F}_t)$) is bounded by a constant; in the multiplicative case, it is proportional to the current potential from the target (Doerr et al., 2010); and in the variable drift setting (Johannsen, 2010; Mitavskiy et al., 2009), the bound is given by a monotone function of the current state x_t.

Intuitively, a drift theorem transforms the expected drift of a stochastic process at a given state into the expected time needed to reach a target. More precisely, if we can calculate the expected drift (or step progress) of a process towards the optimum at each step, we can derive bounds on how long the process will take to reach the optimum in expectation.

In this thesis, we use a particular formulation of drift theorems to derive expected runtime

[10]We define a stochastic process in Appendix A.1 (See Definition A.1.8).

bounds for coevolutionary and bandit-based algorithms, presented as follows.

Theorem 2.8.1 (Additive Drift Theorem (Doerr and Neumann, 2019; He and Yao, 2001))**.** *Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables with a finite state space $\mathcal{S} \subseteq \mathbb{R}_0^+$ such that $0 \in \mathcal{S}$. Let $T := \inf\{t \geq 0 \mid X_t = 0\}$.*

(1) If there exists $\delta > 0$ such that for all $s \in \mathcal{S} \setminus \{0\}$ and for all $t \geq 0$,

$$\mathrm{E}(X_t - X_{t+1} - \delta; X_t = s \mid \mathcal{F}_t) \geq 0,$$

then $\mathrm{E}(T) \leq \mathrm{E}(X_0)/\delta$.

(2) If there exists $\delta > 0$ such that for all $s \in \mathcal{S} \setminus \{0\}$ and for all $t \geq 0$,

$$\mathrm{E}(X_t - X_{t+1} - \delta; X_t = s \mid \mathcal{F}_t) \leq 0,$$

then $\mathrm{E}(T) \geq \mathrm{E}(X_0)/\delta$.

Theorem 2.8.2 (Negative Drift Theorem (Doerr and Neumann, 2019; Oliveto and Witt, 2012; Rowe and Sudholt, 2012))**.** *For constants $a, b, \delta, \eta, r > 0$, with $a < b$, there exist $c > 0$, $n_0 \in \mathbb{N}$ such that the following holds for all $n \geq n_0$. Suppose $(X_t)_{t\geq 0}$ is a sequence of random variables with a finite state space $S \subset \mathbb{R}_0^+$ and with associated filtration $\mathcal{F}_t$. Assume $X_0 \geq bn$, and let $T_a := \min\{t \geq 0 \mid X_t \leq an\}$ be the hitting time of $\mathcal{S} \cap [0, an]$. Assume further that for all $s \in \mathcal{S}$ with $s > an$, for all $j \in \mathbb{N}_0$, and for all $t \geq 0$, the following conditions hold:*

(1) $\mathrm{E}(X_t - X_{t+1} + \delta; X_t = s \mid \mathcal{F}_t) \leq 0$

(2) $\Pr[|X_t - X_{t+1}| \geq j; X_t = s \mid \mathcal{F}_t] \leq \frac{r}{(1+\eta)^j}$

Then, $\Pr[T_a \leq e^{cn}] \leq e^{-cn}$.

2.8.2 Level-Based Theorems

2.8.2.1 Original Level-Based Theorem

Many population-based algorithms are difficult to analyse due to their stochastic nature and reliance on sampling rather than strict selection. The level-based theorem provides a general framework for estimating the expected runtime of such algorithms (Corus et al., 2018; Dang et al., 2019a; Lehre and Nguyen, 2021; Wegener, 2002).

The core idea is to partition the search space $\mathcal{X}$ into a sequence of disjoint subsets, called *levels*, denoted $A_1, A_2, \ldots, A_m$ (Corus et al., 2018; Dang et al., 2021a). Often, the levels will be defined so that the fitness is monotonically increasing with respect to the level ordering. We refer to the *current level* as the maximum level such that it contains a certain fraction of the population $\gamma_0\lambda$. The theorem then analyses how the population evolves via these levels over time. The theorem formalises this idea using three sufficient conditions (See Theorem 2.8.3 for the exact statement). Informally, Condition (G1) ensures that there is always a non-zero probability of generating an individual in a higher level; Condition (G2) ensures that the number of individuals in the current level keeps expanding; and Condition (G3) requires the population size to be sufficiently large to support this expected progress.

When these conditions are satisfied, the theorem provides an upper bound on the expected number of generations until at least one individual in the population reaches the final level A_m, which typically is defined to be the set of optimal solutions. This framework has been widely applied in the runtime analysis of population-based evolutionary algorithms (Case and Lehre, 2020a; Corus et al., 2018; Dang et al., 2021b).

Theorem 2.8.3 (Level-based theorem (Corus et al., 2018))**.** *Given a partition $(A_1, \ldots, A_m)$ of $\mathcal{X}$ and a population P_t in Algorithm 3, let $T := \min\{t\lambda \mid |P_t \cap A_m| > 0\}$ be the first point in time that the elements of A_m appear in P_t of Algorithm 3. Let $A_{\geq j} := \cup_{i=j}^{m} A_i$. 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*

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

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

2.8.2.2 Level-Based Theorem for Coevolution

The traditional level-based theorem is used in single-objective optimisation. To take adversarial interaction into account, we need another tool to cope with payoff interaction. Lehre (2024) extended the original level-based theorem as follows. Informally, Lehre (2024) partitions the strategies space into $A_1 \times B_1, \cdots, A_m \times B_m$, allowing overlaps between each level. Then, by showing the population satisfying (G1) there exists a non-zero probability of generating an individual in both next levels A_{j+1} and B_{j+1}; (G2a) and (G2b) ensure the expanding probability in the current and next level; (G3) guarantee a sufficiently large population size, we can bound the runtime of Algorithm 5 by polynomial runtime in terms of population size λ and number of level m.

Algorithm 5 requires two populations P, Q. At each iteration, a new pair of solutions is sampled from P, Q via the variation operators, including selection and mutation operators. The population is then updated with this new pair of solutions, and the process continues until the termination criteria are met. Additionally, we use x_t and y_t to denote the search point generated by coevolutionary algorithms at iteration $t \in \mathbb{N}$. These stochastic processes

Algorithm 5 coevolutionary Process (Lehre, 2024)

Require: Population size $\lambda \in \mathbb{N}$ and strategy spaces $\mathcal{X}$ and $\mathcal{Y}$.

Require: Initial populations $P_0 \in \mathcal{X}^\lambda$ and $Q_0 \in \mathcal{Y}^\lambda$.

1: **for** each generation number $t \in \mathbb{N}_0$ **do**

2: **for** each interaction number $i \in [\lambda]$ **do**

3: Sample an interaction $(x, y) \sim \mathcal{D}(P_t, Q_t)$.

4: Set $P_{t+1}(i) := x$ and $Q_{t+1}(i) := y$.

track the dynamics of the coevolutionary algorithms, and we will analyse their first hitting time on the target set (i.e., their runtime).

Theorem 2.8.4 (Level-based theorem for coevolution (Lehre, 2024))**.** *Given subsets $A_j \subseteq \mathcal{X}$ and $B_j \subseteq \mathcal{Y}$ for $j \in [m]$ with $A_1 := \mathcal{X}$ and $B_1 := \mathcal{Y}$, let*

$$T := \min \{t\lambda \mid (P_t \times Q_t) \cap (A_m \times B_m) \neq \emptyset\}$$

be the first time step when a pair in $A_m \times B_m$ appears in the population pair (P_t, Q_t) of Algorithm 5 at generation t, where $P_t \in \mathcal{X}^\lambda$ and $Q_t \in \mathcal{Y}^\lambda$.

Assume that there exist constants $z_1, \ldots, z_{m-1} > 0$, $\delta \in (0, 1]$, and $\gamma_0 \in (0, 1)$ such that for any populations $P \in \mathcal{X}^\lambda$, $Q \in \mathcal{Y}^\lambda$, we define the current level

$$j := \max \left\{ i \in [m] \mid |(P \times Q) \cap (A_i \times B_i)| \geq \gamma_0 \lambda^2 \right\}.$$

(G1) For all $j \in [m-1]$, if $(x, y) \sim \mathcal{D}(P, Q)$ then

$$\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq z_j,$$

(G2a) For all $\gamma \in (0, \gamma_0)$, if $j \in [m-2]$ and $|(P \times Q) \cap (A_{j+1} \times B_{j+1})| \geq \gamma\lambda^2$, then for $(x, y) \sim \mathcal{D}(P, Q)$,

$$\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq (1 + \delta)\gamma,$$

(G2b) For all $j \in [m-1]$*, if* $(x, y) \sim \mathcal{D}(P, Q)$*, then*

$$\Pr(x \in A_j) \Pr(y \in B_j) \geq (1+\delta)\gamma_0,$$

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

$$\lambda \geq 2\left(\frac{1}{\gamma_0 \delta^2}\right)^{1+\nu} \ln\left(\frac{m}{z_*}\right), \quad \text{where } z_* := \min_{i \in [m-1]} z_i \text{ and } \nu > 0,$$

then for any constant $c'' > 1$ *and sufficiently large* λ*, the expected time until reaching* $A_m \times B_m$ *satisfies*

$$\mathbb{E}[T] \leq \frac{c''\lambda}{\delta}\left(m\lambda^2 + 16\sum_{i=1}^{m-1}\frac{1}{z_i}\right).$$

2.8.3 Negative Drift Theorems for Populations

While upper bounds on runtime are a central focus in the analysis of population-based evolutionary algorithms, it is important to understand situations where these population-based algorithms (Algorithm 6) are inefficient, specifically, when the expected runtime becomes exponentially large. The negative drift theorem for populations (Lehre, 2010) provides a rigorous tool to identify such scenarios. It is particularly relevant for population-based EAs over a finite state space, such as those described by Algorithm 6.

Intuitively, the negative drift theorem quantifies how a population may consistently drift away from the optimum due to mutation, selection pressure, and stochastic sampling. When selection is too weak or mutation is too strong, the population may struggle to retain high-quality individuals, leading to stagnation or regression. The theorem formally bounds the probability of reaching a desirable state, such as an optimal solution, within an exponential time, and shows that this probability can be exponentially small under certain conditions.

Theorem 2.8.5 (Negative Drift Theorem for Populations (Lehre, 2010))**.** *Given Algorithm 6 on* $\mathcal{X} = \{0,1\}^n$ *with population size* $\lambda \in$ *poly*(n) *and transition matrix* p_{mut}

Algorithm 6 Population Selection-Variation Algorithm (Lehre, 2010)

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

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

Require: Mutation transition matrix $p_{\text{mut}} : \mathcal{X} \to \mathcal{X}$.

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

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

2: **for** $i = 1$ to λ **do**

3: Choose a parent index $I_t(i) \in \{1, \ldots, \lambda\}$, and set $x := P_t(I_t(i))$.

4: Sample $x' \sim p_{\text{mut}}(x)$, and set $P_{t+1}(i) := x'$.

corresponding to flipping each bit independently with probability χ/n. Let $a(n)$ and $b(n)$ be positive integers such that $b(n) \leq n/\chi$ and $d(n) = b(n) - a(n) = \omega(\ln n)$. For any $x^* \in \{0,1\}^n$, let $T(n)$ be the smallest $t \geq 0$ such that $\min_{j \in [\lambda]} H(P_j(t), x^*) \leq a(n)$, where H is the Hamming distance. Let $R_t(i) := \sum_{j=1}^{\lambda} \mathbf{1}_{\{I_t(j)=i\}}$. If there are constants $\alpha_0 \geq 1$ and $\delta > 0$ such that

(i) $\mathrm{E}(R_t(i) \mid a(n) < H(P_t(i), x^*) < b(n)) \leq \alpha_0$ *for all* $i \in [\lambda]$,

(ii) $\psi := (\ln(\alpha_0)/\chi + \delta) < 1$, *and*

(iii) $b(n)/n < \min\left\{\frac{1}{5}, \frac{1}{2} - \frac{1}{2}\sqrt{\psi(2-\psi)}\right\}$,

then $\Pr\left(T(n) \leq e^{cd(n)}\right) \leq e^{-\Omega(d(n))}$ *for some constant* $c > 0$.

Chapter Three

New Theoretical Tools for Coevolution

This chapter is based on the following publications:
Runtime Analysis of a Coevolutionary Algorithm: Overcoming Negative Drift in Maximin-Optimisation (Hevia Fajardo et al., 2023), and *Concentration Tail-Bound Analysis of Coevolutionary and Bandit Learning Algorithms* (Lehre and Lin, 2024a), which are published in Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA'23) and Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI'24).

3.1 Introduction

Drift analysis is a powerful technique in understanding the performance of randomised algorithms, particularly in the field of runtime analysis of heuristic search. This introductory overview draws on key resources in the field (Doerr and Neumann, 2019; Hajek, 1982; He and Yao, 2001; Jansen, 2013a; Neumann and Witt, 2010b). The majority of existing drift theorems provide an upper bound on the expected runtime needed to reach a target state, such as an optimal solution set (He and Yao, 2001). By identifying an appropriate potential function and demonstrating a positive drift towards the target state, the expected runtime can be bounded by the reciprocal of the drift multiplied by the maximum distance from the target state.

A tail bound gives different information about the runtime distribution than the expectation. Both are useful. The focus of concentration tail-bound analysis is on quantifying the deviation of the runtime of a randomised algorithm T, from its expected value. By providing insights into the distribution of T, this approach offers a more detailed understanding of an algorithm's performance (Doerr and Goldberg, 2013; Kötzing, 2016; Lehre and Witt, 2021). Concentration tail-bound analysis has gained significant interest due to its potential for delivering tighter upper bounds on the runtime of various algorithms. For instance, an exponential tail bound is used to bound the expected runtime of RLS on separable functions (Doerr et al., 2013). Moreover, in the case of (1+1)-cooperative coevolutionary algorithms, concentration tail-bound analysis can establish a $\Theta(n \log(n))$ bound (Lehre and Lin, 2023), where a previous analysis (Jansen and Wiegand, 2004) using expectation only led to an $O(n \log^2 n)$ bound. The concentration tail-bound analysis is also useful in the context of restarting arguments, for example, see (Case and Lehre, 2020b). This more precise runtime estimation can be valuable in optimising and comparing different algorithms, potentially leading to improved algorithm design and performance (Bian et al., 2020; Dang et al., 2021a; Doerr and Qu, 2023; Zheng et al., 2022).

Concentration tail bound is not only used in runtime analysis to help us understand randomised algorithms, including evolutionary algorithms, but can also be used in regret analysis of reinforcement learning algorithms. A typical example is the use of concentration inequalities (such as the Azuma-Hoeffding inequality) to provide precise bounds for regret. For instance, concentration inequalities are employed in the development of optimal UCB family algorithms, which incorporate the concept of optimism in the face of uncertainty (Auer et al., 2002).

The chapter is structured as follows: Section 3.2 provides a general variance drift theorem for analysing CoEAs when coming across negative drift before the equilibrium. In Section 3.3, we consider a tail-bound version of this variance drift theorem. Next, we present some applications of the above new variance theorems in Random 2-SAT in Section 3.4, Coevolutionary Algorithms in Section 3.5 and Bandit Algorithm in Section 3.6. We complete our theoretical analysis with experiments in Section 3.7. Finally, Section 3.8 concludes the chapter.

3.2 A New Drift Theorem: Overcoming Negative Drift

Most drift theorems require the process to have, in expectation a positive drift. However, sometimes the processes studied have a small negative drift, pushing the process away from the target in expectation. In the following theorem, we present a transformation for such a process where the variance of the process can be used to counteract the small negative drift and allows us to apply drift theorems to show that the target is reached efficiently despite the negative drift. Doerr and Kötzing (2021) use a similar idea to tackle the case of low drift. Compared with (Doerr and Kötzing, 2021), we use a quadratic transformation instead of log-transformation to allow the $\Omega\left(\frac{1}{n}\right)$ negative drift before reaching the optimum.

Theorem 3.2.1 (Variance drift transformation)**.** *Let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random*

variables with a finite state space $\mathcal{S} \subseteq \mathbb{R}$ *adapted to a filtration* $(\mathcal{F}_t)_{t\in\mathbb{N}}$*, and let* $T = \inf\{t \mid X_t \leq 0\}$*. Furthermore, suppose that, given* $b > 0$*,*

(A_1) *there exist* $\delta > 0$ *such that for all* $t \in \mathbb{N}$*, it holds that*

$$\mathrm{E}_t\left((X_{t+1} - X_t)^2 - 2(X_{t+1} - X_t)(b - X_t) - \delta; t < T\right) \geq 0$$

Then, given that $Y_t = b^2 - (b - X_t)^2$ *for all* $t \in \mathbb{N}$

$$\mathrm{E}_t(Y_t - Y_{t+1} - \delta; t < T) \geq 0.$$

Proof. We define the process $Y_t = b^2 - (b - X_t)^2$ for all $t \in \mathbb{N}$. For all $t < T$ we determine the drift of Y_t:

$$\begin{aligned}
\mathrm{E}_t(Y_t - Y_{t+1}; t < T) &= \mathrm{E}_t\big((b - X_{t+1})^2 - (b - X_t)^2; t < T\big) \\
&= \mathrm{E}_t\big(b^2 - 2bX_{t+1} + X_{t+1}^2 - (b^2 - 2bX_t + X_t^2); t < T\big) \\
&= \mathrm{E}_t\big(X_{t+1}^2 - X_t^2 - 2b(X_{t+1} - X_t); t < T\big) \\
&= \mathrm{E}_t(X_{t+1}^2 - 2X_{t+1}X_t + X_t^2 - 2X_t^2 + 2X_{t+1}X_t - 2b(X_{t+1} - X_t); t < T) \\
&= \mathrm{E}_t((X_{t+1} - X_t)^2 - 2X_t^2 + 2X_{t+1}X_t - 2b(X_{t+1} - X_t); t < T) \\
&= \mathrm{E}_t((X_{t+1} - X_t)^2 + 2X_t(X_{t+1} - X_t) - 2b(X_{t+1} - X_t); t < T) \\
&= \mathrm{E}_t\big((X_{t+1} - X_t)^2 - 2(X_{t+1} - X_t)(b - X_t); t < T\big) \\
&\geq \delta.
\end{aligned}$$

The final inequality holds since we use Condition (A_1). □

Corollary 3.2.1 (Variance upper additive drift)**.** *Let* $a \leq 0$ *and* $b > 0$*. Let* $(X_t)_{t\in\mathbb{N}}$ *be a sequence of random variables with a finite space state* $\mathcal{S} \subseteq \mathbb{R}$ *adapted to a filtration* $(\mathcal{F}_t)_{t\in\mathbb{N}}$*, and let* $T = \inf\{t \mid X_t = 0\}$*. Furthermore, suppose that, given* $b > 0$*,*

(A_1) *there exist* $\delta > 0$ *such that for all* $t \in \mathbb{N}$*, it holds that*

$$\mathrm{E}_t\left((X_{t+1} - X_t)^2 - 2(X_{t+1} - X_t)(b - X_t) - \delta; t < T\right) \geq 0$$

(A_2) *and for all* $t \in \mathbb{N}$, *it holds that* $a \leq X_t \leq b$

Then,

$$\mathrm{E}_0(T) \leq \frac{\mathrm{E}_0(b - X_T)^2 - (b - X_0)^2}{\delta}.$$

Proof. We define the process $Y_t = b^2 - (b - X_t)^2$ for all $t \in \mathbb{N}$ again. Notice that $Y_t = 0$ is equivalent to $X_t = 0$ by using A_2 condition. Then

$$T = \inf\{t \mid X_t = 0\} = \inf\{t \mid Y_t = 0\}.$$

We apply a version of the additive drift theorem from (Kötzing and Krejca, 2019, Theorem 7) on the stochastic process Y_t and get the claimed upper bounds for the expected runtime. □

3.3 A Tail-Bound for a New Drift Theorem

With the development of runtime analysis, researchers have established several concentration tail bounds for EAs. For example, Lehre and Witt (2021) provides an exponential tail-bound for the basic (1+1)-EAs on OneMax functions, which is a well-studied benchmark function to analyse the performance of EAs. To the best of our knowledge, the current best general tail bounds for both processes under the additive drift and variance-dominated processes can be found in (Kötzing, 2016; Lehre and Witt, 2021). Kötzing (2016) shows that the runtime is at most quadratic in n with probability $1 - p$ for any $p > 0$. If we replace $1/p^{\ell \log(c)}$ by $r > 0$ and rewrite it in terms of an upper tail bound, then the original bound becomes that given two constants $1 \leq c < n, \ell > 0$ and for any $r > 0$,

$$\Pr\left(T \geq rn^2\right) \leq r^{-\frac{1}{\ell \log(c)}}. \tag{3.1}$$

Although we have established a tail bound for variance-dominated processes that concentrate on the expectation in a polynomial order with respect to r in Equation (3.1), it is

worth exploring whether a more precise concentration tail bound can be derived for such processes, such as an exponential tail. A sharper exponential tail bound can improve the expected runtime estimation and thus provide useful insights into randomised algorithms.

We define the k-th stopping time, also called k-th hitting time, which will be used in later proofs.

Definition 3.3.1. *(k-th stopping time) Given a stochastic process $(X_t)_{t\geq 0}$ on a state space in $\mathbb{R}$. Let the target set A be a finite non-empty subset of $\mathbb{R}$, and then for any $k \geq 0$, we define $T_k = \min\{t \geq k \mid X_t \in A\}$. In particular, T_0 is the first hitting time at A.*

We first provide a formal definition of variance-dominated and variance-transformed processes.

Definition 3.3.2. *(Variance-dominated processes) A sequence of random variables $X_0, X_1, \ldots \in [0, n]$ is a variance-dominated process with respect to the filtration $\mathcal{F}_0, \mathcal{F}_1, \ldots$ if for all $t \in \mathbb{N}$, the following conditions hold:*

(1) $\mathrm{E}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq 0$;

(2) $\exists \delta > 0$ *such that* $\mathrm{E}((X_{t+1} - X_t)^2 \mid \mathcal{F}_t) \geq \delta$.

Definition 3.3.3. *(Variance-transformed processes) A sequence of random variables $X_0, X_1, \ldots \in [0, n]$ is a variance-transformed process with respect to the filtration $\mathcal{F}_0, \mathcal{F}_1, \ldots$ if for all $t \in \mathbb{N}$, the following conditions hold:*

(1) $0 > \mathrm{E}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq -\frac{c}{n}$;

(2) $\exists \delta > 0$ *such that* $\mathrm{E}((X_{t+1} - X_t)^2 \mid \mathcal{F}_t) \geq \delta$.

This section mainly focuses on random processes which consist of positive, weak (almost zero) or even a small negative drift with a constant second moment since these processes

exhibit more complicated dynamics (Doerr and Zheng, 2020; Friedrich et al., 2016; Göbel et al., 2022; Kötzing et al., 2015). A general polynomial tail bound is provided for these in (Kötzing, 2016), but any general exponential tail bounds for these processes are still missing.

In the following sections, we exploit the Optional Stopping Time theorem (Theorem A.3.2 in Appendix A.3) to obtain our exponential tail bound. This theorem is crucial for proving the original additive drift theorem, as highlighted by He and Yao, 2001. This recurrence method can provide a different perspective to derive the exponential tail bound in runtime analysis.

3.3.1 A Recurrent Method in Upper Tail Bound

Next, we explore how to derive a general framework for providing exponential tail bounds for randomised algorithms, including evolutionary algorithms which satisfy certain conditions. We explore the exit time of X_t out of some interval $[0, b]$, using the same set-up as (Kötzing, 2016). In the proof of McDiarmid's inequality as in (McDiarmid, 1989) also uses the Hoeffding lemma and conditions on the past events to establish the recurrence. (Doerr and Goldberg, 2013) uses the multiplicative drift condition directly to build up the exponential recurrence relation and hence obtain an exponential tail bound. We borrow these ideas to derive an exponential tail bound for variance-dominated processes. To do this, we introduce the k-th hitting time of the target state, which is similar in the theorem (Theorem 2.6.2) of (Menshikov et al., 2016). We combine this recurrent method with the extended Optional Stopping Time theorem.

3.3.1.1 Variance Overcomes Negative Drift w.h.p.

We proceed to prove our main theorem by considering the most general variance drift theorem which overcomes some negative drift. Following the setting of (Hevia Fajardo et al., 2023), in variance-transformed cases, we focus on the first hitting time of a discrete-time stochastic process X_t at 0 given that $X_t \in [0, b]$. The proof of Theorem 3.3.1 uses Lemma 3.3.1.

Lemma 3.3.1. *Let $(X_t)_{t\geq 0}$ be random variables over $\mathbb{R}_{\geq 0}$, each with finite expectation. Let T be any stopping time of X_t. If there exist constants $r, \eta > 0$ with respect to j, t such that for any $j \geq 0$, $\mathrm{E}\big(\mathbb{1}_{\{T>t\}}\mathbb{1}_{\{|X_t - X_{t+1}|\geq j\}} \mid \mathcal{F}_t\big) \leq r/(1+\eta)^j$, then there exists a positive constant c such that $\mathrm{E}\big(|X_{t+1} - X_t| \cdot \mathbb{1}_{\{T>t\}} \mid \mathcal{F}_t\big) \leq c$ for all $t \in \mathbb{N} \cup \{0\}$.*

Proof of Lemma 3.3.1. Notice that for all $T > t$,

$$\mathrm{E}(|X_{t+1} - X_t| \mid \mathcal{F}_t) = \int_0^\infty \Pr(|X_{t+1} - X_t| \geq j \mid \mathcal{F}_t)\,\mathrm{d}j$$

Using the step size condition, we have

$$\begin{aligned} &\leq \int_0^\infty \frac{r}{(1+\eta)^j}\,\mathrm{d}j \\ &= \frac{r}{\log(1+\eta)} := c. \end{aligned}$$

□

By using Lemma 3.3.1, we now satisfy condition (4) in the Optional Stopping Time Theorem, enabling us to proceed with the main proof. Utilising the extended Optional Stopping Time Theorem, we follow a classic approach for generalisation, which also frees us from the fixed step size condition and the need for the Azuma-Hoeffding inequality for sub-Gaussian supermartingales[1]. We further construct a new stochastic process $Y_t =$

[1]The proofs in (Kötzing, 2016) mainly rely on the Azuma-Hoeffding inequality for sub-Gaussian supermartingales.

$b^2 - (b - X_t)^2 + \delta t$, connected to the original process and the variance of the drift. By employing the idea of the k-th hitting time from (Menshikov et al., 2016), we obtain the upper bound for the k-th hitting time, allowing us to construct the recurrence. We present the main theorem of this paper.

Theorem 3.3.1. *Let $(X_t)_{t\in\mathbb{N}}$ be a sequence of random variables in a finite state space $S \subseteq \mathbb{R}$ adapted to a filtration $(\mathcal{F}_t)_{t\in\mathbb{N}}$, and let $T = \inf\{t \geq 0 \mid X_t \leq 0\}$. Suppose*

(A1) there exist $\delta > 0$ such that for all $t < T$, it holds that

$$\mathrm{E}_t\left((X_{t+1} - X_t)^2 - 2(X_{t+1} - X_t)(b - X_t)\right) \geq \delta$$

(A2) and for all $t \leq T$, it holds that $0 \leq X_t \leq b$.

Moreover, for all $t \geq 0$, assume there exist constants $r, \eta > 0$ with respect to j, t, for any $j \geq 0$, $\mathrm{E}\big(\mathbb{1}_{\{T>t\}}\mathbb{1}_{\{|X_t - X_{t+1}|\geq j\}} \mid \mathcal{F}_t\big) \leq r/(1+\eta)^j$. Then, for $\tau > 0$, $\Pr(T > \tau) \leq e^{-\tau\delta/eb^2}$.

Proof of Theorem 3.3.1. For any $k \in \mathbb{N}$, we define $T_k = \inf\{t \geq k \mid X_t \leq 0\}$. Note that for any k and m, if $m < k$, then $\{\omega \in \Omega \mid T_k(\omega) = m\} = \emptyset \in \mathcal{F}_m$ and if $m \geq k$, from definition of T_k (consider the process after X_k), then $\{\omega \in \Omega \mid T_k(\omega) = m\} = \{\omega \in \Omega \mid X_i(\omega) > 0 \text{ and } X_m(\omega) \leq 0 \text{ for } i = k, \ldots, m-1\}$ is in the σ-algebra $\mathcal{F}_m$ since X is adapted to $\mathcal{F}$. Thus, T_k is a stopping time with respect to $(\mathcal{F}_t)_{t\geq 0}$. Since X_t has a negative drift, we want to transform the process to overcome such a negative drift by using variance. We proceed by defining Y_t. Also, we define Z_t to connect Y_t and T_k to establish the recurrence step.

We would like to apply the extended optional stopping theorem for the random variables X_t, Z_t. To do so, we first need to prove that the expectation of T is finite and thus the expectation of T_k is finite for any given k. We want to apply the additive drift theorem He and Yao, 2001 on Y_t instead of directly on X_t. Note that if $X_t \leq 0$ and $X_t \in [0, b]$, then $Y_t = b^2 - (b - X_t)^2 \leq 0$. If $Y_t \leq 0$, then $b^2 - (b - X_t)^2 \leq 0$ which implies that $b^2 \leq (b - X_t)^2$.

$X_t \in [0, b]$ and $b - X_t \in [0, b]$, then we have $b \leq b - X_t$ and thus we get $X_t \leq 0$. We can see the equivalence of these two events from here. So $Y_t \leq 0$ is equivalent to $X_t \leq 0$ from the definition of Y_t and the equivalence of these two events gives

$$T = \inf\{t \geq 0 \mid X_t \leq 0\} = \inf\{t \geq 0 \mid Y_t \leq 0\}.$$

Consider the process $Y_t = b^2 - (b - X_t)^2$ for all $t \in \mathbb{N}$, we have

$$\begin{aligned}
\mathrm{E}_t(Y_t - Y_{t+1}) &= \mathrm{E}_t\big((b - X_{t+1})^2 - (b - X_t)^2\big) \\
&= \mathrm{E}_t\big(X_{t+1}^2 - X_t^2 - 2b\,(X_{t+1} - X_t)\big) \\
&= \mathrm{E}_t\left((X_{t+1} - X_t)^2 - 2\,(X_{t+1} - X_t)\,(b - X_t)\right)
\end{aligned}$$

Using (A_1) gives

$$\geq \delta > 0. \tag{3.2}$$

By the classic additive drift theorem (He and Yao, 2001) on Y_t, we can derive the upper bound for the expected runtime is $\mathrm{E}(T) \leq \frac{b^2 - (b - X_0)^2}{\delta}$. Now, we define another process $Z_t = Y_t + \delta t$, then

$$\begin{aligned}
\mathrm{E}_t(Z_t - Z_{t+1}) &= \mathrm{E}_t(Y_t - \delta t - Y_{t+1} - \delta(t+1)) \\
&= \mathrm{E}_t(Y_t - Y_{t+1} - \delta) \geq 0,
\end{aligned}$$

where the last inequality follows from Eq (3.2). Also notice that Y_t, Z_t are adapted to the filtration $\mathcal{F}_t$ and due to X_t is finite then $\mathrm{E}(|X_t|) < \infty$. This implies that $\mathrm{E}(|Y_t|), \mathrm{E}(|Z_t|)$ are bounded as well for any given t. Then, both Z_t and Y_t are super-martingales according to Definition A.1.10. Since we have shown that T_k is a stopping time, we can apply Theorem A.3.2 with respect to Z_t.

$$\mathrm{E}_k(Z_k - Z_{T_k}) \geq 0$$

Substituting Z_t gives

$$\mathrm{E}_k(Y_k + \delta k - (Y_{T_k} + \delta T_k)) \geq 0$$

Rearranging the equation gives

$$\mathrm{E}_k(Y_k - Y_{T_k}) \geq \delta \mathrm{E}_k(T_k - k)$$

Now, note that

$$\begin{aligned} 0 \leq \mathrm{E}_k(Y_{T_k}) &= \mathrm{E}_k\big(b^2 - (b - X_{T_k})^2\big) \\ &= b^2 - \mathrm{E}_k\big((b - X_{T_k})^2\big). \end{aligned}$$

Using Jensen's inequality gives,

$$\leq b^2 - (\mathrm{E}_k(b - X_{T_k}))^2$$

Since $\mathrm{E}(T) < \infty$ as shown and we have shown Lemma 3.3.1, we satisfy condition (4) in the extended Optional Stopping Time Theorem. Using the Optional Stopping Theorem (Theorem A.3.2) on stopping time T_k gives,

$$\leq b^2 - (b - X_k)^2.$$

So we can see the bounds for $\mathrm{E}_k(Y_{T_k}) \in [0, b^2 - (b - X_k)^2]$ and deduce that $b^2 - (b - X_k)^2 \geq \mathrm{E}_k(Y_k - Y_{T_k}) \geq 0$. By setting

$$\theta_k := \frac{\mathrm{E}_k(Y_k - Y_{T_k})}{\delta} \in [0, \frac{b^2}{\delta}],$$

and using $\mathrm{E}_k(Y_k - Y_{T_k}) \geq \delta \mathrm{E}_k(T_k - k)$, we have shown the following estimate:

$$\mathrm{E}(T_k - k \mid \mathcal{F}_k) \leq \theta_k \quad \text{for some } \theta_k \geq 0. \tag{3.3}$$

Taking $\theta := eb^2/\delta$, we get for all natural numbers $k > 0$,

$$\mathrm{E}\big(\mathbb{1}_{\{T_k - k > \theta\}} \mid \mathcal{F}_k\big) = \Pr(T_k - k > \theta \mid \mathcal{F}_t).$$

Since $T_k - k \geq 0$, we can apply Markov's inequality

$$\leq \mathrm{E}(T_k - k \mid \mathcal{F}_k)/\theta.$$

Using Eq. (3.3) gives

$$\begin{aligned} &\leq \theta_k/\theta \\ &\leq e^{-1}. \end{aligned} \tag{3.4}$$

Next, we construct a recurrence relation by using Equation 3.4 above. We consider the intersection decomposition (A similar intersection decomposition can be found in Menshikov et al., 2016) between event $\{T_0 > (k+1)\theta\}$ and $\{T_0 > k\theta\}$ by introducing $T_{k\theta}$

$$\begin{aligned} \{T_0 > (k+1)\theta\} &= \{T_0 > k\theta\} \cap \{T_{k\theta} > (k+1)\theta\} \\ &= \{T_0 > k\theta\} \cap \{T_{k\theta} - k\theta > \theta\} \end{aligned}$$

Then,

$$\mathbb{1}_{\{T_0>(k+1)\theta\}} = \mathbb{1}_{\{T_0>k\theta\}} \cdot \mathbb{1}_{\{T_{k\theta}-k\theta>\theta\}}.$$

So, we have the recurrence relation:

$$\begin{aligned} \Pr(T_0 > (k+1)\theta) &= \mathrm{E}\big(\mathbb{1}_{\{T_0>(k+1)\theta\}}\big) \\ &= \mathrm{E}\big(\mathbb{1}_{\{T_0>k\theta\}} \cdot \mathbb{1}_{\{T_{k\theta}-k\theta>\theta\}}\big). \end{aligned}$$

Noting that $\{T_0 - k\theta > 0\} \in \mathcal{F}_{k\theta}$ and use the tower property of conditional expectation,

$$= \mathrm{E}\big(\mathbb{1}_{\{T_0>k\theta\}}\mathrm{E}\big(\mathbb{1}_{\{T_{k\theta}-k\theta>\theta\}} \mid \mathcal{F}_{k\theta}\big)\big)$$

Using Eq. 3.4 gives

$$\begin{aligned} &\leq \mathrm{E}\big(\mathbb{1}_{\{T_0>k\theta\}} \cdot e^{-1}\big) \\ &= \Pr(T_0 > k\theta) \cdot e^{-1} \end{aligned}$$

By induction, we have

$$\leq e^{-(k+1)}.$$

Then, finally, we derive (rearranging all the terms) for $\tau \geq 0$:

$$\Pr(T_0 \geq \tau) \leq e^{-\tau/\theta}$$

where $\theta \geq eb^2/\delta$. □

Our main result (Theorem 3.3.1) allows the increased tolerance of negative drift rather than non-negative drift tendency and only necessitates a constant second moment of drift, instead of variance, as outlined in Kötzing, 2016. Consequently, we can establish a more precise exponential tail bound for stochastic processes with a constant second moment of the drift, even under weak, zero, or negative drift.

3.3.1.2 Standard Variance Drift

This section presents the standard variance drift scenario (Theorem 3.3.2) as a corollary of Theorem 3.3.1. More precisely, we now restrict to the non-negative drift tendency. We first define several conditions which will be used later.

(C1*) There exist constants $r, \eta > 0$ with respect to j, t, such that for any $j \geq 0$ and for all $t \geq 0$,

$$\mathrm{E}\big(\mathbb{1}_{\{T>t\}}\mathbb{1}_{[|X_t - X_{t+1}|\geq j\}} \mid \mathcal{F}_t\big) \leq \frac{r}{(1+\eta)^j}.$$

(C1) There exists a constant $c > 0$ such that $|X_t - X_{t+1}| < c$ for all $t \geq 0$.

(C2) $\mathrm{E}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq 0$ for all $t \geq 0$.

(C3) There exists some constant $\delta > 0$ such that $\mathrm{E}(X_{t+1} - X_t)^2 \mid \mathcal{F}_t) \geq \delta$ for all $t \geq 0$.

Theorem 3.3.2. *Let $(X_t)_{t\geq 0}$ be random variables over $\mathbb{R}_{\geq 0}$, each with finite expectation, such that conditions $(C1^*), (C2)$ and $(C3)$ hold. For any $b > 0$, define $T = \inf\{t \geq 0 \mid X_t \geq b\}$. If $X_0 \in [b]$, then $\mathrm{E}(T) \leq (b^2 - X_0^2)/\delta$. Moreover, for $\tau > 0$, $\Pr(T \geq \tau) \leq e^{-\tau\delta/eb^2}$.*

Proof. Let us define $Y_t = n - X_t$ and $T := \{t \geq 0 \mid X_t \geq n\} = \{t \geq 0 \mid Y_t \leq 0\}$. So from $(C2), (C3)$ we have

$$\mathrm{E}_t(Y_t - Y_{t+1}) \geq 0$$
$$\mathrm{E}_t\big((Y_t - Y_{t+1})^2\big) \geq \delta$$

This implies that Y_t satisfies $(A1)$ in Theorem 3.3.1. Notice that $Y_t \in [0, n]$ so we set $b = n$ in Theorem 3.3.1, and thus we satisfy $(A2)$. Now Y_t satisfies all conditions in Theorem 3.3.1. We apply Theorem 3.3.1 to Y_t with $Y_T = 0$, and obtain the desired results. □

Theorem 3.3.2 tells us that under the standard variance drift case as discussed in Kötzing, 2016, we can derive an exponential tail bound for the runtime and such a process exhibits a high concentration around the expectation.

Now, we present a corollary which consists of the fixed step size condition.

Corollary 3.3.1. *Let $(X_t)_{t\geq 0}$ be random variables over $\mathbb{R}_{\geq 0}$, each with finite expectation which satisfy conditions $(C1), (C2)$ and $(C3)$. For any $b > 0$, define $T = \inf\{t \geq 0 \mid X_t \geq b\}$. Given that $X_0 \in [0, b]$, then $\mathrm{E}(T) \leq (b^2 - X_0^2)/\delta$. Moreover, for $\tau > 0$, $\Pr(T \geq \tau) \leq e^{-\tau\delta/eb^2}$.*

Proof. This proof is a direct result of Theorem 3.3.2 by using fixed step size condition (C1). □

Furthermore, we derive a tail bound for the variance-dominated processes with two absorbing states. Following the setting of Theorem 10 in Göbel et al., 2022, we will prove the next theorem.

Theorem 3.3.3. *Let $(X_t)_{t\geq 0}$ be random variables over $\mathbb{R}_{\geq 0}$ adapted to a filtration $(\mathcal{F}_t)_{t\in\mathbb{N}}$, each with finite expectation such that $(C1*)$, $(C3)$ and $\mathrm{E}(X_{t+1} - X_t \mid \mathcal{F}_t) = 0$ hold. For any $b > 0$, define $T = \inf\{t \geq 0 \mid X_t \in \{0, b\}\}$. If $X_0 \in [0, b]$, then $\mathrm{E}(T) \leq (X_0(b - X_0))/\delta$.*

Moreover, for $\tau > 0$, $\Pr(T \geq \tau) \leq e^{-2\tau\delta/eb^2}$.

This proof is similar to Theorem 3.3.1 except that we use a different stochastic process $Y_t = X_t(b - X_t) + \delta t$ and show that Y_t is a super-martingale. Unlike the proof of Theorem 3.3.1, the proof of Theorem 3.3.3 uses the extended Optional Stopping Time Theorem for super-martingales (Williams, 1991). We defer the proof to the appendix.

3.3.1.3 Standard Drift

If a stochastic process has drift ε where $\varepsilon > 0$ is some positive constant, then we can give a different proof for the upper tail bound for additive drift from the proof in Kötzing, 2016. This provides a more precise exponential upper tail bound.

Theorem 3.3.4. *Let* $(X_t)_{t\geq 0}$ *be random variables over* $\mathbb{R}$*, each with finite expectation which satisfy condition* $(C1*)$ *and* $\mathbb{E}[X_{t+1} - X_t \mid \mathcal{F}_t] \geq \varepsilon$ *for some* $\varepsilon > 0$*. For any* $b > 0$*, define* $T = \inf\{t \geq 0 \mid X_t \geq b\}$*. If* $X_0 \in [0, b]$*, then* $E(T) \leq \frac{b - X_0}{\varepsilon}$*. Moreover, for* $\tau > 0$*,* $\Pr(T \geq \tau) \leq e^{-\tau\varepsilon/eb}$.

We recover the exponential upper tail bound for the additive drift theorem. The bound we obtain gets rid of the coefficients $1/8c^2$ in Kötzing, 2016. Theorem 2.5.12 of Menshikov et al., 2016 provides a similar result and uses a similar recurrence proof idea. While our result generalises the result in Menshikov et al., 2016 by relaxing the assumption of fixed step size, we provide a meaningful bound on the first hitting time instead of bounding it above by infinity.

In summary, we have discovered a simple alternative to the Azuma-Hoeffding inequality that provides an exponential tail bound by only relying on basic martingale theory. This result can be applied to the random local search (RLS) type algorithms that make finite steps at each iteration, as well as other randomised algorithms including evolutionary al-

gorithms (EAs) that account for the possibility of large jumps occurring. Another benefit from Theorem 3.3.1 is that it allows the tolerance of the negative drift up to $-\Omega(1/b)$.

3.4 Applications to Random 2-SAT & Graph Colouring

In this section, we illustrate our theorems on examples. We consider the examples provided by (Göbel et al., 2022; McDiarmid, 1993; Mitzenmacher and Upfal, 2005). which include variance-dominated processes. We first discuss the Random 2-SAT problem.

3.4.1 Applications to Random 2-SAT

The 2-SAT Algorithm is designed to solve instances of the 2-SAT problem, where a formula consists of clauses and each of them contains exactly two literals (either variables or negations). In each iteration, the algorithm selects an unsatisfied clause and picks one of the literals uniformly at random. The truth value of the variable corresponding to this literal is then inverted. The process is repeated until either we meet the stopping criteria or a valid truth assignment is found.

Papadimitriou, 1991 firstly provided a time complexity analysis on such a simple randomised algorithm that returns a satisfying assignment of a satisfiable 2-SAT formula ϕ with n variables. Later, Göbel et al., 2022 recovered the results using the drift analysis tools in Appendix A.4.

By applying Theorem 3.3.2 with a variance bound 1, we can bound the number of function evaluations of order $O(n^4)$ with an upper exponential tail bound.

Theorem 3.4.1. *Given any $r \geq 0$, the randomised 2-SAT algorithm, when run on a satisfiable 2-SAT formula over $n \in \mathbb{N}$ variables, terminates in at most rn^4 time with probability at least $1 - e^{-r/e}$.*

Proof. This proof follows the same proof exactly in (Göbel et al., 2022) except we consider the tail bound for the runtime. We define $X_t \in [n]$ to be the number of variables that the current truth assignment agrees with a satisfying assignment, and T is to be the first hitting time that $X_t = n$. From the proof of Theorem 15 in (Göbel et al., 2022), it has been shown that the variance bound $\mathrm{Var}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq 1$ and $\Pr(X_{t+1} = X_t + 1 \mid \mathcal{F}_t) \geq 1/2$ and $\Pr(X_{t+1} = X_t - 1 \mid \mathcal{F}_t) \leq 1/2$. Thus, we can easily have $\mathrm{E}((X_{t+1} - X_t)^2 \mid \mathcal{F}_t) \geq \mathrm{Var}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq 1$ and $\mathrm{E}(X_{t+1} - X_t \mid \mathcal{F}_t) \geq 0$. This satisfies conditions (C2) and (C3) in Theorem 3.3.2. Then we need to verify that the randomised 2-SAT algorithm satisfies the step size condition. It is easy to see that from Algorithm 16, X_t always has a fixed step size 1 (either backwards or forwards). Then, we have verified the step size condition and can now apply Theorem 3.3.2 with $\theta = en^2$ to obtain the desired tail bound.

$$\Pr(T \geq rn^2) = e^{-rn^2/en^2} \leq e^{-r/e}.$$

The rest of the step follows from the proof of Theorem 15 in (Göbel et al., 2022) and by multiplying the executed time of the algorithm (i.e. $O(n^2)$), we can conclude the desired tail bound. □

3.4.2 Applications to Graph Colouring

Now we consider the example of graph colouring which has already been studied by McDiarmid, 1993 and Göbel et al., 2022. The Recolour algorithm is a method that aims to generate a 2-colouring mapping, with the condition that no monochromatic edges can be found. The method assumes a subroutine SEEK which, given a 2-colouring of the points, outputs a monochromatic edge if one exists. If there are no monochromatic edges, then the subroutine reports that there are none, and the algorithm terminates. Otherwise, the algorithm repeatedly picks a point uniformly at random from the given monochromatic edge and changes its colour.

Göbel et al., 2022 provides a simpler proof of the $O(n^4)$ expected runtime of the recolouring algorithm for finding a 2-colouring with no monochromatic triangles on 3-colorable graphs. Following the setting and the proof of Göbel et al., 2022, by using Theorem 3.3.3, we can derive the following:

Theorem 3.4.2. *The randomised Recolouring algorithm on a 3-colorable graph with $n \in \mathbb{N}$ vertices, terminates in at most rn^4 with probability at least $1 - e^{\frac{-4r}{3e}}$ for any $r \geq 0$.*

Proof. This proof follows the proof in (Göbel et al., 2022) except we consider the tail bound for the runtime. We first fix a 3-colouring of this given graph, two colours out of three and a set of vertices which consists of these two colours. We first fix a 3-colouring of the given graph, choose two colours from the three, and consider the set of vertices that are assigned one of these two colours. Let m denote the size of this set, and define $Y_t \in [m]$ as the number of vertices in this set whose colour at iteration $t \in \mathbb{N}$, under the 2-colouring produced by the Recolour algorithm, matches their colour in the original 3-colouring, restricted to the two chosen colours. We define the first hitting time by $T := \inf\{t > 0 \mid Y_t = 0 \text{ or } m\}$.

It has been shown in the proof of Theorem 14 in (Göbel et al., 2022) that the runtime of Recolour can be upper bounded by the time it takes the algorithm to find a valid 2-colouring on the selected subset of vertices, such that no edge between these vertices is monochromatic. Therefore, we focus on analysing the tail bound of T. It has also been shown in (Göbel et al., 2022) that for $s \in [m] \setminus \{0, m\}$,

$$\Pr\left(Y_{t+1} = Y_t \pm 1 \mid Y_t = s\right) = 1/3$$

Note that we can compute the drift and the second moment of the drift

$$\begin{aligned}
\mathrm{E}(Y_t - Y_{t+1} \mid Y_t = s) &= 1 \times \frac{1}{3} + (-1) \times \frac{1}{3} + 0 \times \frac{1}{3} \\
&= 0 \\
\mathrm{E}\big((Y_t - Y_{t+1})^2 \mid Y_t = s\big) &= 1 \times \frac{1}{3} + (-1)^2 \times \frac{1}{3} = \frac{2}{3}
\end{aligned}$$

Now, we satisfy condition (C1*), (C3) and zero drift condition in Theorem 3.3.3. Notice that $Y_0(m - Y_0) \leq m^2/4$ since $Y_0 \in [m]$ and this attains the maximum when $Y_0 = m/2$. Also, as the definition of m, we have $m \leq n$. Thus, we derive the following

$$\mathrm{E}(T) \leq \frac{3\mathrm{E}(Y_0\,(m - Y_0))}{2} \leq \frac{3n^2}{8}.$$

Moreover, for any $r \geq 0$, we have

$$\Pr\left(T \geq rn^2\right) \leq e^{-2rn^2 \cdot \frac{2}{3}/en^2} \leq e^{\frac{-4r}{3e}}.$$

The rest of the step follows from the proof of Theorem 14 in (Göbel et al., 2022) and by multiplying the executed time of the algorithm at each step (i.e. $O(n^2)$), so we finish the proof. □

3.5 Applications to coevolutionary algorithms

Next, we consider a slightly more complicated example, which is about coevolutionary algorithms (CoEAs). Competitive coevolutionary algorithms are designed to solve Maximin optimisation problems or adversarial optimisation problems, including a typical class of problems: two-player zero-sum games. There are various applications, including CoEA-GAN (Al-Dujaili et al., 2018), competitive coevolutionary search heuristics on cybersecurity problems (Lehre et al., 2023) and enhancing GAN by using the coevolutionary approach for image translation (Shu et al., 2019).

We are interested in whether competitive CoEAs can help us find the Nash equilibrium efficiently. We use the formulation in Nisan et al., 2007 to define Nash equilibrium rigorously. In this dissertation, we only focus on Pure Strategy Nash Equilibrium (abbrev. NE) defined in Definition 2.3.1.

Inspired by the Nash solution concept, Lehre, 2022 has defined the following pairwise

dominance relation and embedded it into a population-based CoEA (details can be checked in Lehre, 2022).

Definition 3.5.1 (Pairwise dominance)**.** *Given a function* $g = \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ *and two pairs* $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$*, we say that* (x_1, y_1) *dominates* (x_2, y_2) *with respect to* g*, denoted* $(x_1, y_1) \succeq_g (x_2, y_2)$*, if and only if* $g(x_1, y_2) \geq g(x_1, y_1) \geq g(x_2, y_1)$*.*

There is a single-pair CoEA called Randomised Local Search with Pairwise Dominance (RLS-PD) (Hevia Fajardo et al., 2023). We will also compare both single-pair CoEA and population-based CoEA in Chapter 6 later. In this chapter, we use RLS-PD to illustrate how to use our new drift theorem. It has been shown that RLS-PD (Algorithm 18) can find the NE of a simple pseudo-Boolean benchmark called Bilinear in expected polynomial runtime. The stochastic processes induced by the randomised local search with pairwise dominance (RLS-PD) on the Bilinear problem are exactly a variance-transformed process. We use Theorem 3.3.1 to show the exponential tail bound for the runtime.

Algorithm 18 samples a search point (a pair of points (x_1, y_1)) uniformly at random. In each iteration, Algorithm 18 uses the local mutation operator to generate a new search point. If the new search point pairwise dominates the original search point in a pairwise-dominance manner, then the original search point is replaced by the new one. Otherwise, the original search point remains the same.

3.5.1 The Bilinear Problem

In this section, we consider a simple class of discrete Maximin benchmark called Bilinear, which was first proposed by Lehre (2022). In this work, we use a variation of Bilinear as (Hevia Fajardo et al., 2023) (Definition 2.5.3) to simplify our calculation and illustrate applications of our main theorem.

We consider OPT as our solution concept, which defines the optimal solution for this

MAXIMIN optimisation problem. We focus on the problem setting $\alpha = 1/2 \pm O(1/\sqrt{n})$ and $\beta = 1/2 \pm O(1/\sqrt{n})$, as considered in (Hevia Fajardo et al., 2023). This setting represents one of the simplest cases (with the help of the genetic drift towards the optimum), making it a natural starting point for theoretical analysis. Despite its simplicity, the behaviour of RLS-PD in this regime still exhibits non-trivial cyclic dynamics that are challenging to analyse. To better understand the drift (see detailed discussion of drift in Section 2.2) induced by RLS-PD in such settings, we introduce a new drift theorem (Theorem 3.2.1) tailored to capture its stochastic behaviour. From the perspective of classical evolutionary computation, RLS (Randomised Local Search) is one of the most fundamental algorithms: it selects a single bit uniformly at random to flip and accepts the resulting solution if it does not worsen fitness. Studying RLS-PD, a simple variant of RLS adapted to game-theoretic settings, can therefore provide valuable insight into the interplay between evolutionary computation and game-theoretic setting. Next, let us derive the exponential tail bound of RLS-PD for finding the Nash equilibrium.

3.5.2 RLS-PD solves BILINEAR efficiently w.h.p.

Theorem 3.5.1. *Consider $\alpha \in [1/2 - A/\sqrt{n}, 1/2 + A/\sqrt{n}]$ and $\beta \in [1/2 - B/\sqrt{n}, 1/2 + B/\sqrt{n}]$, where $A, B > 0$ are constants and $3(A+B)^2 \leq 1/2 - \delta'$ for some constant $\delta' > 0$. The expected runtime of RLS-PD on $\text{BILINEAR}_{\alpha,\beta}$ is $O(n^{1.5})$. Moreover, the runtime is at most $2rn^{1.5}$, with probability at least $1 - e^{-\Omega(r)}$ for any $r \geq 0$.*

Proof. This proof follows the proof in (Hevia Fajardo et al., 2023) except we consider the tail bound for the runtime. We define $T := \inf\{t > 0 \mid (x_t, y_t) \in \text{OPT}\} = \inf\{t > 0 \mid M_t = 0\}$, where (x_t, y_t) are the current solutions of RLS-PD and $0 \leq M_t \leq n(\alpha + \beta) \leq 2n$ is defined

in Definition A.4.1. Let $b = 2(A+B)\sqrt{n}+1$ and a new potential function defined by

$$h(x) := \begin{cases} b^2 - (b-x)^2 & \text{if } x \leq 2(A+B)\sqrt{n} \\ b^2 - (b-x) & \text{if } x > 2(A+B)\sqrt{n} \end{cases}$$

where $x \in \mathbb{R}$. It has been shown in the proof of Theorem 3.1 in (Hevia Fajardo et al., 2023) that for every generation $t < T$, the drift of $h(M_t)$ is lower bounded by $\delta(M_t)$ and $\delta(M_t) \geq \frac{1}{2\sqrt{n}} := \delta_1$, where

$$\delta(x) = \begin{cases} \delta_1 & \text{if } x \leq 2(A+B)\sqrt{n} \\ \frac{x-(A+B)\sqrt{n}}{2n} & \text{if } x > 2(A+B)\sqrt{n} \end{cases}$$

and $x \in \mathbb{R}$. Due to the piece-wise drift, we cannot directly apply our variance drift theorem (tail bound). To derive the exponential tail bound for the runtime, we divide the analysis into two phases. We define

$$T_{\text{phase1}} := \inf\{t > 0 \mid M_t < b \text{ given } M_0 \geq b\}.$$

Note that $T = \inf\{t > 0 \mid M_t = 0\} = \inf\{t \geq T_{\text{phase1}} \mid M_t = 0\}$. So we define $T_{\text{phase2}} := T - T_{\text{phase1}}$. From the definition, we can bound T_{phase2} from above by the first hitting time

$$T'_{\text{phase2}} := \inf\{t > 0 \mid M_t = 0 \text{ given } M_0 = b-1\}.$$

From the drift condition above for the case $M_t \leq 2(A+B)\sqrt{n}$, we satisfy (A1), (A2) in Theorem 3.3.1 and the step size condition directly follows from the fact that RLS-PD only makes one step jump at each iteration. By applying Theorem 3.3.1, we get: for any $r > 0$,

$$\Pr(T'_{\text{phase2}} \geq rn^{1.5}) \leq e^{-rn^{1.5}\delta_1/e(b-1)^2}.$$

Taking $\delta_1 = \frac{1}{2\sqrt{n}}$ and substituting $b = 2\,(A+B)\,\sqrt{n}+1$ give

$$\begin{aligned} &\leq \exp\left(-\frac{rn^{1.5}\frac{1}{2\sqrt{n}}}{4e(A+B)^2 n}\right) \\ &\leq \exp\left(-\frac{r}{8e(A+B)^2}\right) = e^{-\Omega(r)}. \end{aligned}$$

Then, we consider the tail bound for T_{phase1}. Let $h_{\max} := b^2 - b + n(\alpha + \beta) = O(n)$. We use Theorem 3.3.4 by setting $X_t = h_{\max} - h(M_t)$ for $t < T_{\text{phase1}}$. The drift for X_t is at least δ_1. So we have the tail bound

$$\Pr(T_{\text{phase1}} > rn^{1.5}) \leq e^{-rn^{1.5}\delta_1/eh_{\max}}$$

Taking $\delta_1 = \frac{1}{2\sqrt{n}}$ and substituting $h_{\max} = b^2 - b + n(\alpha + \beta)$ give

$$\begin{aligned} &\leq \exp\left(-\frac{rn^{1.5}\frac{1}{2\sqrt{n}}}{e\left(b^2 - b + n(\alpha + \beta)\right)}\right) \\ &= e^{-\Omega(r)}. \end{aligned}$$

Note that $\{T \geq 2rn^{1.5}\} \subseteq \{T_{\text{phase1}} \geq rn^{1.5}\} \cup \{T_{\text{phase2}} \geq rn^{1.5}\}$. The union bound gives

$$\Pr\left(T \geq 2rn^{1.5}\right) \leq \Pr\left(T_{\text{phase1}} \geq rn^{1.5}\right) + \Pr\left(T_{\text{phase2}} \geq rn^{1.5}\right)$$

Note that $T'_{\text{phase2}} \geq T_{\text{phase2}}$ and thus we have $\{T_{\text{phase2}} \geq rn^{1.5}\} \subseteq \{T'_{\text{phase2}} \geq rn^{1.5}\}$. Substituting the bounds gives

$$\leq 2e^{-\Omega(r)} = e^{-\Omega(r)}.$$

This completes the proof. □

Theorem 3.5.1 shows that RLS-PD can find the Nash Equilibrium in $O(n^{1.5})$ with overwhelmingly high probability. An exponential tail bound provides a stronger performance guarantee up to the distribution of the runtime rather than the sole expectation. This is one of the direct applications of Theorem 3.3.1.

3.5.3 RLS-PD forgets the Nash Equilibrium w.h.p.

After the algorithm finds a Nash Equilibrium efficiently, the algorithm not only forgets the Nash Equilibrium found but also moves away from it by a distance $\Omega(\sqrt{n})$ in $O(n)$ iterations w.h.p. This is shown by the following theorem.

Theorem 3.5.2. *Let $\alpha = 1/2 \pm A/\sqrt{n}$ and $\beta = 1/2 \pm B/\sqrt{n}$, where $A, B > 0$ are constants. Consider RLS-PD on $\textsc{Bilinear}_{\alpha,\beta}$. For any initial search point (x_0, y_0), the expected time for the search point to first move away from* OPT *by a Manhattan distance of at least $(A+B)\sqrt{n}$ is $O(n)$. Moreover, the runtime is at most rn, with probability at least $1-e^{-\Omega(r)}$ for any $r > 0$.*

Proof. This proof follows the proof in (Hevia Fajardo et al., 2023) except we consider the tail bound for the runtime. Define $T := \inf\{t \mid M_t \geq (A+B)\sqrt{n}\}$, where M_t is the current Manhattan distance to the set OPT. We assume $M_0 \leq (A+B)\sqrt{n}$. Otherwise, we reduce to the trivial case with $T = O(1)$. We would like to show $\mathrm{E}(T) \leq O(n)$. It has been shown in the proofs of Lemma A.4.1 in Hevia Fajardo et al., 2023 that for every generation $t < T$,

$$\mathrm{E}\left(M_t - M_{t+1} - \frac{M_t - (A+B)\sqrt{n}}{2n}; t < T \mid M_t\right) \geq 0.$$

We cannot directly use additive drift on $Y_t := (A+B)\sqrt{n} - M_t$ since the drift $\mathrm{E}_t(Y_t - Y_{t+1})$ can potentially be negative. Instead of using additive drift, we use the variance drift theorem (Theorem 3.3.1) to show the runtime with the tail bound. Let $a = 0, b = (A+B)\sqrt{n}$ and Y_t as defined above in Theorem 3.3.1. As shown in the proof of Theorem 4.1 in Hevia Fajardo et al., 2023,

$$\begin{aligned}&\mathrm{E}_t\big((Y_{t+1} - Y_t)^2 - 2(Y_{t+1} - Y_t)(b - Y_t); t < T\big)\\ &\geq \frac{1}{2} - \frac{2(A+B)}{\sqrt{n}}\end{aligned}$$

For sufficiently large n, there exists a constant $\delta_2 > 0$ s.t.

$$\geq \delta_2.$$

The expected runtime is $\mathrm{E}(T) = O(b^2) = O(n)$. For the worst case $Y_0 = b$ and $\tau = rn$, we get the tail bound: for any $r > 0$,

$$\Pr(T > rn) \leq \exp\left(-\frac{rn\delta_2}{eb^2}\right)$$

Taking constant $\delta_2 > 0$ and substituting b give

$$= \exp\left(-\frac{rn \cdot \delta_2}{(A+B)^2 n}\right)$$
$$= e^{-\Omega(r)}.$$

□

To conclude, in competitive coevolution, we can see that even though the algorithm can find the optimum in polynomial runtime, it can still suffer from evolutionary forgetting (i.e. forget the optimum found in previous iterations) with high probability. So, only the expected runtime estimate might not be sufficient to determine whether a coevolutionary algorithm is good or not. The current runtime analysis highlights a weakness of RLS-PD. Additionally, other performance measures may be necessary to study such algorithms with complex dynamics, such as regret. It remains unclear under which conditions RLS-PD or other variants of coevolutionary algorithms stay within a close neighbourhood of NE. We conjecture that RLS-PD might move $O(\sqrt{n})$ away from NE and then move back to NE and keep such a looping.

3.6 Applications to Regret Analysis of Bandit Learning

We first briefly introduce bandit problems. We have K decisions or "arms", where we obtain the corresponding reward r_a^t when we choose one specific decision at each iteration t. The goal of the bandit algorithm is to maximise cumulative reward among a time horizon T (i.e., $\sum_{t=1}^{T} r_a^t$) (Lattimore and Szepesvári, 2020; Sutton and Barto, 2018). In other words, we want to minimise the opposite quantity (regret), which is the difference between the reward of the chosen arm and the optimal arm at each iteration. We provide a formal definition of regret as follows.

Definition 3.6.1 (Larcher et al., 2023; Lattimore and Szepesvári, 2020)**.** *Given time horizon T, each arm a is associated with a probability distribution $\mathcal{D}(a)$, which we assume to be over $[0,1]$ and for which the mean is denoted as $\mu(a)$; whenever arm a is pulled, the agent receives a reward distributed according to $\mathcal{D}(a)$. The regret (missed reward) of the agent at round $t \in \mathbb{N}$ is defined as $R_t = r_{a^*} - r_{a_t}$, where a_t is the arm chosen at round t, r_a is the reward obtained from reward distribution $\mathcal{D}(a)$ and $a^* = \arg\max_{a \in [K]} \mu(a)$. The goal of the agent is to minimise the total regret $\mathcal{R} = \sum_{t=1}^{T} R_t$ or the total expected regret $\mathrm{E}(\mathcal{R})$.*

In this dissertation, we focus on the non-stationary 2-armed bandit problem, in which the reward distributions may swap over time and $K = 2$. More precisely, the agent receives a reward according to a reward distribution $\mathcal{D}(a_1)$ by pulling arm a_1 and another reward according to distribution $\mathcal{D}(a_2)$ by pulling arm a_2. We assume these two distributions $\mathcal{D}(a_1), \mathcal{D}(a_2)$ are fixed and they will swap if there is a change occurs along the time horizon. To simplify the calculation, we assume these distributions over $[0,1]$. We present the application of our drift theorem (concentration tail bound) on regret analysis of a simple reinforcement learning algorithm for such a bandit problem.

We consider the pseudocode from (Larcher et al., 2023).

Algorithm 7 Random Walk with Asymmetric Boundaries (RWAB)

Require: Time horizon T, number of changes L, two arms a_1, a_2 and Challenge probability $p = \sqrt{L/T}$.

1: Set $a^+ \leftarrow a_1$ and $a^- \leftarrow a_2$.
2: **for** $t \in \{1, 2, \ldots, T\}$ **do**
3: Set $(a^+, a^-) \leftarrow$ Challenge(a^+, a^-) with prob. p.
4: Otherwise pull a^+ once.

Algorithm 8 CHALLENGE

Require: Two arms a^+, a^-

Set $S \leftarrow 0$.
while $-\sqrt{T/L} < S < 1$ **do**
    Pull both a^+ and a^- once and observe rewards r^+ and r^-.
    Update $S \leftarrow S + r^+ - r^-$.
    if $S \geq 1$ **then**
        return (a^+, a^-) otherwise **return** (a^-, a^+)

Note that RWAB is designed to balance the exploration and exploitation for non-stationary bandit problems. RWAB mainly relies on the CHALLENGE operator to determine which arm we prefer to pull or whether we swap the arms. The CHALLENGE operator is designed to use the random walk of action value S on $[-\sqrt{T/L}, 1]$. Larcher et al., 2023 shows that the expected regret of RWAB is $\Theta\left(\sqrt{LT}\right)$ where T is the time horizon and L is the number of changes. We want something stronger to provide more performance guarantees for the regret of RWAB, i.e. a concentration tail bound for the regret estimate. We would like to characterise the distribution of the regret.

Next, we present the main theorem for the regret of RWAB algorithm. In this theorem, we assume $L = o(T)$. We need some important definitions used in the analysis of the RWAB algorithm here. Following the definitions used in (Larcher et al., 2023), we define

Definition 3.6.2. *We say that a^+ swaps or we have a swap if in the call of CHALLENGE, the value of a^+ changes. A swap is called a mistake when $a^+ = a^*$ at the start of CHALLENGE and no change of the underlying distribution of the rewards (abbrev. change) occurs while the CHALLENGE was running.*

Definition 3.6.3. *An era denotes the time between two consecutive changes in the reward distribution. A sub-era is an interval in which both the underlying reward distribution and*

the arm a^+ remain constant. We use L to denote the number of eras and M denote the number of sub-eras.

Theorem 3.6.1. *With probability at least $1-2e^{-\frac{\sqrt{\varepsilon}}{e}}$ for any $\varepsilon \geq 1$, the regret of Algorithm* RWAB *is at most $480\varepsilon(L+\sqrt{LT})$.*

Proof. This proof partially follows from the proof in (Larcher et al., 2023) since we need to deal with the tail bound of the product of multiple random variables. We use era/sub-era defined in Definition 3.6.3 and swap/mistakes defined in Definition 3.6.2. We have L eras since there are L changes from the problem setting, and we let M denote the number of such sub-eras.

Following the analysis of (Larcher et al., 2023), we divide the proof into four parts. We denote the action value S by S_t at iteration t in Algorithm 8. Define τ_{+1} as the hitting time of $+1$ in Algorithm 8, i.e., the minimal $t \geq 1$ such that $S_t \geq 1$. Similarly, for $s = \sqrt{T/L}$, define τ_{-s} to be the minimal $t \geq 1$ such that $S_t \leq s$. Denote the difference of rewards per iteration by $R_t = r^+ - r^- \in [-1, 1]$ and the expected regret is $\Delta = |\mathrm{E}(R_t)| \in (0, 1]$ (the expectation over the reward distributions). Since two reward distributions are fixed from the problem settings, we can see Δ is some constant in $(0, 1]$.

We denote the initial position by S_0 and $S_t \in (-s, 1)$ for all $t < \min\{\tau_{-s}, \tau_{+1}\}$. To simplify the calculation, we overestimate the regret per iteration simply by $R_t \leq 1$. We also define the accumulated regret for each case denoted by $\mathcal{R}_i$ for all $i \in [4]$ in Section A.4.

(1) We first estimate $\mathcal{R}_1$. In this case, we accumulate regret only when running CHALLENGE with $a^+ = a^*$.

The CHALLENGE breaks if either S_t hits $+1$ or $-s$. To simplify the calculation, we only estimate the time when S_t hits $+1$ in the following analysis. By applying Theorem 3.3.4 to S_t with $\mathrm{E}_t(S_{t+1} - S_t) \geq \Delta > 0$, we have τ_{+1} is at most $\frac{\delta(1-S_0)}{\Delta}$ with

probability at least $1 - \exp\left(-\frac{\delta(1-S_0)}{e\cdot 1}\right) \geq 1 - e^{-\delta/e}$ for any $\delta > 0$. In other words, a CHALLENGE lasts at most $\frac{\delta(1-S_0)}{\Delta}$ with probability at least $1 - e^{-\delta/e}$ for any $\delta > 0$. Note that the regret per CHALLENGE is $1 \cdot \frac{\delta(1-S_0)}{\Delta} = \frac{\delta(1-S_0)}{\Delta}$. From Algorithm 7, we start a CHALLENGE with probability $\sqrt{\frac{L}{T}}$. We denote the number of CHALLENGE by Z and Z is subject to a binomial distribution $\text{Bin}(T, \sqrt{\frac{L}{T}})$. By a Chernoff Bound, we obtain for any $\delta > 0$,

$$\Pr\left(Z \geq (1+\delta)\sqrt{LT}\right) \leq e^{-\delta^2\sqrt{LT}/(2+\delta)}. \tag{3.5}$$

Using the union bound with $\{\mathcal{R}_1 \geq \delta(1+\delta)(1-S_0)\sqrt{LT}\} \subseteq \{Z \geq (1+\delta)\sqrt{LT}\} \cup \{\tau_{+1} \geq \frac{\delta(1-S_0)}{\Delta}\}$, we can derive

$$\Pr\left(\mathcal{R}_1 \leq \delta(1+\delta)\frac{(1-S_0)}{\Delta}\sqrt{LT}\right) \geq 1 - e^{-\frac{\delta}{e}} - e^{-\frac{\delta^2\sqrt{LT}}{2+\delta}}. \tag{3.6}$$

(2) Next, we estimate $\mathcal{R}_2$. Recall an era is defined in Definition 3.6.3. As CHALLENGE breaks if the S_t hits either $+1$ or $-s$. To simplify the calculation, we only estimate the time when S_t hits $-s$ in the following analysis. By applying Theorem 3.3.4 to $X_t = s + S_t$ (for the case that $\tau_{-s} < \tau_{+1}$). The time a CHALLENGE breaks is, at most $\frac{(s+S_0)\delta}{\Delta}$ with probability at least $1 - \exp\left(-\frac{((s+S_0))\delta}{es}\right) \geq 1 - e^{-\delta/e}$ for any $\delta > 0$. Note that in each era, the regret accumulated is at most $1 \cdot \frac{\delta(s+S_0)}{\Delta} = \frac{\delta(s+S_0)}{\Delta}$. And we have L eras as defined. So we derive with $S_0 \in (-s, 1)$ and $s = \sqrt{T/L}$,

$$\Pr\left(\mathcal{R}_2 \leq \frac{\delta(s+S_0)}{\Delta}L\right) \geq 1 - e^{-\delta/e} - e^{-\frac{\delta^2\sqrt{LT}}{2+\delta}}. \tag{3.7}$$

(3) We then estimate $\mathcal{R}_3$. Recall the sub-era, is defined in Definition 3.6.3. To simplify our analysis, we overestimate $\mathcal{R}_3$ by assuming we accumulate regret at most 1 at each step during this phase. We define K to be the number of such steps in which we start a new CHALLENGE that ends the sub-era when no CHALLENGE is active. We intend to show for any $\delta > 0$, we have

$$\Pr\left(K \geq \frac{12\delta\sqrt{T/L}}{\Delta}\right) \leq e^{-\Omega(\delta)}. \tag{3.8}$$

To see Eq. (3.8) holds, note that by Lemma 1.1 (ii) in Larcher et al., 2023, each CHALLENGE has probability at least $\frac{\Delta}{12}$ of reaching $-s$ before $+1$ and thus end the sub-era. Recall that RWAB runs CHALLENGE with probability $\sqrt{L/T}$ at each step. So, the probability of starting a new CHALLENGE that ends the sub-era, when no CHALLENGE is active, is at least $p := \frac{\Delta}{12}\sqrt{L/T}$. We can see $K \sim \text{Geo}(p_K)$ where $p_K \geq p$. We define $K' \sim \text{Geo}(p)$. By Lemma A.4.2, we have $K' \succeq K$, and thus $\text{E}(K) \leq \text{E}(K') = 1/p$. We compute the following by using the fact that $K' \succeq K$. For any $r > 0$,

$$\Pr\left(K \geq \frac{r}{p}\right) \leq \Pr\left(K' \geq \frac{r}{p}\right)$$

We set $q := 1 - p \in (0, 1)$ and $K' \sim \text{Geo}(p)$.

$$\leq \sum_{k=r/p}^{\infty} q^k p$$

Geometric series gives

$$= p \cdot \frac{q^{r/p}}{1-q}.$$

This gives

$$= q^{r/p}.$$

Using $q^\alpha = e^{\alpha \ln(q)}$ gives

$$= e^{r \ln(q)/p}$$
$$= \exp\left(-\frac{\ln\left(\frac{1}{1-p}\right)}{p} r\right)$$

We define $f(p) := \frac{\ln\left(\frac{1}{1-p}\right)}{p}$ and note that $f(p) > 1$ for any $p \in (0, 1)$. Thus, we can conclude that

$$\leq e^{-\delta}.$$

This completes the proof of Eq. (3.8). Then, the number of such steps K is thus at most $\frac{12\delta\sqrt{T/L}}{\Delta}$ with probability $1-e^{-\delta}$, which is exactly Eq. (3.8). So the regret contributing to $\mathcal{R}_3$ is bounded by $1\cdot\frac{12\delta\sqrt{T/L}}{\Delta}$ for each sub-era with probability at least $1-e^{-\delta}$.

For the number of sub-era M, from Lemma 2.1 in Larcher et al., 2023, we have $M \leq 2L+2N$ where L is the number of changes and N is the number of mistakes[2]. Note that at each step there is a probability $\sqrt{L/T}$ of starting a CHALLENGE and by Lemma 1.1(i) in Larcher et al., 2023, "this CHALLENGE has probability $2\sqrt{L/T}$ of ending with a mistake". Thus, N is stochastic dominated by $\mathrm{Bin}(T, \sqrt{L/T}\cdot 2\sqrt{L/T}) = \mathrm{Bin}(T, \frac{2L}{T})$. By a Chernoff bound, we have for any $\delta > 0$,

$$\Pr\left(N \geq (1+\delta)2L\right) \leq e^{-2L\delta^2/(2+\delta)}.$$

Note that $M \leq 2L+2N$ implies that $\{N \leq (1+\delta)2L\} \subseteq \{M \leq 2L+4L(1+\delta)\}$. We deduce that

$$\begin{aligned}\Pr\left(M \leq 2L+4L(1+\delta)\right) &\geq \Pr\left(N \leq (1+\delta)2L\right)\\ &\geq 1-e^{-2L\delta^2/(2+\delta)}. \end{aligned} \tag{3.9}$$

Using $\mathcal{R}_3 = M \cdot \text{Regret/per sub-era}$, we can obtain

$$\Pr\left(\mathcal{R}_3 \geq 2L(3+2\delta)\cdot\frac{12\delta\sqrt{T/L}}{\Delta}\right)$$

Notice that $\{\mathcal{R}_3 \geq 2L(3+2\delta)\cdot 12\delta\sqrt{T/L}\} \subseteq \{M \geq 2L(3+2\delta)\} \cup \{\text{Regret/per sub-era} \geq \frac{12\delta\sqrt{T/L}}{\Delta}\}$. We denote Regret/per sub-era by Q. Using Union bound gives

$$\begin{aligned}&\leq \Pr\left(M \geq 2L(3+2\delta)\right)\\ &\quad + \Pr\left(Q \geq \frac{12\delta\sqrt{T/L}}{\Delta}\right)\end{aligned}$$

[2]This is defined in Definition 3.6.2.

Using Eq. (3.8) and Eq. (3.9) gives

$$\leq e^{-2L\delta^2/(2+\delta)} + e^{-\delta}.$$

In other words, we obtain

$$\Pr\left(\mathcal{R}_3 \leq 24\delta(3+2\delta)\frac{\sqrt{LT}}{\Delta}\right) \geq 1 - e^{-2L\delta^2/(2+\delta)} - e^{-\delta}. \tag{3.10}$$

(4) We finally estimate $\mathcal{R}_4$. In this case, "each sub-era with $a^+ \neq a^*$ consists of several CHALLENGE which hit the $+1$, but one hits $-s$ to end the sub-era. Since this happens, a^+ is swapped, and then by definition, the sub-era ends Larcher et al., 2023". Consider the steps in which a random walk with negative drift $-\Delta$ towards $-s$ is reset to 0 whenever it goes above $+1$. By applying Theorem 3.3.4 to estimate the τ_{-s} with drift Δ towards $-s$, we have the total number of steps spent inside each sub-era is at most $\frac{\delta(s+S_0)}{\Delta}$ with probability $1 - e^{-\delta(s+S_0)/e}$ for any $\delta > 0$. Since each step costs at most 1, the regret is at most $\frac{\delta(s+S_0)}{\Delta}$ with probability $1 - e^{-\delta(s+S_0)/e}$. Recall that we derive the tail bound for the number of sub-era M in Eq. (3.9) and that $\mathcal{R}_4 = M \cdot \text{Regret/per sub-era}$. Using Union bound with $\{\mathcal{R}_4 \geq 2\delta(3+2\delta)(s+S_0)L\} \subseteq \{M \geq (3+2\delta)2L\} \cup \{\text{Regret/per sub-era} \geq \frac{\delta(s+S_0)}{\Delta}\}$ and Eq. (3.9), we obtain

$$\Pr\left(\mathcal{R}_4 \leq 2\delta(3+2\delta)\frac{(s+S_0)}{\Delta}L\right) \geq 1 - e^{-\frac{2L\delta^2}{2+\delta}} - e^{-\frac{\delta(s+S_0)}{e}}. \tag{3.11}$$

By taking $\delta = \sqrt{\Delta\varepsilon}$ for any $\varepsilon \geq 1$ in Eq. (3.6), we can deduce that $\delta(1+\delta) = \delta + \delta^2 \leq 2\varepsilon$ since Δ is some constant in $(0, 1]$. Thus, we can obtain

$$\Pr\left(\mathcal{R}_1 \leq 2\varepsilon\sqrt{LT}\right) \geq 1 - 2e^{-\sqrt{\varepsilon}/e}$$

Using $\delta = \sqrt{\Delta\varepsilon} \leq \varepsilon$ for $\varepsilon \geq 1$ in Eq. (3.7) gives

$$\Pr\left(\mathcal{R}_2 \leq \varepsilon(L + \sqrt{LT})\right) \geq 1 - 2e^{-\sqrt{\varepsilon}/e}$$

Using $\delta(3+2\delta) = 3\delta + 2\delta^2 \leq 5\varepsilon$ in Eq. (3.10) gives

$$\Pr\left(\mathcal{R}_3 \leq 120\varepsilon\sqrt{LT}\right) \geq 1 - 2e^{-\sqrt{\varepsilon}}$$

Using $2\delta(3+2\delta) = 6\delta + 4\delta^2 \leq 10\varepsilon$ in Eq. (3.11) gives

$$\Pr\left(\mathcal{R}_4 \leq 10\varepsilon(L+\sqrt{LT})\right) \geq 1 - 2e^{-\sqrt{\varepsilon}/e}.$$

Above all, the total regret $\mathcal{R}$ consists of $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$. We define the following event:

$$\begin{aligned} E &:= \{\mathcal{R} \leq 480\varepsilon(L+\sqrt{LT})\} \\ &= \{\sum_{i=1}^{4} \mathcal{R}_i \leq 480\varepsilon(L+\sqrt{LT})\} \end{aligned}$$

Considering the complement of Event E and using the tail bound for each $\mathcal{R}_i$ give

$$\begin{aligned} &\Pr\left(\mathcal{R} > 480\varepsilon(L+\sqrt{LT})\right) \\ \leq &\Pr\left(\text{At least one } \mathcal{R}_i > \frac{480\varepsilon(L+\sqrt{LT})}{4}\right) \end{aligned}$$

Using tail bound for regret of each stage, we obtain

$$\begin{aligned} &\leq \max\{2e^{-\sqrt{\varepsilon}/e}, 2e^{-\sqrt{\varepsilon}}\} \\ &= 2e^{-\sqrt{\varepsilon}/e}. \end{aligned}$$

□

From Theorem 3.6.1, we can say that RWAB Algorithm has regret at most order $O\left(\sqrt{LT}\right)$ for a 2-armed non-stationary bandit problem w.h.p. By using minimax lower bound (Bubeck, Cesa-Bianchi, 2012), any algorithm on K-armed stationary bandit problems has regret at least $\frac{\sqrt{Kn}}{20}$ for time horizon n. In particular, for $K = 2$, the lower bound for the expected regret of any algorithm on bandit stationary bandit problems is $\Omega(\sqrt{n})$ for time horizon n. It has been shown in (Larcher et al., 2023) that by considering L changes and each change of average steps T/L, RWAB has expected regret at least $\Omega\left(L \cdot \sqrt{T/L}\right) = \Omega\left(\sqrt{LT}\right)$. Theorem 3.6.1 confirms that RWAB is optimal with overwhelmingly high probability, and we propose a new perspective of analysing such a bandit

algorithm by using drift analysis, which is rarely employed by the reinforcement learning community.

3.7 Experiments

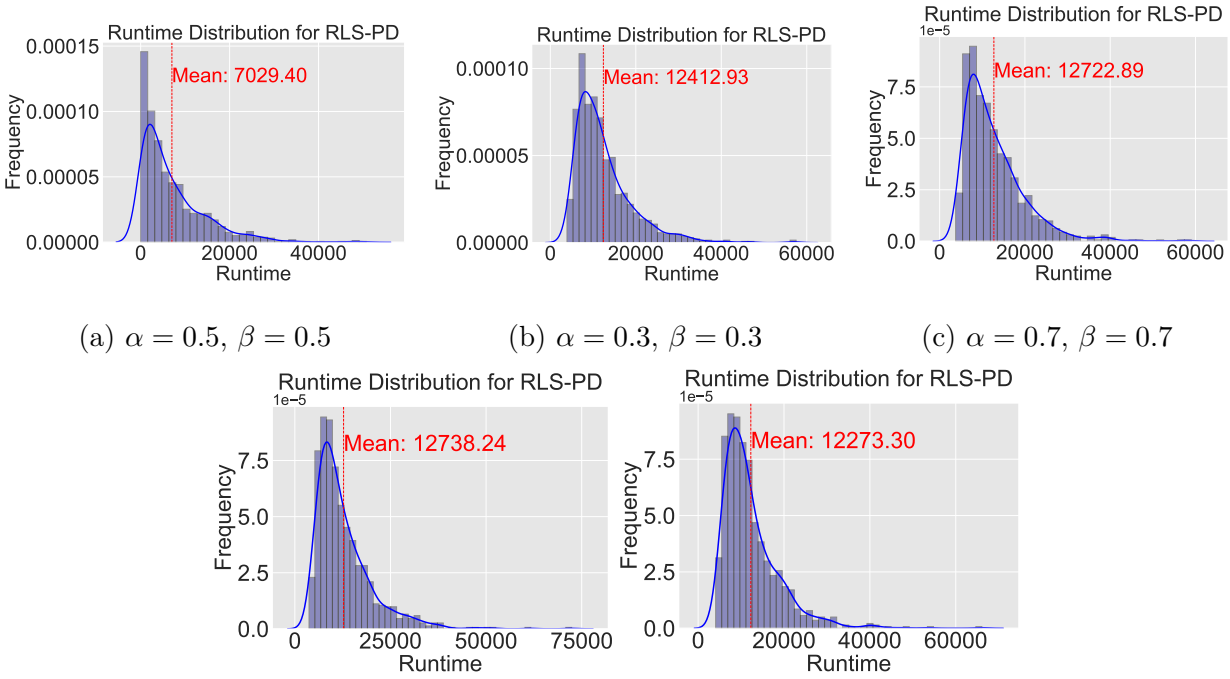
To complement our asymptotic results with data for concrete problem sizes, we conduct the following experiments.

3.7.1 Empirical Investigation of RLS-PD on Bilinear

Settings: We conduct experiments with RLS-PD for the Maximin Bilinear problem. The problem setup is $(\alpha, \beta) \in \{(0.5, 0.5), (0.3, 0.3), (0.3, 0.7), (0.7, 0.3), (0.7, 0.7)\}$. These five scenarios can cover the cases when the optimum lies in four different quadrants and the centre of the search space. We set the default mutation rate $\chi = 1$ and problem size $n = 1000$. We run 1000 independent simulations for each configuration. For each run, we initialise the search point uniformly at random.

Results: As we can see, Figure 3.1 (on Page 91) is the density plot for the runtime distribution. The x-axis represents the runtime, the y-axis represents the frequency or density, and the red dot line represents the average value of runtime for each problem setting. As x increases, Figure 3.1 shows that we have an exponential decay tail for the runtime of RLS-PD on each problem configuration. It is very unlikely that the runtime of RLS-PD on Bilinear deviates too much from the mean or the expected runtime from Figure 3.1 for each problem configuration. From Table 3.1, we can see for each configuration, the frequency that the actual runtime bounded above by the mean runtime is asymptotic to 1 as the increases in the multiplicative factors of the upper bounds. When $(\alpha, \beta) = (0.5, 0.5)$, the empirical results are consistent with our theoretical bounds. The results for other prob-

lem configurations raise a conjecture about whether our theoretical results can also hold for all $\alpha, \beta \in [0, 1]$.



(a) $\alpha = 0.5$, $\beta = 0.5$ (b) $\alpha = 0.3$, $\beta = 0.3$ (c) $\alpha = 0.7$, $\beta = 0.7$

(d) $\alpha = 0.3$, $\beta = 0.7$ (e) $\alpha = 0.7$, $\beta = 0.3$

Figure 3.1: Runtime distribution for RLS-PD for various α and β.

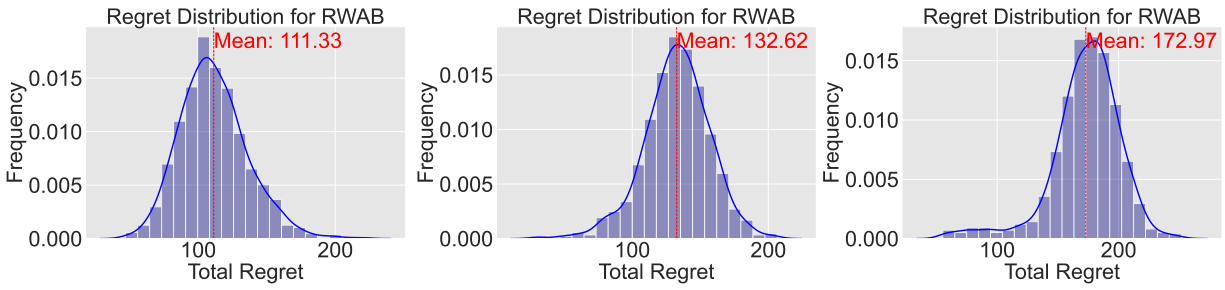
3.7.2 Empirical Investigation of RWAB Algorithm

Settings: We conduct experiments of RWAB Algorithm for 2-armed non-stationary bandit problem. The environment is set up as two Bernoulli bandits with mean $\mu_1 = 0.2, \mu_2 = 0.8$ and the number of changes $L = 5, 10, 20, 40, 80, 100$. The changes are set up uniformly at random along the time horizon $T = 1000$. 1000 independent simulations are run for each configuration.

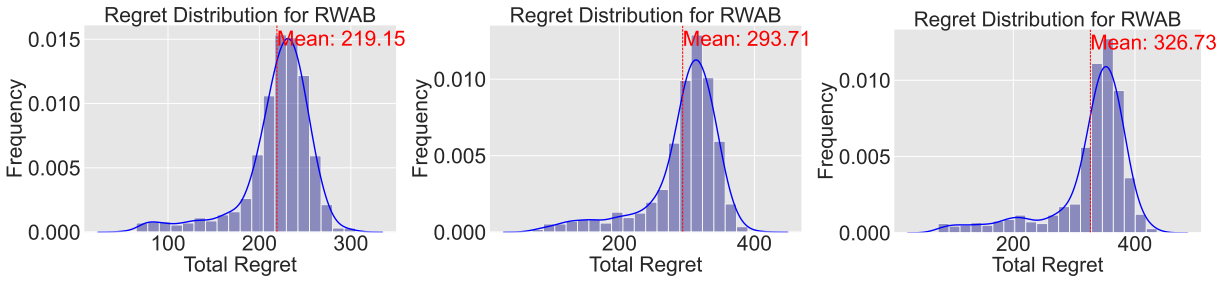
Results: Figure 3.2 displays the regret distribution which presents the performance distribution of RWAB. The x-axis represents the regret of RWAB, the y-axis represents the frequency or density and the red dot line represents the average value of regret for each

Table 3.1: Runtime statistics for RLS-PD on Bilinear ($n = 1000$). T denotes the actual runtime of RLS-PD on Bilinear on each run. $\bar{T}$ denotes the empirical mean of the runtime and Fr denotes the frequency of runtimes.

Problem Configuration	$\bar{T}$	**Fr** $(T \leq \bar{T})$	**Fr** $(T \leq 2\bar{T})$	**Fr** $(T \leq 4\bar{T})$	**Fr** $(T \leq 6\bar{T})$	**Fr** $(T \leq 8\bar{T})$
$\alpha = 0.5, \beta = 0.5$	**7029.405**	0.633	0.845	0.983	0.996	**1.0**
$\alpha = 0.3, \beta = 0.3$	**12412.929**	0.63	0.945	0.998	**1.0**	1.0
$\alpha = 0.7, \beta = 0.7$	**12722.888**	0.604	0.95	0.997	**1.0**	1.0
$\alpha = 0.3, \beta = 0.7$	**12738.235**	0.627	0.935	0.997	**1.0**	1.0
$\alpha = 0.7, \beta = 0.3$	**12273.301**	0.641	0.95	0.998	**1.0**	1.0



(a) $T = 1000, L = 5$ (b) $T = 1000, L = 10$ (c) $T = 1000, L = 20$



(d) $T = 1000, L = 40$ (e) $T = 1000, L = 80$ (f) $T = 1000, L = 100$

Figure 3.2: Regret distribution for various values of T and L.

Table 3.2: Regret statistics for Rwab ($T = 1000$). R denotes the actual regret of Rwab on 2-armed non-stationary Bernoulli Bandits on each run. $\bar{R}$ denotes the empirical mean of the regret and Fr denotes the frequency of regrets.

Problem Configuration	$\bar{R}$	**Fr** $(R \leq \bar{R})$	**Fr** $(R \leq 1.2\bar{R})$	**Fr** $(R \leq 1.4\bar{R})$	**Fr** $(R \leq 1.6\bar{R})$	**Fr** $(R \leq 1.8\bar{R})$	**Fr** $(R \leq 2\bar{R})$
$L = 5$	**111.329**	0.543	0.829	0.949	0.990	0.998	**1.0**
$L = 10$	**132.621**	0.491	0.871	0.988	**1.0**	1.0	1.0
$L = 20$	**172.975**	0.457	0.920	0.996	**1.0**	1.0	1.0
$L = 40$	**219.151**	0.383	0.949	**1.0**	1.0	1.0	1.0
$L = 80$	**293.709**	0.343	0.945	**1.0**	1.0	1.0	1.0
$L = 100$	**326.728**	0.299	0.958	**1.0**	1.0	1.0	1.0

problem setting. As x increases, Figure 3.2 shows that we have an exponential decay tail for the regret of RWAB on 2-armed non-stationary bandit problem with various change L. Figure 3.2 shows the concentration of regret around the empirical mean or the expected regret and it is also unlikely that the regret of RWAB deviates too much from the expectation. This suggests that our theoretical tail bound is asymptotically tight regardless of the leading coefficient. Table 3.2 shows that for each configuration, the frequency that the actual regret bounded above by the mean regret is asymptotic to 1 as the increases in the multiplicative factors of the upper bounds. Compared with Table 3.1, the convergence rate is significantly faster than the counterpart in the case of RLS-PD on BILINEAR. This means that the theoretical bound (i.e. leading coefficient) obtained has room to improve. One conjecture may be the process governing the dynamics of runtime for RLS-PD on BILINEAR relies heavily on high variance to overcome the negative drift, while the process governing the dynamics of regret induced by RWAB already exhibits positive drift everywhere before reaching the target state. Thus, it yields a faster convergence.

3.8 Conclusion

This chapter proves a more general and stronger drift theorem (tail-bound). This provides a more straightforward and novel perspective with higher precision. Such a method can be used to analyse different random processes. As a sub-product, this paper also resolves the open problem left in (Kötzing, 2016), which looks for a suitable replacement for the Azuma-Hoeffding inequality to improve the tail bounds for random processes. More significantly, we apply our theorems in several practical examples, including Random 2-SAT, Recolouring, Competitive CoEAs and RWAB. To the best of our knowledge, it is the first tail-bound drift analysis of CoEA and bandit learning algorithms. Our drift theorems provide more precise information on how the runtime concentrates and provide a stronger performance guarantee. In practice, it shows the limitation of the current coevolutionary algorithm on adversarial optimisation. It suggests a need for a deeper understanding of the mechanism of CoEAs, which may help to design a more stable CoEA. Moreover, our results confirm that randomness in RWAB can be helpful for stochastic non-stationary bandit problems.

For future study, both runtime analysis of CoEA on adversarial optimisation and regret analysis of stochastic reinforcement learning algorithms via drift analysis are still unexplored areas. In particular, on the technical side, can we derive more precise bounds for RWAB since the leading coefficient seems not to be optimal from empirical results or can we use these results to analyse more complicated CoEAs or bandit algorithms? On the practical side, we could try to use such concentration bound to design more efficient algorithms. For example, we could try to design more stable CoEAs or develop a general optimal bandit algorithm like RWAB for more than 2-armed non-stationary bandit problems by using the random-walk design analysed in this work.

Chapter Four

No Free Lunch Theorem for Black-Box Adversarial Optimisation

This chapter is based on the following publication:

No Free Lunch Theorem and Black-Box Complexity Analysis for Adversarial Optimisation (Lehre and Lin, 2024b) which is published in Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS'24).

4.1 Introduction

Black-Box Optimisation (BBO) is crucial for optimising complex, unknown, or expensive-to-evaluate functions in real-world scenarios, such as aerodynamic design and hyperparameter tuning, where only input-output observations are available. Formally, BBO is the task of optimising objective functions from some function class $\mathcal{F}$ where $\mathcal{F}$ consists of functions $f : \mathcal{X} \to \mathbb{R}$, where the algorithm is limited to making queries to f (Droste et al., 2002b; Golovin et al., 2017; Sarafian et al., 2020). In this case, the algorithm is only able to sample and query the function value $f(x)$ of search points $x \in \mathcal{X}$ from a "black-box" or "oracle" without access to any description of the objective functions f. A similar framework can be extended to game-theoretic BBO, namely adversarial BBO. Specifically, adversarial BBO is the task of optimising payoff functions from some payoff function class $\mathcal{G}$ where $\mathcal{G}$ consists of black-box functions $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, with a limited budget of function evaluations. Additionally, two players $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ form another class of optimisation problems with multiple inputs (as presented in Figure 4.1). In particular, this dissertation will focus on a specific class of optimisation problems called adversarial optimisation, where two players compete with each other (one wants to maximise the payoff g and the other one wants to minimise the payoff, and thus call it adversarial optimisation).

The original No Free Lunch Theorems for traditional optimisation can be summarised as follows:

> "For all possible metrics, no search algorithm or supervised learning algorithm is better than another when its performance is averaged over all possible problem instances (Wolpert, 2002; Wolpert and Macready, 1997)."

Wolpert and Macready (1997) and Wolpert (2002) have shown the facts about the usefulness of traditional black-box optimisation algorithms, including various randomised search heuristics (such as evolutionary algorithms and simulated annealing) and machine learning

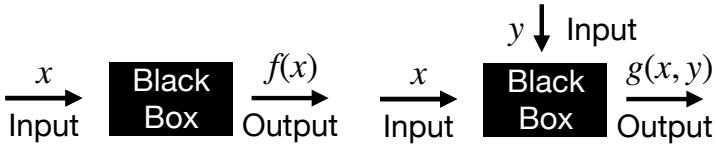


Figure 4.1: Comparison between traditional black-box optimisation and maximin black-box optimisation. Instead of querying at x in traditional optimisation, maximin optimisation queries at (x, y) include both strategy x and the best response y from the opponent, i.e. $\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)$. Their interaction is converted to the payoff $g(x, y)$ in the given black-box model.

algorithms (such as supervised learning). In particular, it shows that the performance of all black-box optimisation algorithms (Wolpert and Macready, 1997) and learning algorithms (Wolpert, 2002), when averaged over all problem instances, is the same for any maximisation or minimisation tasks. Considering all possible problem instances is a rather limited assumption for the original NFL theorem to hold, and in most realistic scenarios, we do not consider all possible problem instances. Thus, Droste et al. (2002b) later has provided a generalised NFL theorem with more realistic scenarios by relaxing the NFL theorem holds from all problem instances to problem instances closed under permutation. Another seminal work by Schaffer (1994) also showed a conservation law for the generalised performance of learning algorithms in classification problems.

Adversarial optimisation tasks, such as maximin optimisation, are more complex and often counter-intuitive compared to traditional optimisation problems. In adversarial settings, defining solution concepts, or establishing what is meant by optimality, is essential. Common solution concepts include 'maximisation over all test cases', 'maximin', and Nash Equilibrium. A 'free lunch' in adversarial optimisation implies that, for a given solution concept, some algorithms consistently outperform others when averaged across all possible problem instances. This phenomenon was demonstrated by Wolpert and Macready (2005) for adversarial optimisation with respect to the 'maximin' solution concept (Defini-

tion A.5.2). Similarly, Service and Tauritz (2008) established a 'free lunch' result for the 'maximisation over all test cases' concept (Definition A.5.1).

Beyond 'maximin' and 'maximisation over all test cases', Nash Equilibrium is another widely studied solution concept in adversarial learning and maximin optimisation. This concept is central to various applications, including adversarial learning models such as GANs (Goodfellow et al., 2014; Heusel et al., 2017) and spatial games (Hemberg et al., 2021; Lehre, 2022; Lehre et al., 2023). While previous studies by Wolpert and Macready (2005) and Service and Tauritz (2008) have demonstrated the existence of 'free lunches' in adversarial optimisation, the following key questions remain open:

(1) Does adversarial optimisation exhibit a 'free lunch' for all possible solution concepts?

(2) If we use Nash Equilibrium as the solution concept, how can we characterise the difficulty of black-box adversarial optimisation problems for problem-independent, possibly randomised, search heuristics?

In this dissertation, we answer Question (1) in the negative by showing a No Free Lunch theorem for Nash Equilibrium solution concept and address Question (2) by introducing Black-Box Complexity (BBC) tools. Moreover, there are some differences in our cost model: we only charge one unit of cost per row and column evaluated in the payoff matrix.

4.1.1 Contribution

We prove a new impossibility result on black-box adversarial optimisation. In a two-player zero-sum game setting, there is no free lunch with respect to an approximation of the real cost model regarding the unique Nash Equilibrium as the solution concept. In other words, all black-box adversarial optimisation algorithms have the same expected performance over a uniform distribution of all possible problem instances with the unique Nash

Equilibrium as the solution concept. It is the first step to resolve this long-standing open research problem about NFL in general black-box adversarial optimisation since (Wolpert and Macready, 2005; Wolpert and Macready, 1997). Our results highlight the significance of the choice of solution concepts and the limitation of general black-box adversarial optimisation. Additionally, we introduce the first general black-box model and the notion of black-box complexity for adversarial optimisation. Under this general black-box model, we provide the general lower bounds for query complexity of computing Nash Equilibrium in adversarial optimisation. Finally, we illustrate our results on examples of computing unique Nash Equilibrium in two-player zero-sum games.

4.1.2 Challenges and Technical Overview of the NFL and BBC Results

There are several challenges in showing the NFL and BBC results. First, we highlight the challenges and our technical details in the derivation process compared to previous NFL work. The classical No Free Lunch theorem applies proof by induction with respect to the size of the function domain. A natural idea is to extend the previous proof-by-induction method in our NFL proof. However, one of the most challenging aspects of deriving NFL for Nash Equilibrium (NE) is that the adversarial setting significantly increases the difficulty since the problem depends on the performances of both the player and the opponents. To address this, we introduce two technical lemmas from game theory (i.e. Lemma A.5.1 and Lemma A.5.2) and help to construct the isomorphism between two different problem classes (i.e. Corollary 4.3.1 and Lemma 4.3.2), which is an essential step in the proof of the original NFL theorem for traditional optimisation (see Section A.5.2 for more details). Another significant challenge is that we start with a very weak assumption: we only allow access to the payoff function and make no assumptions about its properties, such as convexity, continuity, or differentiability. In fact, no gradients, continuity, or differentiability make the

NFL and black box results easier to prove (i.e., the function class lacks a specific structure). On the other hand, it makes it more difficult for the algorithm. Consequently, we have limited analytical tools to proceed. To analyse the black-box complexity of adversarial optimisation, we introduce Yao's minimax principle and apply our new NFL result.

4.2 Preliminaries

4.2.1 Notation in This Chapter

Given a finite set $\mathcal{X}$, we denote the permutation group by $\mathfrak{S}(\mathcal{X}) := \{\sigma : \mathcal{X} \to \mathcal{X} \mid \sigma \text{ is a permutation of } \mathcal{X}\}$. For any $v \in \mathbb{R}^n$, let $\textsc{Supp}(v) := \{i \in [n] \mid v_i \neq 0\}$. We refer to query complexity or runtime as the number of payoff function evaluations until the given algorithm finds the optimum. We consider the search space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are finite. $X \simeq Y$ denotes the isomorphism between X and Y In this section, we define an "isomorphism" following (Droste et al., 2002b) rather than defining it as a usual group or ring isomorphism since we do not require any group or ring structure of $\mathcal{F}$ in the proof. In other words, $X \simeq Y$ if there exists a one-to-one correspondence map from X to Y.

Let $f : X \to Y$ be a function from a set X to a set Y. If A is a subset of X, then the restriction of f to A is the function: $f|_A : A \to Y; x \mapsto f(x)$. Let $f : X \to Y$ be any function and A and B be sets such that $X \subseteq A$ and $Y \subseteq B$. An extension of f to A is a function $g : A \to B$ such that $f(x) = g(x)$ for all $x \in X$. Alternatively, g is an extension of f to A if f is a restriction of g to X.

In this dissertation, we assume that the black-box optimisation algorithms do not make the same query twice. This can be achieved by memorising the outcome of previous queries. This assumption is consistent with previous work (Wolpert and Macready, 2005; Wolpert and Macready, 1997).

4.2.2 Solution Concepts

Solution concepts for classical function optimisation are not directly applicable to adversarial optimisation in general-sum or zero-sum game settings (Hu, Wellman, 1998; Osborne and Rubinstein, 1994; Popovici et al., 2012). Each agent or player's payoff depends on not only its action but also the response from its opponents, and thus, we need to introduce different solution concepts to specify what kind of optimum we look for.

Pure Strategy Nash equilibrium is considered as our solution concept in this work. We are interested in whether the black-box adversarial optimisation can efficiently find any given game's Nash equilibrium. We use the formulation in (Nisan et al., 2007) to define Nash equilibrium rigorously. In this dissertation, we only focus on Pure Strategy Nash Equilibrium (abbrev. NE) defined in Definition 2.3.1.

We also defer other solution concepts for comparison in the appendix. We use the formulation in (Service and Tauritz, 2008; Wolpert and Macready, 2005).

4.3 NFL Theorem for Computing Nash Equilibrium in Adversarial Optimisation

In this section, we prove a No Free Lunch Theorem for black-box adversarial optimisation. As a first step toward the No Free Lunch Theorem, we consider two-player zero-sum games with the unique NE as the same setting in (Panageas et al., 2023), but we relax the restriction from potential games to general two-player zero-sum games. The original NFL theorem for traditional optimisation assumed all problems instances (Wolpert and Macready, 1997), and later, sharpened work also proved that this holds for functions 'closed under permutation' (Droste et al., 2002b; Igel and Toussaint, 2005; Schumacher et al., 2001). We now explain what 'closed under permutation' means and how it can be extended to games.

The permutation closure of a set of functions means that let $\mathcal{X}, \mathcal{Y}$ be two finite sets and $f : \mathcal{X} \to \mathcal{Y}$ defined by $f(x_i) = y_i$. Let σ be a permutation $\sigma : \mathcal{X} \to \mathcal{X}$ and we can permute the function: $f(\sigma(x)) := f \circ \sigma(x)$. Schumacher et al. (2001) and Droste et al. (2002b) defined 'closed under permutation (c.u.p.)' with respect to a single search space $\mathcal{X}$. A class of functions $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$ is called c.u.p. if for all $f \in \mathcal{F}$ and all permutations $\sigma \in \mathfrak{S}(\mathcal{X}), f \circ \sigma \in \mathcal{F}$. For our adversarial setting, we need to extend this notion to $\mathcal{X} \times \mathcal{Y}$. Given a payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, we define $g \circ (\sigma \otimes \tau)(x, y) := g(\sigma(x), \tau(y))$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and any permutations on $\mathcal{X}, \mathcal{Y}$ denoted by $\sigma \in \mathfrak{S}(\mathcal{X}), \tau \in \mathfrak{S}(\mathcal{Y})$.

Definition 4.3.1. *Let $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{F}, \textsc{Nash})$ be a class of two-player zero-sum games, where $\mathcal{F}$ is a subset of all the payoff functions in these games $g : \mathcal{X} \times \mathcal{Y} \to O$, and $O \subseteq \mathbb{R}$. We say $\mathcal{F}$ is closed under permutation (c.u.p.) if for any $g \in \mathcal{F}$ and any permutations on $\mathcal{X}, \mathcal{Y}$ denoted by $\sigma \in \mathfrak{S}(\mathcal{X}), \tau \in \mathfrak{S}(\mathcal{Y})$, we have $g \circ (\sigma \otimes \tau) \in \mathcal{F}$.*

If a set of two-player zero-sum games is c.u.p., we will also call it structure-free. In this dissertation, we restrict to the case where $\mathcal{X}, \mathcal{Y}, O$ are finite sets following the settings in (Wolpert and Macready, 2005; Wolpert and Macready, 1997). The proof of the No Free Lunch theorem for traditional optimisation by Droste et al. (2002a) is by induction, where one assumes that the statement is true for all smaller problems. Here, a smaller problem refers to the following definition of a sub-game or a sub-problem class.

Definition 4.3.2. *Let $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{F}, \textsc{Nash})$ be a class of two-player zero-sum games, where $O \subseteq \mathbb{R}$ and $\mathcal{F}$ be any subset of the set of payoff functions in these games $g : \mathcal{X} \times \mathcal{Y} \to O$ such that $\mathcal{F}$ is closed under permutation. For any given $(x_1, y_1) \in \mathcal{X} \times \mathcal{Y}$, any function $b_1 : \mathcal{Y} \to O$, and any function $b_2 : \mathcal{X} \to O$, we define a sub-problem class $\mathcal{F}((x_1, y_1), (b_1, b_2))$ with respect to $\mathcal{F}$ as follows: $f \in \mathcal{F}((x_1, y_1), (b_1, b_2))$ if and only if there exists a function $g \in \mathcal{F}$ such that*

(1) $g(x_1, y) = b_1(y)$ for all $y \in \mathcal{Y}$;

(2) $g(x, y_1) = b_2(x)$ *for all* $x \in \mathcal{X}$*;*

(3) f *is a restriction of* g *on* $(\mathcal{X} \setminus \{x_1\}) \times (\mathcal{Y} \setminus \{y_1\})$*.*

Next, before we show that a sub-problem class $\mathcal{F}((x_1, y_1), (b_1, b_2))$ is c.u.p., we need a lemma to guarantee that after any permutation, the smaller game has the same unique Nash equilibrium as the larger game[1]. We provide a corollary to Lemma A.5.2.

Corollary 4.3.1. *Let* $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{F}, \textsc{Nash})$ *be a class of two-player zero-sum games, and* $|\mathcal{X}| = |\mathcal{Y}|$ *where* $\mathcal{F} \subseteq \{g : \mathcal{X} \times \mathcal{Y} \to O\}$ *be any set of payoff functions with a unique NE and* $\mathcal{F}$ *is c.u.p.. Then, for any payoff function* $g \in \mathcal{F}$*, let* $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ *denote the unique Nash Equilibrium of* g*. For any permutations over* $\mathcal{X}, \mathcal{Y}$ *denoted by* $\sigma \in \mathfrak{S}(\mathcal{X}), \tau \in \mathfrak{S}(\mathcal{Y})$*,* $g \circ (\sigma \otimes \tau) \in \mathcal{F}$ *exhibits the unique Nash Equilibrium* $(\sigma(x^*), \tau(y^*))$*. Moreover, if for any* $x_0 \neq x^*$ *and* $y_0 \neq y^*$*, the restriction of* g *on* $(\mathcal{X} \setminus \{x_0\}) \times (\mathcal{Y} \setminus \{y_0\})$*, denoted by* $g|_{(\mathcal{X}\setminus\{x_0\})\times(\mathcal{Y}\setminus\{y_0\})}$*, exhibits the unique Nash Equilibrium* (x^*, y^*)*.*

Proof. We employ the result in Maiti et al., 2023 (See Lemma A.5.2). For any permutations on $\mathcal{X}, \mathcal{Y}$ denoted by $\sigma \in \mathfrak{S}(\mathcal{X}), \tau \in \mathfrak{S}(\mathcal{Y})$, $g \circ (\sigma \otimes \tau) \in \mathcal{F}$ is defined as for any $x \in X, y \in \mathcal{Y}$,

$$g \circ (\sigma \otimes \tau)(x, y) := g\left(\sigma(x), \tau(y)\right).$$

We consider the payoff matrix defined by $P = (p_{i,j})$ where $p_{i,j} = g(x_i, y_j)$ with $x_i \in \mathcal{X}, y_j \in \mathcal{Y}$. We denote row index set by $I_{\mathcal{X}} \subseteq \mathbb{N}$ and column index set by $I_{\mathcal{Y}} \subseteq \mathbb{N}$. Notice that $|I_{\mathcal{X}}| = |\mathcal{X}| = |\mathcal{Y}| = |I_{\mathcal{Y}}|$. We denote the row index by $i \in I_{\mathcal{X}}$ and the column index by $j \in I_{\mathcal{Y}}$. $P \in \mathbb{R}^{|I_{\mathcal{X}}| \times |I_{\mathcal{Y}}|}$ is the payoff matrix of two-player zero-sum game g with a unique Nash Equilibrium (x^*, y^*). Let us denote the row index of x^* in the payoff matrix P by i_* and the column index of y^* by j_* in the payoff matrix P. So $(e_{i^*}, e_{j^*}) \in \Delta_n \times \Delta_n$ is the unique Nash Equilibrium in the form of probability vector.

[1] A further explanation for sub-problem classes can be found in Section A.5.4.

Now, given any permutation σ, τ, we consider a submatrix P' such that $P' = (p_{\sigma(i),\tau(j)})$ with a new row index set $I'_{\mathcal{X}} \subseteq \mathbb{N}$ and column index set $I'_{\mathcal{Y}} \subseteq \mathbb{N}$. Since σ only permutes around the row indices in $I_{\mathcal{X}}$, $I'_{\mathcal{X}} \simeq I_{\mathcal{X}}$ with the isomorphism σ. Similarly, we have $I'_{\mathcal{Y}} \simeq I_{\mathcal{Y}}$ with the isomorphism τ. Now, we can see that $\sigma(i^*) = \textsc{Supp}(e_{\sigma(i^*)}) \in I'_{\mathcal{X}}$ and $\tau(j^*) = \textsc{Supp}(e_{\tau(j^*)}) \in I'_{\mathcal{Y}}$, so $(\hat{p}, \hat{q}) \in \Delta_n \times \Delta_n$ is the unique Nash Equilibrium of P' (in the form of a probability vector) where $(\hat{p})_i = (e_{\sigma(i^*)})_i$ for all $i \in I'_{\mathcal{X}}$ and $(\hat{q})_j = (e_{\tau(j^*)})_j$ for all $j \in I'_{\mathcal{Y}}$. In other words, $(\hat{p}, \hat{q}) = (e_{\sigma(i^*)}, e_{\tau(j^*)})$. So by using Lemma A.5.2, we can conclude that $(\sigma(x^*), \tau(y^*)) \in \mathcal{X} \times \mathcal{Y}$ is the unique Nash Equilibrium of $g \circ (\sigma \otimes \tau)$.

For the second claim, we construct a submatrix Q such that $Q = (q_{i,j})$ where $q_{i,j} = g(x_i, y_j)$ with $x_i \in \mathcal{X} \setminus \{x_0\}, y_j \in \mathcal{Y} \setminus \{y_0\}$. We denote the row index set $I_{\mathcal{X}\setminus\{x_0\}}$ and the column index set $I_{\mathcal{Y}\setminus\{y_0\}}$. Since $x_0 \neq x_*$ and $y_0 \neq y_*$, $i^* = \textsc{Supp}(e_{i^*}) \in I_{\mathcal{X}\setminus\{x_0\}}$ and $j^* = \textsc{Supp}(e_{j^*}) \in I_{\mathcal{Y}\setminus\{y_0\}}$, by using Lemma A.5.2 again, we can conclude that (x^*, y^*) is the unique Nash Equilibrium of $g|_{(\mathcal{X}\setminus\{x_0\})\times(\mathcal{Y}\setminus\{y_0\})}$. This completes the proof. □

This permutation essentially swaps the rows and columns in the payoff matrix. Corollary 4.3.1 shows that the permutation of rows and columns does not change the optimum (Nash Equilibrium in any given payoff function). Moreover, if we remove the row and column in the given payoff function, which do not contain the unique Nash Equilibrium, then the unique Nash Equilibrium remains the same in the restricted sub-problem.

Next, we prove that the sub-problem class is closed under permutation.

Lemma 4.3.1. *If $\mathcal{F}$ is c.u.p., then $\mathcal{F}((x_1, y_1), (b_1, b_2))$ is also c.u.p..*

Proof. For any $\sigma' \in \mathfrak{S}(\mathcal{X} \setminus \{x_1\}), \tau' \in \mathfrak{S}(\mathcal{Y} \setminus \{y_1\})$ and any $g' \in \mathcal{F}((x_1, y_1), (b_1, b_2))$, we want to show that $g' \circ (\sigma'\tau') \in \mathcal{F}((x_1, y_1), (b_1, b_2))$. We consider the extensions of each permutation. We define $\sigma : \mathcal{X} \to \mathcal{X}$ by its restriction on $\mathcal{X} \setminus \{x_1\}$ as

$$\sigma|_{\mathcal{X}\setminus\{x_1\}} = \sigma' \text{ and } \sigma(x_1) = x_1.$$

We define $\tau : \mathcal{Y} \to \mathcal{Y}$ by its restriction on $\mathcal{Y} \setminus \{y_1\}$ is

$$\tau|_{\mathcal{X}\setminus\{y_1\}} = \tau' \text{ and } \tau(y_1) = y_1.$$

Now, let g be the extension of g' which satisfies (1), (2) and (3) of Definition 4.3.2 (such an extension g exists since we take $g' \in \mathcal{F}((x_1, y_1), (b_1, b_2))$ and it follows from the definition of sub-problem class). As $g \in \mathcal{F}$, so are $g \circ (\sigma\tau) \in \mathcal{F}$. Thus, we construct $g \circ (\sigma\tau)$ as an extension of $g' \circ (\sigma'\tau')$ and $g \circ (\sigma\tau)$ satisfies (1), (2) and (3) of Definition 4.3.2. So $g' \circ (\sigma'\tau') \in \mathcal{F}((x_1, y_1), (b_1, b_2))$. □

Then, we want to know if we choose different (x, y) in the sub-problem class, are they still isomorphic (i.e., essentially the same problem for any black-box optimisation algorithm)? This is an essential step for applying proof by induction, since if they are not isomorphic, it means that choosing different search points and evaluation functions can change the game structure for black-box optimisation algorithms, and thus, there might exist some superior algorithms that can outperform the others.

Lemma 4.3.2. *For all $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$ and $b_1 : \mathcal{Y} \to \mathbb{R}$, $b_2 : \mathcal{X} \to \mathbb{R}$, we have the isomorphism:*

$$\mathcal{F}((x_1, y_1), (b_1, b_2)) \simeq \mathcal{F}((x_2, y_2), (b_1, b_2)).$$

Proof. To prove the isomorphism, we need to find a bijection between these two sets. We consider the following map between sets defined by:

$$\phi : \mathcal{F}((x_1, y_1), (b_1, b_2)) \to \mathcal{F}((x_2, y_2), (b_1, b_2)) \text{ where } \phi(g') = h' \text{ such that}$$

for $(x, y) \in (\mathcal{X} \setminus \{x_1, x_2\}) \times (\mathcal{Y} \setminus \{y_1, y_2\})$,

$$h'(x, y) = \phi(g')(x, y) = g'(x, y).$$

For $y \in \mathcal{Y} \setminus \{y_2\}$, $h'(x_1, y) = \phi(g')(x_2, y) = g'(x_2, y)$.
For $x \in \mathcal{Y} \setminus \{x_2\}$, $h'(x, y_1) = \phi(g')(x, y_2) = g'(x, y_2)$ Let h be the extension of h' defined by

$$h(x_2, y) = b_1(y), h(x, y_2) = b_2(x)$$

For all $(x, y) \in (\mathcal{X} \setminus \{x_2\}) \times (\mathcal{Y} \setminus \{y_2\})$,

$$h(x, y) = h'(x, y)$$

Now, we notice that $h = g \circ (\pi_x \pi_y)$ where π_x, π_y are two transpositions in $\mathfrak{S}(\mathcal{X}), \mathfrak{S}(\mathcal{Y})$ defined by

$$\pi_x(x_1) := x_2, \pi_x(x_2) := x_1; \pi_y(y_1) := y_2, \pi_y(y_2) := y_1$$

and for all $x' \in \mathcal{X} \setminus \{x_1, x_2\}$ and for all $y' \in \mathcal{Y} \setminus \{y_1, y_2\}$,

$$\pi_x(x') := x', \pi_y(y') := y'$$

Since $g \in \mathcal{F}$, so is $h = g \circ (\pi_x \pi_y) \in \mathcal{F}$. So h' has the extension h that satisfies (1), (2) in Definition 4.3.2. Also, $h' = \phi(g')$ is uniquely determined by g' and the argument would also hold when swapping h', g' with the map ϕ^{-1}. So we have proved that there exists an isomorphism between these two sub-problem classes, and the proof is complete. □

For each iteration $t \in \mathbb{N}$, assume that (x_t, y_t) is the search point queried by Algorithm H on payoff function g, and (x^*, y^*) is the unique NE in a two-player zero-sum game defined by $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Theorem 4.3.1 considers the following query complexity: assume the cost C_t is the unique queries made by Algorithm H,

$$T_{\text{LB}}(H, g) := \inf\{C_t > 0 \mid x_t = x^* \text{ or } y_t = y^*\}.$$

Now, we prove our main theorem. Briefly, we use a proof by induction on the size of the search space $\mathcal{X} \times \mathcal{Y}$. During the induction hypothesis, problem class is reduced from $\mathcal{F}$ to $\mathcal{F}((x_1, y_1), (b_1, b_2))$, decreasing the search space from size $N \times N$ to size $(N - 1) \times$

$(N-1)$, where $|\mathcal{X}| := N$. Then using Lemma 4.3.2 and the inductive step, it follows that $r(H, \mathcal{F}) = r(H', \mathcal{F})$ for any two deterministic algorithms H, H', and the claim for randomised algorithms quickly follows from the fact that any randomised algorithm can be viewed as a probability distribution among all deterministic algorithms.

Theorem 4.3.1. *Let $\mathcal{F}$ be a subset of all the payoff functions $g : \mathcal{X} \times \mathcal{Y} \to O$ with a unique Nash Equilibrium where $O \subseteq \mathbb{R}$ and $|\mathcal{X}| = |\mathcal{Y}|$. Let H be an arbitrary (randomised or deterministic) black-box adversarial optimisation algorithm for any $g \in \mathcal{F}$ where $\mathcal{F}$ is closed under permutations. Let $r(H, \mathcal{F})$ be the average (under the uniform distribution on $\mathcal{F}$) of the expected query complexity of H on $g \in \mathcal{F}$ (i.e., $\mathrm{E}_{g\sim\mathcal{F}}(T_{LB}(H, g))$). Then, for all BBO algorithms H, H', $r(H, \mathcal{F}) = r(H', \mathcal{F})$.*

Proof. Recall that the query complexity of H on g is $T_{\mathrm{LB}}(H, g) := \inf\{t > 0 \mid x_t = x^* \text{ or } y_t = y^*\}$ where (x_t, y_t) is the search point queried by Algorithm H on payoff function g at iteration t and (x^*, y^*) is the unique Nash Equilibrium in a two-player zero-sum game defined by $g : \mathcal{X} \times \mathcal{Y} \to O$. We denote the set of all well-defined functions from a set $\mathcal{X}$ to a set O by $\mathcal{H}(\mathcal{X}, O)$ and the set of all well-defined functions from a set $\mathcal{Y}$ to a set O by $\mathcal{H}(\mathcal{Y}, O)$. Let $\mathcal{F}$ be a class of games with payoff functions $\{g : \mathcal{X} \times \mathcal{Y} \to O$ with a unique NE$\}$, and assume that $\mathcal{F}$ is closed under permutation. For all $(x_0, y_0) \in \mathcal{X} \times \mathcal{Y}$, define

$$B^{(1)}_{x_0} := \{b_1 \in \mathcal{H}(\mathcal{Y}, O) \mid \text{there exists } g \in \mathcal{F} \text{ s.t. } b_1(y) = g(x_0, y) \text{ for all } y \in \mathcal{Y}\};$$
$$B^{(2)}_{y_0} := \{b_2 \in \mathcal{H}(\mathcal{X}, O) \mid \text{there exists } g \in \mathcal{F} \text{ s.t. } b_2(x) = g(x, y_0) \text{ for all } x \in \mathcal{X}\}.$$

Let $(u, v) \in \mathcal{X} \times \mathcal{Y}$ be the first query point that Algorithm 1 makes, we consider $b_1 \in B^{(1)}_u$ and $b_2 \in B^{(2)}_v$.

We first prove the claim holds for deterministic heuristics by induction. This is done by induction on the size of search space $N = |\mathcal{X}|$. Suppose for any two deterministic algorithms A, B on $\mathcal{F}$, if $N = 1$, then it means that $\mathcal{X} \times \mathcal{Y}$ only consists of one unique NE and $\mathcal{F}$ consists of one payoff function g. So for any two deterministic algorithms A, B, $r(A, \mathcal{F}) = r(B, \mathcal{F}) = 1$.

Next, assume $N \geq 2$ and $r(A, \mathcal{G}) = r(B, \mathcal{G})$ where $\mathcal{G}$ is any set of payoff functions with unique NE and search space in payoff functions of size $N-1$ and $\mathcal{G}$ is closed under permutation. Moreover, we assume $\mathcal{F}$ now consists of payoff functions with unique NE and search space in payoff functions of size N. We expand out the average of $r(A, \mathcal{F})$.

$$r(A, \mathcal{F}) = \sum_{g \in \mathcal{F}} \Pr(g \text{ is selected from } \mathcal{F}) \cdot T(A, g)$$

If the algorithm is lucky, then the first point $(u, v) \in \mathcal{X} \times \mathcal{Y}$ we query is the optimum (opt.) and (x^*, y^*) is the NE of the selected g.

$$\begin{aligned} &= 1 \cdot \Pr\left((u, v) \text{ is the opt. s.t. } u = x^* \text{ or } v = y^* \right) \\ &+ \Pr\left((u, v) \text{ is not the opt. s.t. } u = x^* \text{ or } v = y^* \right) \\ &\quad \times \left(1 + r\left(A, \mathcal{F}\left(u, v\right), \left(b_1, b_2\right)\right)\right)\end{aligned}$$

Notice that after we query (u, v), we can reduce the whole problem class $\mathcal{F}$ to $\mathcal{F}((u, v), (b_1, b_2))$ with case $N-1$ where b_1, b_2 are defined above.

Lemma 4.3.1 shows that if $\mathcal{F}$ is closed under permutation, then $\mathcal{F}((u, v), (b_1, b_2))$ is closed under permutation. Corollary 4.3.1 shows that if we restrict $g \in \mathcal{F}((u, v), (b_1, b_2))$, then all the restriction of g on $(\mathcal{X} \setminus \{u\}) \times (\mathcal{Y} \setminus \{v\})$, denoted by $g|_{(\mathcal{X} \setminus \{u\}) \times (\mathcal{Y} \setminus \{v\})}$, exhibits the same unique Nash Equilibrium (x^*, y^*) as g. So $\mathcal{F}((u, v), (b_1, b_2))$ is a set of payoff functions with unique NE (x^*, y^*) and the search space of size $N-1$ and closed under permutation for all (u, v) in which (u, v) is not the opt. s.t. $u = x^*$ or $v = y^*$.

With the help of Corollary 4.3.1 and Lemma 4.3.1, we apply the induction hypothesis to this sub-problem class $\mathcal{F}((u, v), (b_1, b_2))$ [2]. Also note that we consider the uniform distribution on problem class $\mathcal{F}$ and thus for any $(u', v') \neq (u, v)$, we have

$$\Pr\left((u, v) \text{ is the opt. s.t. } u = x^* \text{ or } v = y^* \right) = \Pr\left((u', v') \text{ is the opt. s.t. } u = x^* \text{ or } v = y^* \right).$$

[2] We defer further explanations of the role of Corollary 4.3.1 and Lemma 4.3.1 to the supplementary material Section D.

The induction hypothesis on the case $N-1$ with Lemma 4.3.2 shows that two sub-problem classes are essentially isomorphic, and thus have the same average query complexity from the induction hypothesis step.

$$r\left(A, \mathcal{F}((u,v),(b_1,b_2))\right) = r\left(B, \mathcal{F}((u',v'),(b_1,b_2))\right).$$

Thus, we can conclude that $r(A,\mathcal{F}) = r(B,\mathcal{F})$ for any deterministic algorithms A, B.

Now, we generalise the result to randomised algorithms. Let $m \in \mathbb{N}$ be a finite number to denote the number of different deterministic search heuristics. We consider a randomised search strategy to be a probability distribution $p = (p_1, \cdots, p_m)$ and choose the i-th deterministic search strategy with probability p_i. It is well-known (see details in Motwani and Raghavan, 1995) that the expected cost of a randomised search heuristic is the weighted average of the cost of the deterministic search heuristics. Since all deterministic search heuristics have the same cost, this also holds for all randomised search strategies. □

Theorem 4.3.1 reveals an important underlying result: All black-box adversarial algorithms exhibit the same average runtime $r(H,\mathcal{F})$ of all possible problem instances (or problem instances c.u.p.) with a unique Nash Equilibrium in a two-player zero-sum game setting. It is a reasonable result since Theorem 4.3.1 tells us that if a class of payoff functions with a unique Nash Equilibrium does not change by any permutation on the input space, there is no structure provided for any search heuristic or any optimisation algorithm to use and it cannot help to find the Nash Equilibrium. This is also the reason why the original No Free Lunch theorem for traditional optimisation holds Wolpert, 2002; Wolpert and Macready, 1997. As concluded by Ho and Pepyne (2002), “if anything is possible and occurs with the same probability, then nothing can be expected".

This result is also surprising since Wolpert and Macready (2005) and Service and Tauritz (2008) both show there exists a free lunch with respect to the two solution concepts in Definition A.5.1 and Definition A.5.2. However, our result does not contradict the pre-

vious result. We consider the performance measure $T(H, g)$ and a different query model considered by the previous work. In particular, Definition A.5.1 and Definition A.5.2 only take the player x into account, while the opponent optimum and different query model and performance measures can make a difference, resulting in either FL or NFL results. In summary, our new NFL theorem highlights the significance of solution concepts and also reveals that adversarial optimisation can exhibit "no-free-lunch", in particular for the unique NE solution concept.

4.4 Black-Box Complexity of Adversarial Optimisation

As shown in the previous NFL theorem, no better universal algorithms exist on structure-free problems (i.e., under the assumption that the payoff function has a unique NE). In order for an algorithm to guarantee good performance, it is necessary to restrict the algorithm to classes of games that possess some structure. To compute the Nash equilibrium of certain classes of problems, including Nash equilibrium in a black-box setting, there are many results on analysing the computational complexity for black-box algorithms, and researchers aim to minimise the query complexity and provide more efficient algorithms to compute Nash equilibrium in two-player zero-sum game settings (Benford and Lehre, 2024b; Hevia Fajardo et al., 2023; Lehre, 2022; Maiti et al., 2023) The following questions remain under-explored: How does the performance of an algorithm (like query complexity) depend on the size of the search space, and for some classes of problems, does there exist non-trivial lower bounds on the runtime that hold for all black-box adversarial algorithms? We answer these questions here using Black-Box Complexity (BBC).

4.4.1 The Unrestricted Black-Box Model and the BBC

This section focuses on adversarial black-box optimisation and studies the query complexity of learning Nash equilibrium in a two-player zero-sum game setting. We refer to (Doerr et al., 2013; Doerr and Neumann, 2020; Droste et al., 2006) for a more detailed introduction to the black-box complexity theory for traditional black-box optimisation.

To prove a lower bound that holds for all algorithms, it is necessary first formally to define what constitutes an algorithm. To do so, we construct an unrestricted black-box model of adversarial optimisation.

Algorithm 9 An Unrestricted Black-Box Model with Unique Query History

Require: Search spaces $\mathcal{X}, \mathcal{Y}$.

Require: Payoff functions $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Require: Initial distribution $P_\emptyset$.

1: Initialise (x_0, y_0) based on $P_\emptyset$; Initialise $H_0 = \emptyset$ and $C_0 = 0$.

2: **for** $t = 1, 2, \cdots$ until the termination criterion met **do**

3: Choose some probability distribution $P_{I(t)}$, depending only on $I(t)$ where $I(t) := \Pi_{j=1}^{t-1}(x_j, y_j, g(x_j, y_j), h(x_j, y_j)) \in (\mathcal{X} \times \mathcal{Y} \times \mathbb{R} \times \mathbb{R})^{t-1}$

4: Produce a random search point (x_t, y_t) based on $P_{I(t)}$.

5: Query the payoffs $g(x_t, y_t), h(x_t, y_t)$

6: **if** $(x_t, y_t) \notin H_{t-1}$ **then** $C_t = C_{t-1} + 1$; $H_t = H_{t-1} \cup \{(x_t, y_t)\}$.

7: **else** $C_t = C_{t-1}$; $H_t = H_{t-1}$.

Algorithm 9 defines a class of algorithms subject to various probability distributions and samples the new strategy pair based on previous pairs and their payoffs. The initial search point (x_0, y_0) is independent of the problem, so we can choose any probability distribution $P_\emptyset$ to initialise the algorithm. Subsequent strategy pairs are obtained by sampling subject to a probability distribution $P_{I(t)}$. By specifying different sample probability distribution

$P_{I(t)}$, Algorithm 9 represents various black-box optimisation algorithms, including several adversarial search (also called competitive coevolutionary) algorithms (Lehre et al., 2023; Wolpert and Macready, 2005) and randomised algorithms FindPsne (designed to learn the NE in bimatrix games) (Maiti et al., 2023). Note that the model only considers the cost of unique queries made by the algorithm (i.e. we check the search point in the previous query history H_t in line 6). We assume that an algorithm which makes the same query twice is only charged for the cost of one of the queries. This is consistent with previous research (Droste et al., 2002b; Wolpert and Macready, 2005; Wolpert and Macready, 1997). Also, note that if we do not make this assumption, then clearly some algorithms can take infinite time. Additionally, a more general extension could be to only query either g or h in line 5, since for some non-zero-sum games, the algorithm could evaluate g more thatn h

Now, we define the query complexity (or runtime) of black-box adversarial optimisation algorithms by extending the idea of the traditional single-objective black-box optimisation algorithm in (Droste et al., 2006; Winzen, 2011) and assuming $h = -g$ (i.e. zero-sum game) in this case.

Definition 4.4.1 (Adapted from (Babichenko, 2020))**.** *Given any unrestricted black-box algorithm with unique query history A and the payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $T(A, g)$ is the query complexity of A with respect to g and (x_t, y_t) is the search point generated by A if*

$$T(A, g) := \inf\{dC_t \in \mathbb{N} \mid (x_t, y_t) \in \arg\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)\},$$

where $d \in \{1, 2\}$. If the game is zero-sum, then $d = 1$; otherwise, set $d = 2$.

Note that $T(A, g) \in \mathbb{N} \cup \{\infty\}$ is the number of payoff evaluations until A queries for the first time some $(x^*, y^*) \in \arg\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)$. Now, we can define what black-box complexity means with respect to a given class of adversarial optimisation algorithms and problem classes.

Definition 4.4.2. *For a class $\mathcal{G}$ of payoff functions $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{N}$, the A-black-box complexity of $\mathcal{G}$ is defined as $T(A, \mathcal{G}) := \sup_{g \in \mathcal{G}} T(A, g)$, the runtime of A under the worst-case scenario on $\mathcal{G}$. Then, the $\mathcal{A}$-black-box complexity of $\mathcal{G}$ is $T(\mathcal{A}, \mathcal{G}) := \inf_{A \in \mathcal{A}} T(A, \mathcal{G}) = \inf_{A \in \mathcal{A}} \sup_{g \in \mathcal{G}} T(A, g)$; the best or minimum complexity among all $A \in \mathcal{A}$ with respect to $\mathcal{G}$. If $\mathcal{A}$ is the whole class of all black-box algorithms, we denote $T(\mathcal{G})$ the unrestricted black-box complexity of $\mathcal{G}$.*

We want to point out the difference between these two definitions. Definition 4.4.1 considers the query complexity for a particular algorithm and problem instance, and Definition 4.4.2 considers the black-box complexity for the best possible query complexity of all possible given algorithms under the worst-case scenario. The traditional black-box complexity theory characterises the difficulty of a certain class of problems and explores the limitations of the given black-box optimisation algorithms. We expect our extension to adversarial optimisation can provide similar insights as well.

4.4.2 A General Lower Bound for Black-Box Adversarial Optimisation

We first provide a lower bound of black-box complexity for a general class of black-box adversarial optimisation problems.

Theorem 4.4.1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be any finite sets. Assume that $B \subset \mathbb{R}$ with $k = |B| \geq 2$. Consider any class of two-player zero-sum games $\mathcal{G} \subseteq \{g : \mathcal{X} \times \mathcal{Y} \to B\}$ such that for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, there exists a game $g_{x,y} \in \mathcal{G}$ which has (x, y) as unique, pure Nash Equilibrium. Then, the class $\mathcal{G}$ has black box complexity at least $\lceil \log_k |\mathcal{X} \times \mathcal{Y}| \rceil - 1$.*

Proof. The proof, which uses Yao's minimax principle (Theorem A.5.1), is analogous to the proof of Theorem 2 in (Droste et al., 2006). We need to construct a suitable probability

distribution p over the set of games. By assumption, for each $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we can associate one game $g_{x,y}$ which has (x, y) as the unique pure NE. We let p be the uniform distribution over the set of the games $\{g_{x,y} \mid (x, y) \in \mathcal{X} \times \mathcal{Y}\} \subseteq \mathcal{G}$.

We now consider the runtime of any deterministic black-box algorithm $A \in \mathcal{A}_{\text{det}}$ with respect to a random game g_{x^*,y^*} which is sampled according to distribution p. The algorithm is a decision tree, where each node is a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ corresponding to a query made by algorithm A, and each edge corresponds to one of at most k possible outcomes $g(x, y)$ of this query. The runtime of algorithm A on the random game g_{x^*,y^*} corresponds to the depth of (x^*, y^*) in this decision tree. The expected depth is therefore lower bounded by $\lceil \log_k(|\mathcal{X} \times \mathcal{Y}|) \rceil - 1$.

Hence, we have for all deterministic algorithms $A \in \mathcal{A}_{\text{det}}$,

$$\mathrm{E}(T(I_p, A)) \geq \log_k(|\mathcal{X} \times \mathcal{Y}|) - 1.$$

This implies that

$$\min_{A \in \mathcal{A}_{\text{det}}} \mathrm{E}(T(I_p, A)) \geq \log_k(|\mathcal{X} \times \mathcal{Y}|) - 1.$$

By Yao's Principle, for any distribution q over the set of deterministic algorithms $\mathcal{A}_{\text{det}}$,

$$\max_{I \in \mathcal{I}} \mathrm{E}(T(I, A_q)) \geq \min_{A \in \mathcal{A}_{\text{det}}} \mathrm{E}(T(I_p, A)) \geq \log_k(|\mathcal{X} \times \mathcal{Y}|) - 1.$$

The proof follows by noting that any randomised algorithm can be described as sampling a deterministic algorithm according some distribution q, and applying this algorithm. □

4.4.3 A General Lower Bound for Two-player Zero-Sum Bimatrix Games

In this subsection, we employ Yao's Principle and the No Free Lunch Theorem to provide a general lower bound for the black-box complexity of searching Nash Equilibrium in zero-sum

bimatrix games, and this leads to a sharper lower bound compared to Theorem 4.4.2 (Maiti et al., 2023).

Theorem 4.4.2 (A Lower Bound for Finding the Nash Equilibrium (Maiti et al., 2023))**.** *Let $\mathcal{A}$ be any randomised algorithm that correctly computes the Nash equilibrium of the input matrix. Let $\tau(M)$ denote the number of entries queried from the matrix M by $\mathcal{A}$. Then, there exists a payoff matrix $A \in \mathbb{R}^{n\times n}$ with a unique Nash equilibrium $(x_\star, y_\star)$ such that $\mathbb{E}[\tau(A)] \geq \frac{nk}{8}$ where $k = |\text{supp}(y_\star)|$ and the randomness in $\tau(A)$ is due to $\mathcal{A}$ only.*

As we can see, if we consider pure Nash Equilibrium (i.e., set $k = 1$), then we obtain a general lower bound for finding pure Nash Equilibrium of at least $n/8$. Next, we present a better lower bound.

Theorem 4.4.3. *Let $\mathcal{A}$ be the set of all randomised algorithms defined by Algorithm 9 and $T(\mathcal{A}, P)$ denote the query complexity of $\mathcal{A}$ with respect to the input payoff matrix P for a two-player zero-sum game[3]. Then, there exists a payoff matrix $P \in \mathbb{R}^{n\times n}$ with a unique pure Nash equilibrium (x^*, y^*) such that $\text{E}(T(\mathcal{A}, P)) \geq (n+1)/2$. Thus, the black-box complexity with respect to $\mathcal{A}$ of the problem class consisting of all bimatrix games with a unique Nash Equilibrium is at least $(n+1)/2$.*

Proof. Given payoff matrix P, recall that for any $A \in \mathcal{A}$,

$$
\begin{aligned}
T_{\text{LB}}(A, P) &:= \inf\{C_t > 0 \mid x_t = x^* \text{ or } y_t = y^*\};\\
T(A, P) &:= \inf\{C_t > 0 \mid x_t = x^* \text{ and } y_t = y^*\};\\
\mathcal{M} &:= \{P \in \mathbb{R}^{n\times n} \text{ with a unique PSNE}\}.
\end{aligned}
$$

Clearly, $T(A, P) \geq T_{\text{LB}}(A, P)$. We denote $\mathcal{M}$ as the set of input instances. Now, we estimate the lower bound by using Yao's minimax principle (we denote the query complexity

[3]Note that $T(A, P) = T(A, g)$ where $g(x, y) := e_y^T P e_x$ with e_x, e_y denote the elementary probability distribution over probability simplex $\Delta_{\{0,1\}^n}$

of a specified algorithm namely ALG searching the PSNE of P by $T(\text{ALG}, P)$ and the set of all deterministic black-box adversarial optimisation algorithms by $\mathcal{A}_{\text{det}}$): for any randomised algorithm $A \in \mathcal{A}$,

$$\max_{P \in \mathcal{M}} \mathrm{E}(T(A, P)) \geq \min_{\text{ALG} \in \mathcal{A}_{\text{det}}} \mathop{\mathrm{E}}_{P \sim \text{Unif}(\mathcal{M})} (T(\text{ALG}, P))$$

Using the definition of $T(\text{ALG}, P)$ and $T_{\text{LB}}(\text{ALG}, P)$ gives

$$\geq \min_{\text{ALG} \in \mathcal{A}_{\text{det}}} \mathop{\mathrm{E}}_{P \sim \text{Unif}(\mathcal{M})} (T_{\text{LB}}(\text{ALG}, P))$$

Using Theorem 4.3.1, we know the expected performance of all algorithms on any all problem instances in $\mathcal{M}$ is the same with respect to the unique Nash Equilibrium solution concept. So, we define a new algorithm ALG* such that it is a deterministic algorithm starting from $i = j = 1$, and it makes one query in each iteration. It continues to query $P_{1,1}, P_{2,2}, \ldots P_{n,n}$ (i.e. query the entries in the diagonal of the payoff matrix P). ALG* continues the processes until it reaches either the column or the row of the position of the unique Nash Equilibrium. So, we derive

$$= \mathop{\mathrm{E}}_{P \sim \text{Unif}(\mathcal{M})} (T_{\text{LB}}(\text{ALG}^*, P))$$

Note that we consider the uniform distribution on $\mathcal{M}$. It means that the probability that (x^*, y^*) lies in the j-th column of payoff matrix P is $1/n$ for all $j \in [n]$. Then, the total expected query complexity is $\sum_{j=1}^{n} j\frac{1}{n} = \frac{n+1}{2}$. So, we derive

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

This completes the proof. □

Theorem 4.4.3 provides a sharper lower bound by a multiplicative factor 4 compared with the current best bound by Maiti et al., 2023. This result also demonstrates that in a two-player zero-sum bimatrix game (with an $n \times n$ payoff matrix), there are complex instances where no randomised algorithm can achieve better than $O(n)$ query complexity unless additional problem structure is provided.

4.4.4 Applications on Two-Player Zero-Sum Games

4.4.4.1 Introduction to Binary Voting Games

This section provides some example applications of our black-box complexity results. Voting games are popular games studied in game theory, computational social choice theory (Brandt et al., 2012; Brandt et al., 2016; Grandi et al., 2015) and Boolean games (Harrenstein et al., 2001). Voting is considered a fundamental tool for analysing multi-agent systems (Aziz et al., 2019; Grandi et al., 2015). We start with simple binary voting games in which the outcome or payoff is 0 or 1 (or -1 and 1). These games also play a role in the analysis of Boolean functions (O'Donnell, 2008).

Convergence to NE in plurality voting – a type of voting game where the winner is determined by the majority – has been studied from the perspective of social choice theory and researchers specify certain conditions to guarantee the voting games to converge to NEs (Meir et al., 2010). Some natural question arises: are there any randomised algorithms that can find these NEs in voting games efficiently? What are the limitations of these black-box optimisation algorithms? Using black-box complexity analysis, we can answer the questions about the efficiency/inefficiency of black-box optimisation algorithms on binary voting games.

We formulate binary voting games in the context of adversarial optimisation as follows. Consider two parties represented by vectors $x, y \in \{0,1\}^n$ where $n \in \mathbb{N}$. Each group has n members that either "in favour" (encoded by 1) or "against" (encoded by 0) a particular proposal or decision. One group seeks a strategy x^* that maximises its minimum gains against any strategy of the other group, while the other group seeks a strategy y^* such that its choice minimises the maximum gains of the first group. It essentially forms a two-player zero-sum game.

Definition 4.4.3. *For $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, the payoff function* DIAGONAL $: \mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$

is

$$\textsc{Diagonal}(x,y) := \begin{cases} 1 & \text{if } |y|_1 \leq |x|_1 \\ -1 & \text{otherwise.} \end{cases}$$

In Definition 4.4.3, we present the votes of both groups by binary bitstrings and the payoff g can be viewed as a binary voting game where the payoff only depends on which group has the stronger majority "influence". If one group has a stronger 'in favour' "influence" in the sense of the number of the support votes (i.e. the number of 1 in the encoding binary bitstring), then we get a payoff 1 and -1 otherwise. We are interested in computing NE in these two-player zero-sum games, i.e. solving $(x^*, y^*) \in \arg\max_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} g(x,y)$. Notice that in Diagonal, $(x_n, y_n) = (1^n, 1^n)$ is one of the NE optima. In this optimum, neither of the two groups is willing to deviate from affecting their payoff $g(x_n, y_n)$ anymore. This exactly coincides with the definition of NE.

Next, we consider a different binary voting game, denoted by Plateau. To make the binary voting game more challenging, we introduce some plateaus in games.

Definition 4.4.4. *For $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, a constant $\delta \in (0,1)$, the payoff function $\textsc{Plateau} : \mathcal{X} \times \mathcal{Y} \to \{-1,1\}$ is defined as*

$$\textsc{Plateau}(x,y) := \begin{cases} f(y) & \text{if } ||x|_1 - \frac{n}{2}| < \frac{\delta n}{2} \\ g(x,y) & \text{otherwise} \end{cases}$$

where $f : \mathcal{Y} \to \{-1,1\}$ and $g : \mathcal{X} \times \mathcal{Y} \to \{-1,1\}$ are any functions such that the NE of Plateau is $(x^*, y^*) \notin \{(x,y) \mid ||x|_1 - \frac{n}{2}| < \frac{\delta n}{2}\}$.

Definition 4.4.4 introduces a plateau when comparing the "influence" between two groups and defines a general class of pseudo-Boolean benchmarks with a plateau. Imagine a committee deciding on a new policy where there are two groups with equal voting power. If the votes from group $\mathcal{X}$ are balanced or nearly balanced (within the plateau), then the

votes from the second group ($\mathcal{Y}$) come into play. It is like their votes are the tiebreaker. If the second group votes in favour, then the policy passes; if they vote against it, then it fails. If the votes from group $\mathcal{X}$ are outside the plateau, then the payoff is not restricted to be determined by $y \in \mathcal{Y}$.

Finally, we define game instances generated by (u, v) where $u, v \in \{0,1\}^n$.

Definition 4.4.5 ((u,v)-game instance)**.** *For all $u, v \in \{0,1\}^n$, given $f : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$, we define the (u,v)-instance of f, denoted by $f_{(u,v)}$, as $f_{(u,v)}(x,y) := f(u \oplus x, v \oplus v)$.*

We can see that for any $u, v \in \{0,1\}^n$, $f_{(u,v)}$ generates the same payoff landscape as f. In this dissertation, f will be either DIAGONAL or PLATEAU. We defer more details to Section A.5.6[4].

4.4.4.2 Black-Box Complexity of Learning Nash Equilibrium in Binary Voting Games

First, let us illustrate how Theorem 4.4.1 and Theorem 4.4.3 work on these simple examples. If we consider a general class of black-box optimisation algorithms defined by Algorithm 9 on binary voting games with a unique Nash Equilibrium, then we provide a general lower bound of black-box complexity as follows.

Theorem 4.4.4. *The black-box complexity with respect to Algorithm 9 of the binary voting games with problem size $n \in \mathbb{N}$ and a unique Nash Equilibrium is $e^{\Omega(n)}$.*

Proof. Given an arbitrary binary voting game with problem size n defined by a payoff function $g : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$, we consider the corresponding payoff matrix P. For each party, there are 2^n possible strategies encoded by a binary bitstring of length n. So $P \in$

[4]We defer the definition of xor $\oplus$ to Section A.5.6 in the supplementary material.

$\mathbb{R}^{2^n \times 2^n}$. Using Theorem 4.4.3 with arbitrary algorithms in the class defined by Algorithm 9 and the payoff matrix P gives us the black-box complexity is at least $\frac{2^n+1}{2} = e^{\Omega(n)}$. □

Theorem 4.4.4 means that there exist no universally good black-box optimisation algorithms defined by Algorithm 9 that can solve all binary voting games with unique Nash Equilibrium efficiently, (i.e. with polynomial query complexity of the problem size). To yield a better performance of black-box optimisation algorithms on binary voting games, we need to specify the problem class we work on. Next, we consider DIAGONAL and explore the black-box complexity with respect to the class of black-box optimisation algorithms $\mathcal{A}$ defined by Algorithm 9 of DIAGONAL. This reveals that DIAGONAL is a feasible benchmark problem for black-box adversarial optimisation algorithms.

Theorem 4.4.5.

$$\textsc{Diagonal}_n := \{\textsc{Diagonal}_{(u,v)} \mid (u,v) \in \{0,1\}^n \times \{0,1\}^n\},$$

the black-box complexity with respect to Algorithm 9 of $\textsc{Diagonal}_n$ *is* $\Theta(n)$.

Proof. We consider an algorithm $A \in \mathcal{A}$, which operates in two phases. In Phase 1, we use a randomised search heuristic RSH (as summarised in Algortithm 10) that performs random local search as follows. In each iteration of Phase 1, the algorithm samples x and y uniformly at random, then flips one bit uniformly at random in x to obtain x'. It then compares the values of $\textsc{Diagonal}_{u,v}(x,y)$ and $\textsc{Diagonal}_{u,v}(x',y)$. Phase 1 ends when $\textsc{Diagonal}_{u,v}(x,y) \neq \textsc{Diagonal}_{u,v}(x',y)$. Once Phase 1 ends, the algorithm knows that $|u \oplus x|_1 = |v \oplus y|_1$. Note that we do not only use random sampling until the initial search points satisfying $|u \oplus x|_1 = |v \oplus y|_1$ since in this black-box oracle setting, we cannot see bitstrings $u \oplus x, v \oplus y$. Thus, there is no clear stopping criterion for solely random initialisation.

We now consider how long it takes for RSH in Phase 1 to finish (i.e. the search point arrives at the diagonal). We first estimate $\Pr(Z_1 = Z_2)$. Notice that let $Y := n - Z_2$ and

Algorithm 10 RSH: Randomised Search Heuristic (Phase 1)

Require: Problem size n; payoff function $\textsc{Diagonal}_{u,v} : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}$.

repeat
  Sample $x \sim \text{Unif}(\{0,1\}^n)$ and $y \sim \text{Unif}(\{0,1\}^n)$ independently.
  Sample a bit position $i \in \{1,\dots,n\}$ uniformly at random.
  Obtain y' from y by flipping its i-th bit.
  $d \leftarrow \textsc{Diagonal}_{u,v}(x,y)$; $\quad d' \leftarrow \textsc{Diagonal}_{u,v}(x,y')$.
until $d \neq d'$
if $\textsc{Diagonal}_{u,v}(x,y) = 1$ **then**
  $(x,y) = (x,y)$.
else
  $(x,y) = (x,y')$.
return (x,y).

Y is also subject to $\text{Bin}(n, 1-1/2) = \text{Bin}(n, 1/2)$. So the event $\{Z_1 = Z_2\}$ is equivalent to $\{Z_1 = Y\} = \{Z_1 + Z_2 = n\}$. Thus, we can derive the following estimate by using Stirling's approximation (i.e. $n! \sim \sqrt{2\pi n}(\frac{n}{e})^n$),

$$\Pr(Z_1 = Z_2) = \binom{2n}{n} 2^{-2n} \sim \frac{1}{\sqrt{\pi n}} 2^{2n} \cdot 2^{-2n} = \frac{1}{\sqrt{\pi n}}.$$

So the expected runtime for finishing Phase 1 is $O(\sqrt{n})$.

In Phase 2, Algorithm A employs a deterministic search heuristic DSH (as summarised in Algortithm 11) that alternately updates y and x by scanning their bit positions in a fixed order. The heuristic maintains two counters j and i, each remembering the bit position reached in the previous iteration. To update y, DSH repeatedly flips the bit at position j (incrementing j when unsuccessful) until it finds a flip yielding payoff -1, which is then accepted. It then updates x analogously, flipping the bit at position i until it obtains payoff 1. The accepted flips remain in place, and the counters carry forward to the next iteration, ensuring that the search resumes from the last visited bit position. This

alternating procedure continues until the termination criterion is met.

Algorithm 11 DSH: Deterministic Search Heuristic (Phase 2)

Require: Current search point $(x, y) \in \{0,1\}^n \times \{0,1\}^n$; payoff function $\textsc{Diagonal}_{u,v} : \{0,1\}^n \times \{0,1\}^n \to \{-1, 1\}$; termination criterion.

$i \leftarrow 1$; $j \leftarrow 1$ ▷ counters for bit positions in x and y
while termination criterion not met **do**
  $found_y \leftarrow$ **false**
  while not $found_y$ **do**
    Flip the j-th bit of y to obtain y'.
    if $\textsc{Diagonal}_{u,v}(x, y') = -1$ **then**
      $y \leftarrow y'$ ▷ accept new y
      $found_y \leftarrow$ **true**
    else
      $j \leftarrow j + 1$ ▷ move to next bit
  $found_x \leftarrow$ **false**
  while not $found_x$ **do**
    Flip the i-th bit of x to obtain x'.
    if $\textsc{Diagonal}_{u,v}(x', y) = 1$ **then**
      $x \leftarrow x'$ ▷ accept new x
      $found_x \leftarrow$ **true**
    else
      $i \leftarrow i + 1$ ▷ move to next bit
return (x, y)

Now, we consider how long it takes for DSH in Phase 2 to reach the optimum. Notice that for each bit, we need one query of the payoff function to determine whether it is the correct bit to flip. We need to flip the correct bits for both x and y. Since the

maximum Hamming distance to the optimum of DIAGONAL is bounded by $2n$, then the overall runtime is bounded above by $2n = O(n)$. Adding the expected runtime for both Phases gives $O(n) + O(\sqrt{n}) = O(n)$. Together with Theorem 4.4.1, we can conclude that the black-box complexity of DIAGONAL_n is $\Theta(n)$. □

Theorem 4.4.5 implies that DIAGONAL_n is a sensible maximin benchmark for testing black-box optimisation algorithms. It means that if we restrict the problem class to a certain class with a specific structure, then it is possible to solve them in polynomial query complexity. We have seen the black-box complexity results on DIAGONAL_n. Next, we start to consider more challenging binary voting games, PLATEAU. We are interested in whether there is any efficient black-box adversarial optimisation that can solve PLATEAU_n in polynomial query complexity (i.e. $O(n)$). To answer this question, we need to compute its black-box complexity.

Theorem 4.4.6. *Given the game class,*

$$\textit{PLATEAU}_n := \{\textit{PLATEAU}_{(u,v)} \mid (u, v) \in \{0,1\}^n \times \{0,1\}^n\},$$

the black-box complexity with respect to the class of algorithms defined by Algorithm 9 of $\textit{PLATEAU}_n$ *is* $e^{\Omega(n)}$.

Proof. Recall the definition of $\text{PLATEAU}_{u,v}$ for arbitrary u, v. We call the set $\{(x, y) \mid ||u \oplus x|_1 - \frac{n}{2}| \leq \frac{\varepsilon n}{2}\}$ x-independent, and we call any query to this set x-independent. Note that for any x-independent query has payoff $\text{PLATEAU}_{u,v}(x, y) = f(v \oplus y)$, i.e., it is independent of x. Furthermore, note that the Nash Equilibrium is not x-independent.

We will apply Yao's Principle with respect to the distribution p over instances where u is sampled uniformly at random among all bitstrings of length n, and $v = 0^n$. We consider the average case runtime of deterministic algorithms with respect to distribution p. Such algorithms can be modelled as binary decision trees: in each node, the algorithm makes a

query (x, y), and for each outgoing edge, the algorithm obtains one of two possible payoff values $\textsc{Plateau}_{u,v}(x, y) \in \{-1, 1\}$. We call the x-independent path in the decision tree the longest path $(x_1, y_1), \ldots, (x_t, y_t)$ such that (x_1, y_1) is the root node and all nodes (x_i, y_i) for $i \in [t]$ are x-independent. We let t denote the length of the x-independent path. Since the outcome of an x-independent query only depends on $v \oplus y$ which is deterministic, the x-independent path of a decision tree is unique. Furthermore, the length of the x-independent path is a lower bound on the runtime of the algorithm since the Nash Equilibrium is not x-independent.

We now bound the length of the x-independent path from below. Let (x_i, y_i) be any fixed query in the decision tree, and F_i the event that (x_i, y_i) is not x-independent. We sample u by picking each bit independently and uniformly at random in $\{0, 1\}$. This implies that the number of 1 bits for each bit is subject to a Bernoulli random variable $\text{Bin}(1/2)$. We therefore have $|u \oplus x_i|_1$ is binomially distributed $\text{Bin}(n, 1/2)$. Applying Chernoff's bound (Theorem A.2.3), we get

$$\Pr(F_i) = \Pr\left(||u \oplus x_i|_1 - \frac{n}{2}| > \varepsilon\frac{n}{2}\right) \leq 2e^{-\varepsilon^2 n/6}. \tag{4.1}$$

Let E_1 be the intersection of all the failure events, i.e., $E_1 = \cap_{i=1}^{t}\overline{F_i}$. Using De Morgan's laws gives $\cap_{i=1}^{t}\overline{F_i} = \overline{\cup_{i=1}^{t}F_i}$. Therefore, by a union bound, the x-independent path has length at least $t = 2e^{\varepsilon^2 n/12}$ with probability

$$\Pr(E_1) = 1 - \Pr\left(\cup_{i=1}^{t}F_i\right) \geq 1 - t2e^{-\varepsilon^2 n/6} = 1 - e^{-\Omega(n)}.$$

We then have by the law of total probability

$$\min_{A\in\mathcal{A}} \mathrm{E}(T(I_p, A)) \geq \min_{A\in\mathcal{A}} \Pr(E_1)\,\mathrm{E}(T(I_p, A) \mid E_1) \geq (1 - e^{-\Omega(n)})2e^{\varepsilon^2 n/12} = e^{\Omega(n)}.$$

The proof now follows by using Yao's minimax Principle (Theorem A.5.1). The expected worst-case runtime of any randomised algorithm is lower bounded by the average case

runtime of deterministic algorithms.

$$\max_{I \in \mathcal{I}} \mathrm{E}(T(I, A_q)) \geq \min_{A \in \mathcal{A}} \mathrm{E}(T(I_p, A)) = e^{\Omega(n)}.$$

□

Theorem 4.4.6 implies that all black-box adversarial optimisation algorithms defined by the unrestricted model (i.e., Algorithm 9) have exponential runtime on $\textsc{Plateau}_n$. They need at least an exponentially large query complexity with respect to the problem size n. It is evident that Plateau is too challenging that it may not be a proper benchmark for black-box adversarial optimisation algorithms.

4.4.4.3 Summary

Our study introduces the concept of black-box complexity in binary voting games, providing insights into the challenges faced by general black-box adversarial optimisation algorithms. These examples illustrate two kinds of problems within the general class of binary voting games with unique NE: the polynomial-solvable class (i.e., there exists an algorithm that can solve all problem instances of this class in polynomial runtime) and the non-polynomial-solvable class (i.e., there exists no algorithm that can solve all problem instances of this class in polynomial runtime).

Theorem 4.4.4 rigorously shows that no universal algorithm can efficiently learn the unique Nash Equilibrium (NE) in these games due to their structure-free nature, where efficiency means small query complexity (usually polynomial in the size of input). However, when assuming specific problem structures, such as $\textsc{Diagonal}_n$, it is possible to come up with a better algorithm which achieves better polynomial query complexity as shown in Theorem 4.4.5. $\textsc{Diagonal}_n$ can be a promising benchmark for evaluating black-box algorithms. Additional assumptions on the payoff function, like those in $\textsc{Plateau}_n$ problems, proved in Theorem 4.4.6, do not lower the difficulty of the problems. This emphasises the

need for a more careful selection of benchmarks. Black-box complexity emerges as a valuable tool for distinguishing between potentially easy and hard problem instances, guiding the design of black-box algorithms.

4.5 Conclusion

Utilising the tools from game theory and Yao's principle, we rigorously proved impossibility results for a universally effective adversarial algorithm applicable across various problem classes in black-box adversarial optimisation. We emphasise the impact of solution concept selection on the feasibility of a "free lunch" in adversarial optimisation. Additionally, we introduce the notion of black-box complexity in black-box adversarial optimisation and characterise the difficulty of learning the unique optimum in adversarial optimisation and solving two-player zero-sum games.

The results from this paper build up a foundation for future studies on the strengths and limitations of adversarial optimisation. More specifically, it highlights the need for more comprehensive benchmarks and careful selections of solution concepts when using any black-box adversarial optimisation algorithms. Moreover, no black-box optimisation algorithm for learning the Nash equilibrium in two-player zero-sum games can exceed the logarithmic complexity relative to search space size. Meanwhile, no algorithm can solve any bimatrix game with unique NE faster than the linear query complexity in terms of the size of input payoff matrices.

Although our work makes a first step towards the new NFL and BBC results on black-box adversarial optimisation, we want to point out some limitations of our current work and list them as our future work. Firstly, our theoretical results build on discrete exponential large search spaces rather than countably infinite (e.g. $\mathbb{N}$) or uncountable infinite (e.g. $\mathbb{R}$) sets. To generalise our results to infinite sets, we might require further assumptions on our

search spaces. Secondly, our NFL focuses on two-player zero-sum games with unique NE.

Future direction of our work includes extending Theorem 4.3.1 to other solution concepts like mixed strategy Nash Equilibrium or exploring different possible solution concepts which may exhibit free lunch or not. Additionally, it is interesting to generalise NFL and BBC analysis to zero-sum games with multiple NEs and more general search spaces. Finally, it is interesting to analyse other black-box models, such as unbiased black-box complexity models, to characterise the difficulty of adversarial problems that different classes of search heuristics can solve.

Chapter Five

EA vs CoEA on Sparse Binary Zero-Sum Games

This chapter is based on the following publication:

Overcoming Binary Adversarial Optimisation with Competitive Coevolution (Lehre and Lin, 2024c) which is published in Proceedings of the 18th International Conference on Parallel Problem Solving From Nature (PPSN'24).

5.1 Introduction

5.1.1 Background

Competitive CoEAs can be applied to problems that involve adversaries, such as Maximin optimisation problems. With the widespread use and development of GANs (Goodfellow et al., 2020), there are recent successful applications of competitive CoEAs for GANs (Al-Dujaili et al., 2018; Toutouh et al., 2019) and coevolutionary learning (Mitchell, 2006). Competitive CoEAs share similarities with neural network-based adversarial models but require less information, e.g., a gradient. However, despite their potential, the application of CoEAs is challenging. One of the main difficulties is that these algorithms often exhibit pathological behaviours, such as cyclic behaviours, disengagement, and over-specialisation (Popovici et al., 2012; Wiegand, 2004) as discussed in Section 2.2. These challenges limit the widespread use of CoEAs.

As discussed in Chapter 2, there is a growing interest in optimisation problems that involve one or more adversaries, and CoEAs have been suggested to be a promising approach (Popovici et al., 2012). However, there is still a gap in the theoretical understanding of CoEAs on test-based problems. In particular, Hillis (1990) showed empirically that there is a significant improvement in sorting networks via competitive CoEAs compared with normal EAs. But it is still unclear why competitive CoEAs lead to a better design than traditional EAs (Popovici et al., 2012; Rosin, 1997). We would like to understand how competitive CoEAs work on test-based optimisation problems from the simplest example. We formalise a general problem class, which includes Hillis' coevolutionary approach on sorting networks as follows: consider a function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, where $\mathcal{X}$ is a set of designs and $\mathcal{Y}$ is a set of test cases. We define $g(x, y) = 1$ if and only if design x passes test case y. Our optimisation problem can be defined as follows: $\arg\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)$. In

other words, the MAXIMIN Optimisation is to find $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ such that

$$\text{for all } (x, y) \in \mathcal{X} \times \mathcal{Y},\ g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y).$$

So here come the following research questions:

1. Under what circumstances can a competitive CoEA obtain an optimal solution in polynomial expected time?
2. How does the runtime depend on the problem (g) and on the algorithm?

To understand the questions above, we proceed by using runtime analysis. Runtime analysis of traditional evolutionary algorithms considers the time complexity of a given randomised algorithm. It provides either lower or upper bounds for the number of fitness function evaluations (called runtime) to understand the performance of given algorithms (Doerr and Neumann, 2020). Runtime analysis can identify relationships between algorithmic parameters and problem characteristics that determine the efficiency of evolutionary algorithms. We want to develop more runtime analysis for CoEAs and expect the insights from runtime analysis of CoEAs will improve the design of CoEAs (Popovici et al., 2012).

5.1.2 Our Contributions

On one hand, we prove that the traditional $(1, \lambda)$-EA is unable to solve the DIAGONAL Game within expected polynomial time. On the other hand, we rigorously demonstrate for the first time that, with the aid of coevolution, the $(1, \lambda)$-CoEA can solve DIAGONAL problems in expected polynomial time under certain conditions, utilising a two-phase analysis and tools from order statistics. These results highlight the promising potential of coevolutionary approaches for tackling binary adversarial optimisation problems.

The chapter is organised as follows: Section 5.2 introduces the necessary preliminaries and formal definitions. Section 5.3 demonstrates that traditional evolutionary algorithms

cannot efficiently solve the DIAGONAL function. Section 5.4 presents a competitive coevolutionary algorithm that efficiently solves the same function, highlighting its advantages. Section 5.5 provides experimental results to support the theoretical findings. Finally, Section 5.6 concludes the chapter with a discussion of implications and future directions.

5.2 Preliminaries

We focus on the search space $\mathcal{X} \times \mathcal{Y} = \{0,1\}^n \times \{0,1\}^n$ for any $n \in \mathbb{N}$. We consider the filtration $(\mathcal{F}_t)_{t\geq 0}$ including the information of $(X_0, Y_0), \ldots, (X_t, Y_t)$ in this dissertation. For any $\varepsilon \in [0,1]$ and any problem with an optimum (x^*, y^*), we say that algorithm A finds an ε-approximation to the optimum (x^*, y^*) in iteration $T \in \mathbb{N}$ if $||x^*|_1 - |x_T|_1| + ||y^*|_1 - |y_T|_1| < \varepsilon n$ where (x_T, y_T) is the search point of A at iteration T. "With high probability" is abbreviated to "w.h.p.". We define an event E_n with problem size $n \in \mathbb{N}$ that occurs w.h.p. if $\Pr(E_n) \geq 1 - O(1/n)$.

5.2.1 DIAGONAL Games

In order to model the binary test-based optimisation problem inspired by Hillis's method of sorting networks (Hillis, 1990), a payoff function with $\mathcal{X} \times \mathcal{Y}$ as input and $\{0,1\}$ as output is introduced as follows. Here $\mathcal{X} = \{0,1\}^n$ is the solution space of a set of designs for sorting networks and $\mathcal{Y} = \{0,1\}^n$ is the solution space of a set of test cases. We continue to consider a function $g : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. We define $g(x,y) = 1$ if and only if design x passes test case y. Our optimisation problem can be defined in terms of MAXIMIN optimisation: *find* $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ *such that*

$$\text{for all } (x,y) \in \mathcal{X} \times \mathcal{Y},\ g(x,y^*) \leq g(x^*,y^*) \leq g(x^*,y).$$

Recall Definition 4.4.3 in Chapter 2.2 (in Page 25): for $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}^n$, the payoff function $\textsc{Diagonal} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is

$$\textsc{Diagonal}(x,y) := \begin{cases} 1 & |y|_1 \leq |x|_1 \\ 0 & \text{otherwise.} \end{cases}$$

For brevity, we will refer to DIAGONAL as g in the remaining of this chapter. Notice that 1 means the design x passes the test cases $y \in \mathcal{Y}$. In this simple DIAGONAL game, y_n with $|y_n|_1 = n$ represents the most difficult test case, and x_n with $|x_n|_1 = n$ is the solution that can pass y_n (i.e. $g(x_n, y_n) = 1$). Thus, $(1^n, 1^n)$ is the MAXIMIN optimum in this case. In this optimum, neither the design nor the test case is willing to deviate from affecting their payoff $g(x,y)$ anymore. This exactly coincides with the definition of Nash equilibrium. This dissertation aims to explore whether the CoEAs can find such an optimal solution efficiently.

5.3 Traditional EA cannot Solve Diagonal Efficiently

In this section, we would like to explore whether traditional $(1,\lambda)$-EA can efficiently solve problems with only binary fitness. For a fair comparison between traditional evolutionary and coevolutionary algorithms, we chose the $(1,\lambda)$-EA (Algorithm 12) as the closest traditional evolutionary algorithm to the coevolutionary algorithm studied in this dissertation.

Algorithm 12 samples x uniformly at random. We define the same mutation operator $\mathcal{D}^t_{\text{mut}}$ for x. Ω is the sample space and $\omega_t \in \Omega$ means that the algorithm performs bit-wise mutation for each bit in the bit-string with probability χ/n where $\chi \in (0,1]$ in iteration t where x is of length n[1]. Next, we evaluate each individual by taking the fitness of i-th offspring. Then, until the termination criteria are met, only the bit-wise mutation operator

[1]We consider $\chi \in (0,1]$ including the default choice $\chi = 1$ used in Rowe and Sudholt, 2012.

Algorithm 12 $(1, \lambda)$-EA (Jagerskupper and Storch, 2007)

Require: Search spaces $\mathcal{X}$.

Require: Mutation $\mathcal{D}^t_{\text{mut}} : \Omega \times \{0,1\}^n \to \{0,1\}^n$.

Require: Payoff function $f : \mathcal{X} \to \mathbb{R}$.

1: Set $t := 1$ and choose $x_t \in \mathcal{X}$ uniformly at random.

2: **loop** until the termination criteria met

3: Set $t := t + 1$

4: Let $y_{t,1} := \mathcal{D}(\omega_t, x_{t-1}), \ldots, y_{t,\lambda} := \mathcal{D}(\omega_t, x_{t-1})$.

5: Choose $y_t \in \{y_{t,1}, \ldots, y_{t,\lambda}\}$ among all elements with the largest f-value.

6: $(1, \lambda)$-EA: Set $x_t := y_t$.

7: Go to 2.

mutates x. After that, Algorithm 12 selects the individual with the best fitness. If there is a tie in line 5 of Algorithm 12, then we consider choosing among all the individuals of the highest fitness uniformly at random. Next, we prove the following theorem.

Theorem 5.3.1. *Given any $\varepsilon \in (0, 1/4)$, $n \in \mathbb{N}$ and the function $f : \{0,1\}^{2n} \to \{0,1\}$ s.t. $z = (x, y)$ where $x, y \in \{0,1\}^n$ and $f(z) = \textsc{Diagonal}(x, y)$, the runtime of $(1, \lambda)$-EA with $\lambda = poly(n)$ and constant $\chi \in (0, 1]$ on finding an ε-approximation to the optimum (i.e. Nash Equilibrium) of f is at least $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.*

Proof. Note that the Nash Equilibrium of f are search points with $(1^n, \cdot)$. Note that $(1, \lambda)$-EA initialises the search point uniformly at random.

Before proving the result, first, we need some notation. We denote x_t, y_t by the binary bitstring for the leftmost part and rightmost part of the original bitstring and X_t, Y_t by the numbers of 1-bits of x, y respectively at iteration t. Let $M_t = n - X_t$. Next, to show the runtime's lower bound, we define the first hitting time $T_\varepsilon := \inf\{t > 0 \mid M_t \leq \varepsilon n\}$ for any $\varepsilon \in [0, 1)$. Note that X_0 are subject to Binomial distribution $\text{Bin}(n, 1/2)$. We compute the probability of $M_0 \leq (1 - \eta)n$ for any $\eta \in (0, 1)$.

$$\Pr\left(M_0 \leq (1-\eta)n\right) = \Pr\left(n - X_0 \leq (1-\eta)n\right) = \Pr\left(X_0 \geq (1+\eta)n\right)$$

By using Chernoff's bound (Theorem A.2.4) on $Z_0 \sim \mathrm{Bin}(n, 1/2)$, we have

$$\leq \exp\left(-\frac{n\eta^2}{6}\right) = e^{-\Omega(n)}. \tag{5.1}$$

This means that the initial search point X_0 satisfies $M_0 > (1-\eta)n$ with probability $1-e^{-\Omega(n)}$ for any constant $\eta \in (0,1)$ with overwhelmingly high probability. We satisfy one of the assumptions of Theorem 2.8.2 with high probability.

Next, suppose the numbers of 1-bits of λ offspring from x, y respectively at iteration t are denoted by

$$(X_t^{(1)}, Y_t^{(1)}), (X_t^{(2)}, Y_t^{(2)}), \cdots, (X_t^{(\lambda)}, Y_t^{(\lambda)}).$$

Based on the bitwise mutation operator, we have, for $i \in [\lambda]$,

$$X_t^{(i)} \sim \mathrm{Bin}(n - X_t, \frac{\chi}{n}) - \mathrm{Bin}(X_t, \frac{\chi}{n}) \text{ and } Y_t^{(i)} \sim \mathrm{Bin}(n - Y_t, \frac{\chi}{n}) - \mathrm{Bin}(Y_t, \frac{\chi}{n}). \tag{5.2}$$

Next, we divide the analysis into two phases.

(1) First, we show that after initialisation, w.h.p., the algorithm takes polynomial runtime to reach

$$C := \{(x,y) \in \{0,1\}^n \times \{0,1\}^n \mid |x|_1 \geq |y|_1\}.$$

Moreover, once the search point reaches C, $\Pr\left(X_t \geq n(1-c)\right) \geq e^{-\Omega(n)}$ for some constant $c \in (0, 1/2)$. In other words, it takes exponential time to reach $n(1-c)$ in x-direction w.h.p.

At iteration t, if $X_t \leq Y_t$, then there is nothing to show. Suppose $X_t > Y_t$ and consider the potential $H_t := Y_t - X_t$. Then, we define the runtime as

$$T_0 := \inf\{t > 0 \mid H_t \leq 0\}.$$

We first compute the drift:

$$\mathrm{E}(H_t - H_{t+1} \mid H_t) = \mathrm{E}(Y_t - Y_{t+1} \mid H_t) - \mathrm{E}(X_t - X_{t+1} \mid H_t).$$

Note that we consider the case where the algorithm has no selection pressure but mutation-driven. It performs a random walk towards C. Otherwise, if one of the offspring is in C, then our phase 1 is finished as we certainly select one of the offspring in C at the next iteration. Using Equation 5.2 gives

$$\begin{aligned}\mathrm{E}(H_t - H_{t+1} \mid H_t) &= (2Y_t - n)\frac{\chi}{n} - (2X_t - n)\frac{\chi}{n} \\ &= 2(Y_t - X_t)\frac{\chi}{n} \\ &= \frac{2\chi}{n}H_t.\end{aligned}$$

Using the multiplicative drift theorem with the upper tail bound gives

$$\Pr\left(T_0 > \frac{n\,(r + \ln n)}{2\chi}\right) \leq e^{-r}.$$

By choosing $r = n$, we have

$$\Pr\left(T_0 > \frac{n\,(n + \ln n)}{2\chi}\right) \leq e^{-n}.$$

Next, we consider $S_t := \mathbb{1}_{\{t<T_0\}} \cdot (n(1-c) - X_t) + \mathbb{1}_{\{t \geq T_0\}} \cdot n$ for some constant $c \in (0, 1/2)$ and define $T_1 = \inf\{t > 0 \mid S_t = 0\}$. Let us assume $n(1-c) > X_t > n/2$. Before $t < T_0$, the algorithm performs a random walk on search points with fitness 0. Thus, for each step and $t < T_0$,

$$\Pr\left(|S_t - S_{t+1}| \geq j\right) \leq \binom{n}{j}\left(\frac{\chi}{n}\right)^j \leq \frac{\chi^j}{2^j} \leq \frac{1}{2^j}.$$

We satisfy the exponential decay step size condition in Theorem 2.8.2. Next, we compute the negative drift from Eq. 5.2.

$$\mathrm{E}(S_t - S_{t+1} \mid S_t) \leq (n - 2X_t)\,\frac{\chi}{n} \cdot \mathbb{1}_{\{t<T_0\}} \leq -\frac{\chi}{n} < 0.$$

Thus, together, we use Theorem 2.8.2 with $a = 0, b = (1-c)n, \delta = \frac{\chi}{n}, \eta = r = 1$. We can summarise that there exists some constant $d > 0$ such that

$$\Pr\left(T_1 \geq e^{dn}\right) \geq 1 - e^{-\Omega(n)}.$$

This implies that after we finish Phase 1 and enter Phase 2, $X_t \geq (1-c)n$ w.h.p. for any constant $c \in (0, 1/2)$.

(2) Now, the algorithm enters the set C. We also assume that $\varepsilon n \leq M_t < cn$ for some constant c. We now analyse the drift in x-direction using the potential $M_t = n - X_t$. The drift consists of two parts (positive and negative drift, denoted by Δ_x^+ and Δ_x^-). Note that we can optimistically assume that we only flip zero bits and contribute to Δ_x^+. Each bit is flipped with probability $\frac{\chi}{n}$. Thus, we have the trivial upper bound:

$$\mathrm{E}\big(\Delta_x^+ \mid M_t\big) \leq cn \cdot \frac{\chi}{n} = c\chi.$$

We divide the negative drift part into two cases.

(2.1) If $X_t > Y_t$, then the algorithm can flip one of the zero bits to move backwards in x-direction. For y, we consider the case that it remains no bit flipped.

$$\Pr\left(Y_t = Y_{t+1}\right) = (1 - \frac{\chi}{n})^n \geq e^{-\chi}$$

We consider the lower bound for $\Pr\left(\Delta_x^- = -1\right)$.

$$\begin{aligned}\Pr\left(\Delta_x^- = -1\right) &\geq \Pr\left(\Delta_x^- = -1 \wedge Y_t = Y_{t+1}\right) \\ &\geq \binom{n-cn}{1}\frac{\chi}{n}e^{-\chi} = (1-c)\chi e^{-\chi}.\end{aligned}$$

Now, we can compute the negative drift:

$$\mathrm{E}\big(\Delta_x^- \mid M_t\big) \leq (-1) \cdot (1-c)\chi e^{-\chi}.$$

Together, if we choose a constant $0 < c < \frac{\chi}{e^\chi + 1}$, then

$$\begin{aligned}\mathrm{E}(M_t - M_{t+1} \mid M_t) = \mathrm{E}\big(\Delta_x^+ + \Delta_x^- \mid M_t\big) &\leq c\chi - (1-c)\chi e^{-\chi} \\ &\leq \chi\left(\left(1 + \frac{1}{e^\chi}\right)c - \frac{\chi}{e^\chi}\right) < 0.\end{aligned}$$

(2.1) If $X_t = Y_t$ and note that $M_t < cn$ which implies $X_t = Y_t \geq (1-c)n$, then

$$\begin{aligned}\Pr\left(\Delta_x^- = -1\right) \geq \Pr\left(\Delta_x^- = -1 \wedge Y_{t+1} - Y_t = -1\right) &\geq \left(\binom{n-cn}{1}\frac{\chi}{n}(1-\frac{\chi}{n})^{n-1}\right)^2 \\ &\geq \left((1-c)\chi e^{-\chi}\right)^2 \\ &\geq (1-2c)\chi^2 e^{-2\chi}.\end{aligned}$$

Now, we can compute the negative drift:

$$\mathrm{E}\big(\Delta_x^- \mid M_t\big) \leq (-1)\cdot(1-2c)\chi^2 e^{-2\chi}$$

Together, if we choose a constant $0 < c < \frac{\chi}{e^{2\chi}+2\chi}$, then

$$\begin{aligned}\mathrm{E}(M_t - M_{t+1} \mid M_t) = \mathrm{E}\big(\Delta_x^+ + \Delta_x^- \mid M_t\big) &\leq c\chi - (1-2c)\chi^2 e^{-2\chi} \\ &\leq \chi\left(\left(1+\frac{2\chi}{e^{2\chi}}\right)c - \frac{\chi}{e^{2\chi}}\right) < 0.\end{aligned}$$

Thus, by combing these two cases with $0 < c < \min\{\frac{\chi}{e^{\chi}+1}, \frac{\chi}{e^{2\chi}+2\chi}\}$, we obtain the negative drift $\mathrm{E}(M_t - M_{t+1} \mid M_t) < -f$ for some constant $f > 0$.

To verify the exponential decay step condition, we define the number of offspring such that $\Delta_t^{(i)} := X_t^{(i)} - Y_t^{(i)} \geq 0$ by N:

$$N := |\{i \in [\lambda]\} \mid \Delta_t^{(i)} \geq 0\}|.$$

We can see that for each $i \in [\lambda]$,

$$\Pr\left(\Delta_t^{(i)} \geq 0\right) \geq (1-\frac{\chi}{n})^{2n} \geq \frac{1}{2e^{2\chi}} := q_{N^*}$$

where we define $N^* \sim \mathrm{Bin}(\lambda, q_{N^*})$. Then, we can see that N stochastically dominates N^* (i.e. $N^* \preccurlyeq N$) since $N \sim \mathrm{Bin}(\lambda, \Pr\left(\Delta_t^{(i)} \geq 0\right))$ where $\Pr\left(\Delta_t^{(i)} \geq 0\right) \geq q_{N^*}$. By using stochastic dominance, we obtain: for any $\delta \in (0,1)$,

$$\Pr\left(N \leq (1-\delta)\lambda q_{N^*}\right) \leq \Pr\left(N^* \leq (1-\delta)\lambda q_{N^*}\right) = \Pr\left(N^* \leq (1-\delta)\mathrm{E}(N^*)\right)$$

Using Chernoff's bound (Theorem A.2.5) gives

$$\leq \exp\left(-\mathrm{E}(N^*)\delta/2\right) = e^{-\Omega(\lambda)}.$$

Note that $\Delta_t^{(i)} \geq 0$ means that we have $f(z_t^{(i)}) = 1$ and the above concentration result means that when sampling λ offspring, there are at least $(1-\delta)\lambda/2e^{2\chi}$ offspring of fitness 1 with probability $1 - e^{-\Omega(\lambda)}$. Then, for any $j \in \mathbb{N}_0$, the law of total probability gives

$$\begin{aligned}\Pr_t\left(|M_t - M_{t+1}| \geq j\right) &= \Pr_t\left(|M_t - M_{t+1}| \geq j \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) \cdot \Pr(N \geq \frac{\lambda}{4e^{2\chi}}) \\ &\quad + \Pr_t\left(|M_t - M_{t+1}| \geq j \mid N \leq \frac{\lambda}{4e^{2\chi}}\right) \cdot \Pr(N \leq \frac{\lambda}{4e^{2\chi}})\end{aligned}$$

Using $\Pr(N \geq \frac{\lambda}{4e^{2\chi}}) \leq 1$ and $\Pr(N \leq \frac{\lambda}{4e^{2\chi}}) \leq e^{-\Omega(\lambda)}$ give

$$\leq \Pr_t\left(|M_t - M_{t+1}| \geq j \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) + e^{-\Omega(\lambda)}$$

Recall that $X_t^{(i)}, Y_t^{(i)}$ denote the number of 1 bits of i-th offspring for x, y individual respectively (for $i \in [\lambda]$). Let us denote the difference between Hamming distance to the optimum for each i-th offspring (i.e $n - X_t - (n - X_t^{(i)}) = X_t^{(i)} - X_t$) by $Z_t^{(i)} := X_t^{(i)} - X_t$ and the event $\{|M_t - M_{t+1}| \geq j\}$ by E_j. We also denote the event $\{k\text{-th offspring is selected}\}$ by F_k. Using the law of total probability for conditional probability gives

$$= \sum_{k=1}^{\lambda} \Pr\left(E_j \mid N \geq \frac{\lambda}{4e^{2\chi}}, F_k\right) \Pr\left(F_k \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) + e^{-\Omega(\lambda)}$$

When conditioning on F_k, the event E_j becomes $|Z_t^{(k)}| \geq j$. Thus, we have

$$= \sum_{k=1}^{\lambda} \Pr\left(|Z_t^{(k)}| \geq j \mid N \geq \frac{\lambda}{4e^{2\chi}}, F_k\right) \Pr\left(F_k \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) + e^{-\Omega(\lambda)}$$

We can upper bound the probability of $|Z_t^{(k)}| \geq j$ by only considering flipping j bits among n-bitstring. Thus,

$$\leq \sum_{k=1}^{\lambda} \binom{n}{j} (\frac{\chi}{n})^j \Pr\left(F_k \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) + e^{-\Omega(\lambda)}$$

Using $\binom{n}{j} \leq \frac{n^j}{j!}, j! \geq 2^j$ and $\chi \in (0, 1]$ gives

$$\leq \sum_{k=1}^{\lambda} \frac{1}{2^j} \Pr\left(F_k \mid N \geq \frac{\lambda}{4e^{2\chi}}\right) + e^{-\Omega(\lambda)}$$

Recall that N denotes the number of the offspring of fitness 1 (i.e.$\Delta_t^{(i)} \geq 0$) and F_k denotes the event that the k-th offspring will be selected. As mentioned, when the algorithm comes across a tie, we select the offspring uniformly at random among those offspring of fitness 1. Thus, we get

$$\leq \lambda \cdot \frac{1}{2^j} \cdot \frac{1}{N} + e^{-\Omega(\lambda)}$$

Also, note that we condition on the event that $N \geq \lambda/4e^{2\chi}$.

$$\leq \frac{4e^{2\chi}}{2^j} + e^{-\Omega(\lambda)} \leq \frac{8e^{2\chi}}{2^j}.$$

Now, we satisfy condition (2) in Theorem 2.8.2 with $r = 8e^{2\chi} = o(n/\log n)$ and $\eta = 1$.

Note that there is a failure event such that all the offspring will be in $\{0,1\}^{2n} \setminus C$ (with fitness 0). We know that each offspring has at least a constant probability $1/2e^{2\chi}$ to stay in C. The probability of this failure event is at most $(1 - 1/2e^{2\chi})^{\lambda} = e^{-\Omega(n)}$ with $\lambda = \text{poly}(n)$. We optimistically assume that we have a runtime $T_\varepsilon = 0$ in any failure events. By using the law of total probability, we can conclude that the runtime of $(1,\lambda)$-EA with $\lambda = \text{poly}(n)$ and constant $\chi \in (0,1]$ on finding an ε-approximation to the optimum (i.e. Nash Equilibrium) of f is at least $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.

□

From our analysis of the $(1,\lambda)$-EA on Diagonal, Theorem 5.3.1 shows that for any constant $\varepsilon \in (0, 1/4)$, the standard $(1,\lambda)$-EA cannot find any ε-approximation of the Maximin optimum of Diagonal efficiently. Moreover, by Markov's inequality:

$$E[T_\varepsilon] \geq e^{cn} \Pr(T_\varepsilon \geq e^{cn}) = e^{cn}(1 - e^{-\Omega(n)}) = e^{\Omega(n)}.$$

Diagonal only consists of binary values, resulting in a very hard and flat fitness landscape in the search space. It leads to a random walk of the search point on the search space, which consists of fitness 1 (i.e. in a grey area of Figure 5.1). A further insight is that

traditional EAs (e.g. $(1, \lambda)$-EA, $(1+1)$ EA) cannot cope well with the interaction between x and y since these algorithms might only favour the highest (or lowest) fitness other than NASH equilibrium in DIAGONAL[2]. The approach of considering a pair of individuals from two distinct populations ($x \in \mathcal{X}$ and $y \in \mathcal{Y}$) as a single entity within one population (represented as $z := (x, y)$) fails to capture the complexity of interactions between these two populations. So is there any alternative evolutionary approach that could help us to resolve this issue?

5.4 Competitive Coevolution Solves Diagonal Efficiently

After our analysis, we know that $(1, \lambda)$-EA cannot solve the DIAGONAL Games efficiently. We are wondering whether a competitive CoEA exists that can solve this problem by changing the selection mechanism or increasing the size of offspring. The key is to properly capture the complex interaction between populations so the algorithm can find NASH equilibrium efficiently. So, we extend the traditional evolutionary algorithm in the context of competitive CoEAs and consider the $(1, \lambda)$-variant of CoEAs. Then, we prove $(1, \lambda)$-CoEA solves the DIAGONAL problem in expected polynomial runtime.

Algorithm 13 samples both design x and test case y uniformly at random. We define the same mutation operator $\mathcal{D}^t_{\text{mut}}$ for both x and y. Ω is the sample space and $\mathcal{D}^t_{\text{mut}}(x)$ means that for any $x \in \{0,1\}^n$, the algorithm performs bit-wise mutation for each bit in the bit-string with probability χ/n for $\chi \in (0, n)$ in iteration t. Then, instead of using pairwise dominance, we evaluate each design by taking the payoff of i-th offspring against the parent opponent and evaluate each test case by taking the payoff of it against the parent design for λ offspring. Then, until the termination criteria are met, only the bit-wise mutation operator mutates either x or y in an alternating manner. After that, Algorithm 13 selects

[2]The proof of Theorem 5.3.1 can be generalised to a broader class of traditional EAs

Algorithm 13 $(1, \lambda)$-CoEA (Alternating Update)

Require: Search spaces $\mathcal{X}, \mathcal{Y}$.

Require: Mutation $\mathcal{D}^t_{\text{mut}}(x) : \Omega \to \{0,1\}^n$ for all $x \in \{0,1\}^n$

Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

Sample $x \sim \text{Unif}(\mathcal{X})$; Sample $y \sim \text{Unif}(\mathcal{Y})$.
for $t \in \{1, 2, \dots\}$ **do**
    if $t \bmod 2 = 0$ **then**
        for $i = 1$ to λ **do**
            $x^i \sim \mathcal{D}^t_{\text{mut}}(x)$;
        $x := \arg\max_{i \in [\lambda]} g(x^i, y)$;
    else
        for $i = 1$ to λ **do**
            $y^i \sim \mathcal{D}^t_{\text{mut}}(y)$;
        $y := \arg\max_{i \in [\lambda]} -g(x, y^i)$;

the pair of the design and the test case of the best fitness. In the following analysis, we define $f(x^i) := g(x^i, y)$ and $h(y^i) := -g(x, y^i)$ where x, y, x^i, y^i are defined in Algorithm 13. In terms of tie-breaking, Algorithm 13 selects uniformly at random among the optima. In this dissertation, in order to achieve polynomial runtime for Algorithm 13, we restrict $\lambda \in \text{poly}(n)$.

5.4.1 Characteristic Lemma for Alternating Update

In this section, without loss of generality, we write the λ offspring at generation $t \in \mathbb{N}$ in descending order in their 1-norms: $|x_t^{(1)}|_1 \geq |x_t^{(2)}|_1 \geq \cdots \geq |x_t^{(\lambda)}|_1$ and $|y_t^{(1)}|_1 \geq |y_t^{(2)}|_1 \geq \cdots \geq |y_t^{(\lambda)}|_1$. We also use $X_t^{(i)}$ and $Y_t^{(i)}$ to denote $|x_t^{(i)}|_1$ and $|y_t^{(i)}|_1$ respectively. $X_t := |x_t|_1$ and $Y_t = |y_t|_1$ where $x_t, y_t \in \{0,1\}^n$ are the current search point at iteration $t \in \mathbb{N}$. To

avoid disrupting the flow of this chapter, we defer the proofs of the following technical lemmas to the appendix.

Lemma 5.4.1. *Consider the predator payoff of x-bitstring denoted by f and the prey payoff of y-bitstring denoted by h in Algorithm 13,*

(1) If $|x^{(1)}|_1 \geq |x^{(2)}|_1$, then $f(x^{(1)}) \geq f(x^{(2)})$.

(2) If $|y^{(1)}|_1 \geq |y^{(2)}|_1$, then $h(y^{(1)}) \geq h(y^{(2)})$.

By Lemma 5.4.1, if all λ offspring have the same 1-norms, then they have the same fitness. In this case, the current search points in the algorithm will correspond to random walks in the search space. The algorithm makes actual progress based on argmax selection mechanism when "crossing the diagonal". Next, we rigorously define it.

Definition 5.4.1 (Cross the diagonal)**.** *Given Algorithm 13 applied to* DIAGONAL*, and for current search point $(x_t, y_t) \in \{0,1\}^n \times \{0,1\}^n$, assume that we have the λ offspring at iteration $t \in \mathbb{N}$ in descending order in their 1-norms. We say that λ offspring cross the diagonal* ***horizontally*** *at iteration t, if there exist some k such that $|x_t^k|_1 \geq |y_t|_1$ with $|y_t|_1 > |x_t|_1$; We say that λ offspring cross the diagonal* ***vertically*** *at iteration t if there exist some ℓ such that $|y_t^\ell|_1 > |x_t|_1$ with $|x_t|_1 \geq |y_t|_1$. We say λ offspring cross the diagonal if either occurs.*

Definition 5.4.1 means that when crossing the diagonal, the fitness of either the x-bitstring or the y-bitstring strictly improves. Next, we introduce the concept of a c-tube (the purple strip as presented in Fig. 5.1).

Definition 5.4.2 (c-tube)**.** *Define $C := \{(x, y) \in \mathcal{X} \times \mathcal{Y} \mid ||x|_1 - |y|_1| < c\}$ c-tube. We say a current search point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ lies outside the c-tube if $(x, y) \notin C$.*

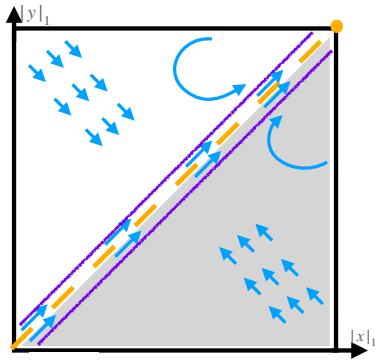


Figure 5.1: Example of DIAGONAL problem and drift tendency. The horizontal axis represents the number of 1-bits in the designs x, and the vertical axis represents the number of 1-bits in the test cases y. The grey area represents search points of payoff 1, and the rest represents search points with payoff 0.

5.4.2 Phase 1

Algorithm 13 updates the search point in an alternating manner. From the characteristic lemma and definition of crossing the diagonal, we know when $Y_t - X_t > 0$, Algorithm 13 makes progress on searching the optimum by updating X_t to let it cross the diagonal and vice versa. In the analysis, we want to avoid the case when $Y_t - X_t > 0$, but Algorithm 13 updates Y_t instead of X_t in c-tube. This inspires the following definition. We define a successful cycle formally.

Definition 5.4.3. *Given $c > 0$, then for all $t \in \mathbb{N}$, a cycle in iteration $2t$ is called successful with respect to c if $X_{2t} + c > Y_{2t} > X_{2t}$, $Y_{2t+1} + c > X_{2t+1} \geq Y_{2t+1}$ and $X_{2t+2} + c > Y_{2t+2} > X_{2t+2}$. A cycle in iteration $2t+1$ with respect to c is called successful if $Y_{2t+1} + c > X_{2t+1} \geq Y_{2t+1}$, $X_{2t+2} + c > Y_{2t+1} \geq X_{2t+2}$ and $X_{2t+3} + c > Y_{2t+3} \geq Y_{2t+3}$.*

A successful cycle implies that the algorithm crosses the diagonal twice without leaving the c-tube. We show that the search point will move along the diagonal towards the optimum. We use $H_t := 2n - (X_t + Y_t)$ as the potential function to show that there is a positive drift towards the optimum $(X_t, Y_t) = (n, n)$ although with a small probability of

escaping from the c-tube.

In the following analysis, we consider a deterministic initialisation to simplify the analysis. The following analysis sufficiently covers the core principle of how Algorithm 13 works via competitive coevolution. We consider an initialisation at $(0^n, 0^n)$ which is the farthest search point with respect to the optimum (n, n) in terms of Hamming distance. First, we present our main lemma in this subsection.

Lemma 5.4.2. *Assume that Algorithm 13 is initialised with* $(X_0, Y_0) = (0, 0)$. *For all* $t \in \mathbb{N}$, *we define* $D_t := |X_t - Y_t|$. *For any constant* $\varepsilon \in (0, 1)$, *define* $T := \inf\{t > 0 \mid H_t := 2n - X_t - Y_t < \varepsilon n\}$ *and for all* $\tau \in \mathbb{N}$, *let* E_τ *denote the event that the algorithm has* τ *consecutive successful cycles with respect to* c *or* $\min_{t\in[2\tau]} H_t < \varepsilon n$. *If there exists* $c > 0, p_c, p_e \in [0, 1]$ *s.t. for all* $t \in [1, T/2)$,

(1) $\Pr(X_{2t+1} \geq Y_{2t+1} \mid X_{2t} < Y_{2t}) \geq 1 - p_c$;

(2) $\Pr(X_{2t+2} < Y_{2t+2} \mid X_{2t+1} \geq Y_{2t+1}) \geq 1 - p_c$;

(3a) $\Pr(D_{2t+1} > c \mid D_{2t} < c) \leq p_e$;

(3b) $\Pr(D_{2t+2} > c \mid D_{2t+1} < c) \leq p_e$,

then $\Pr(E_\tau) \geq 1 - 2\tau(p_c + p_e)$.

Conditions $(1), (2)$ mean that the search point crosses the diagonal at iteration either $2t$ or $2t + 1$ with probability at least $1 - p_c$. Condition (3) means that the search point escapes from the c-tube at iteration $t + 1$ with probability at most p_e for all $t \in [1, T)$. So Lemma 5.4.2 gives a lower bound for the probability that the algorithm has τ consecutive successful cycles.

We will use Lemma 5.4.2 to wrap up all the arguments with $\tau = \Omega(n)$ and $\lambda = n^{\Omega(1)}$ later. We proceed with Phase 2 by assuming a successful cycle always exists in the analysis.

In other words, when $Y_t > X_t$, a successful cycle guarantees that we update X_t and when $Y_t \leq X_t$, a successful cycle guarantees that we update Y_t. We will compute p_c and p_e in Phase 2.

5.4.3 Phase 2

Let us define $D_t := |X_t - Y_t|$ to be the distance of the current pair of search points (x, y) to the diagonal. After the search point crosses the diagonal, we start Phase 2. We divide Phase 2 into three sections. Firstly, we show that given a search point stays within some c-tube, the λ offspring cross the diagonal with probability $1-2\left(\frac{1}{\lambda}\right)^{\frac{1}{2eX}}$. Meanwhile, the search point escapes from the c-tube with probability bounded from below by $\frac{1}{\lambda^{\Omega(1)}}$. Secondly, we show that we always have a positive drift when the search point crosses the diagonal. Finally, we wrap up everything using a restart argument, which accounts for the failed generations.

5.4.3.1 Phase 2.1

Firstly, we need some lemmas to formulate the most likely scenarios when λ offspring are produced.

Lemma 5.4.3. *(Cameron, 1994) Given a binomial random variable $Z \sim Bin(n, p)$, $\Pr(Z \text{ is even}) = \frac{1}{2} + \frac{1}{2} \cdot (1-2p)^n$.*

We will use Lemma 5.4.3 to show the following Lemma 5.4.4. We see from Lemma 5.4.4 that we need a sufficiently large offspring size to avoid the case that $X_t^{(1)}$ coincides with $X_t^{(\lambda)}$ and guarantee that there is some offspring identical to the parents pair with high probability.

Lemma 5.4.4. *Given problem size $n \in \mathbb{N}$, offspring size $\lambda \in \mathbb{N}$, iteration $t \in \mathbb{N}$, and*

mutation rate $\chi = O(1)$*, if* t *is even, then*

$$\Pr\left(\exists k \in [\lambda], x_t^k = x_t\right) \geq 1 - e^{-\Omega(\lambda)}, \quad \Pr\left(\max_{i\in[\lambda]} X_t^i > \min_{i\in[\lambda]} X_t^i\right) \geq 1 - e^{-\Omega(\lambda)}.$$

If t *is odd, then*

$$\Pr\left(\exists \ell \in [\lambda], y_t^\ell = y_t\right) \geq 1 - e^{-\Omega(\lambda)}, \quad \Pr\left(\max_{i\in[\lambda]} Y_t^i > \min_{i\in[\lambda]} Y_t^i\right) \geq 1 - e^{-\Omega(\lambda)}.$$

Next, we provide some useful concentration inequalities, which can be used to show how much deviation is made by each i-th offspring in the number of 1-bits. To obtain the concentration, we proceed by using Moment Generating Functions (MGFs).

Lemma 5.4.5. *For* $n \in \mathbb{N}$ *and any* $s \in [0..n]$*, we define* $U = V_1 - V_2$ *where* $V_1 \sim Bin(n - s, \chi/n)$ *and* $V_2 \sim Bin(s, \chi/n)$ *are independent, with* $\chi < n$*. The MGF (moment generating function)* $M_U(\eta) \leq \exp\left(\chi(e^\eta - 1)\right)$ *for any* $\eta > 0$*.*

Lemma 5.4.6. *With the same setting as Lemma 5.4.5, for any* $s \geq 0$ *and* $\lambda \geq 1$*,* $\Pr(U \geq s) \leq \lambda^\chi e^{-(s \ln\ln \lambda + \chi)}$*. Furthermore, for any* $s \geq e^2\chi$ *and any* $\lambda \geq 1$*,* $\Pr(U \geq s) \leq e^{-(\chi+s)}$*.*

Given Algorithm 13 with constant $\chi > 0$, we define the number of 1-bits in each offspring $i \in [\lambda]$ in t iteration as $X_t^{(i)}$. Given the current number of 1-bits $s \in [n]$ for the parent solution, we define the change of the number of 1 in X_t for each offspring by $\Delta X_t^{(i)} \sim V_1 - V_2$ where $V_1 \sim Bin(n - s, \frac{\chi}{n})$ and $V_2 \sim Bin(s, \frac{\chi}{n})$. So $\Delta X_t^{(i)}$ has the same MGFs from Lemma 5.4.5 for each $i \in [n]$. From Lemma 5.4.6, for any $i \in [\lambda]$ and $s > 0$, we have $\Pr\left(\Delta X_t^{(i)} > s\right) \leq e^{-\chi}\lambda^\chi e^{-s \ln\ln \lambda}$.

5.4.3.2 Phase 2.2

Next, we show in a certain c-tube that the search point crosses the diagonal with high probability, resulting in the positive drift towards the optimum. First, we show the search point induced by Algorithm 13 crosses the diagonal w.h.p.

Lemma 5.4.7. *Given problem size $n \in \mathbb{N}$, offspring size $\lambda \in \mathbb{N}$, iteration $t \in \mathbb{N}$, let $c := \kappa \ln \lambda / \ln \ln \lambda$ for any constant $\kappa \in (0,1)$, we denote E_t to be the event that the algorithm crosses the diagonal as defined in Definition 5.4.1 at iteration $t+1$. Assume any constants $\chi > 0$ and $\varepsilon \in (0,1)$. If $D_t < c$, $n - X_t \geq \varepsilon n$ and $n - Y_t \geq \varepsilon n$, and if either of the two conditions holds (1) t is even and $X_t < Y_t$; (2) t is odd and $X_t \geq Y_t$, then* $\Pr(E_t) \geq 1 - 2\lambda^{-\frac{1}{2e\chi}}$

Lemma 5.4.7 states that if the search point lies in a given tube of width c, then before reaching an ε-approximation, the search point keeps crossing the diagonal at next iteration with high probability.

Next, we show that the search point deviates from some c-tube with a small probability. The idea is to show that the algorithm is more likely to select the offspring which lie inside the c-tube. We first prove some lemmas to proceed.

Lemma 5.4.8. *Let problem size $n \in \mathbb{N}$, offspring size $\lambda \in \mathbb{N}$, iteration $t \in \mathbb{N}$, $n - X_t \geq \varepsilon n$ and $n - Y_t \geq \varepsilon n$ for any constant $\varepsilon \in (0,1)$ and constant mutation rate $\chi \in (0,1)$. For any $\kappa \in (\chi, (1+\chi)/2)$, let $c := \kappa \ln \lambda / \ln \ln \lambda$,*

(1) if t is even and $X_t < Y_t$, then c satisfies

(A) $\Pr\left(\max_{i\in[\lambda]} \Delta X_t^i \geq D_t \mid D_t < c\right) \geq 1 - 2\lambda^{-\frac{1}{2e\chi}}$

(B) $\Pr\left(K/M \leq 2(1+\delta)\left(1/\lambda^{\kappa-\chi}\right)\right) \geq 1 - e^{-\Omega(\lambda)}$ *for any constant* $\delta \in (0,1)$

where $\Delta X_t^i := X_t^i - X_t$, and $K = |\{i \mid \Delta X_t^i \geq D_t + c\}|$ and $M = |\{i \mid \Delta X_t^i \geq D_t\}|$.

(2) If t is odd and $X_t \geq Y_t$, then c satisfies

(C) $\Pr\left(\max_{i\in[\lambda]} \Delta Y_t^i \geq D_t \mid D_t < c\right) \geq 1 - 2\lambda^{-\frac{1}{2e\chi}}$

(D) $\Pr\left(K'/M' \leq 2(1+\delta)\left(1/\lambda^{\kappa-\chi}\right)\right) \geq 1 - e^{-\Omega(\lambda)}$ *for any constant* $\delta \in (0,1)$

where $\Delta Y_t^i := Y_t^i - Y_t$, and $K' = |\{i \mid \Delta Y_t^i \geq D_t + c\}|$ and $M' = |\{i \mid \Delta Y_t^i \geq D_t\}|$.

Notice that in Lemma 5.4.8, ΔX_t^i denotes the change of number of 1-bits in the i-th offspring of x_t, and $K = |\{i \mid \Delta X_t^i \geq D_t + c\}|$ is the number of samples/offspring s.t. $\Delta X_t^i \geq D_t + c$ and $M = |\{i \mid \Delta X_t^i \geq D_t\}|$ is the number of samples/offspring s.t. $\Delta X_t^i \geq D_t$ and similar for y_t. Conditions (A) and (C) in Lemma 5.4.8 mean that for sufficiently large λ, the next search point produced by Algorithm 13 crosses the diagonal with high probability. Conditions (B) and (D) in Lemma 5.4.8 mean that in those offspring which cross the diagonal, the portion of offspring which make a large jump to cross outside the c-tube is rare with overwhelmingly high probability.

Lemma 5.4.9. *(Escape from the c-tube with small probability) Assume that the conditions of Lemma 5.4.8 hold. Consider* $(1, \lambda)$*-CoEA on* DIAGONAL*. We define* $H_t := 2n - X_t - Y_t$*. Let* $T := \inf\{t > 0 \mid H_t \leq \varepsilon n\}$ *for any constant* $\varepsilon \in (0, 1)$*. For any* $t \in [1, T]$*, any constants* $\chi \in (0, 1)$ *and* $\gamma \in (0, (1 - \chi)/2)$*, we have*

$$\Pr(D_{t+1} > c \mid D_t < c) \leq 9 \left(\frac{1}{\lambda}\right)^{\gamma}$$

where c is defined in Lemma 5.4.8.

Lemma 5.4.9 means that if the search point lies in a given tube of width c, then before reaching an ε-approximation, the search point stays in the given tube with probability $1 - O\left(1/\lambda^{\gamma}\right)$ for constant $\gamma > 0$.

We use Lemma 5.4.8 to show that, firstly, within a c-tube, the search point can cross the diagonal with high probability. Then, we consider two extreme cases when the search point lies in the tube. Assume $Y_t > X_t$, the first one is that the search point is close to the upper boundary ($Y = X + c$ in Figure 5.1, with Hamming distance 1), then we need to show that the search point can still cross the diagonal but not cross too much and escape from the lower boundary ($Y = X - c$ in Figure 5.1) with high probability. The second extreme case is if the search point stays very close to the diagonal (with Hamming distance 1). It is a challenge to show that the next search point will not escape from the lower

boundary since we have already shown within the tube that there is a high probability that the offspring jumps at least Hamming distance D_t to cross the diagonal. So, it is the case that a few offspring may escape from the lower boundary. The key is to observe that more offspring that cross the diagonal lie inside the tube compared with those outside the tube. With low mutation rates, the probability of flipping r bits is monotonically decreasing in r. Hence, many offspring only flip a few bits. All the offspring in Algorithm 13 which cross the diagonal have fitness 1 since the offspring will lie in the grey area of Figure 5.1. Then Algorithm 13 selects the next search point uniformly at random from these offspring crossing the diagonal and with an overwhelmingly higher probability to select those inside the tube since most of the offspring sharing the fitness 1 still lie inside the tube.

Lemma 5.4.10. *Let problem size $n \in \mathbb{N}$, offspring size $\lambda \in \mathbb{N}$, iteration $t \in \mathbb{N}$, $n - X_t \geq \varepsilon n$ and $n - Y_t \geq \varepsilon n$ for any constant $\varepsilon \in (0,1)$. For any $\chi \in (0,1)$, let $D_t \in [0,c]$ where $c = (\frac{1+3\chi}{4}) \ln \lambda / \ln \ln \lambda$. If t is even and $Y_t > X_t$, then $X_{t+1} - X_t \geq 1$ with probability at least $1 - O(1/\lambda^{(1-\chi)/4})$. If t is odd and $Y_t \leq X_t$, then $Y_{t+1} - Y_t \geq 1$ with probability at least $1 - O(1/\lambda^{(1-\chi)/4})$. Moreover, let $H_t := 2n - X_t - Y_t$, then $\mathrm{E}_t(H_t - H_{t+1}) \geq 1/2$.*

We defer the proof to Appendix A.6. From Lemma 5.4.10, before reaching an ε-approximation, if the search point stays within the c-tube, there exists a positive constant drift towards the optimum when considering H_t the Hamming distance to (n,n). Since we show the existence of positive drift, next we come to the main theorem of this chapter.

5.4.3.3 Phase 2.3

Now, we wrap up everything and use a restart argument to compute the overall runtime using Lemma 5.4.2 and Lemma 5.4.10. We will restart the algorithm every $2T_c$ generation with $x_0 = y_0 = 0^n$ and $T_c := 4(2-\delta)n$ for any constant $\delta \in (0,1)$. Given the problem size $n \in \mathbb{N}$, we have the main result:

Theorem 5.4.1. *Consider the* $(1,\lambda)$*-CoEA with constant mutation rate* $\chi \in (0,1)$*, and offspring size* $\lambda \geq (264(2-\delta)n)^{4/(1-\chi)}$ *for any constant* $\delta \in (0,1)$*. Assume that the algorithm is initialised at* $x = y = 0^n$*, and restarted at* $x = y = 0^n$ *every* $2T_c$ *generations, with* $T_c := 4(2-\delta)n$*. Then the expected time to find a* δ*-approximation to the* MAXIMIN *optimum of* DIAGONAL *is at most* $24(2-\delta)\lambda n$*.*

Proof. Assume that before reaching a δ-approximation to the MAXIMIN optimum of DIAGONAL, Algorithm 13 starts from $(X_0, Y_0) = (0,0)$, t is even if $X_t < Y_t$ and t is odd if $X_t \geq Y_t$. We use $H_t = 2n - X_t - Y_t$ as our potential function. We define $D_t = |X_t - Y_t|$ as the distance function away from the diagonal in $|x|_1$-$|y|_1$ plane. We define $T := \inf\{t > 0 \mid H_t \leq \delta n\}$ for any constant $\delta \in (0,1)$. Then we want to show $\mathrm{E}(T) \leq 2(2-\delta)n$. Note that c defined in Lemma 5.4.10 and

$$\begin{aligned} T &:= \inf\{t > 0 \mid H_t \leq \delta n\} \\ &\leq \inf\{t > 0 \mid D_t < c \wedge H_t \leq \delta n\}. \end{aligned}$$

We replace δ with 3ε where $\varepsilon \in (0, 1/3)$ and $c = \frac{1+3\chi}{4 \ln \ln \lambda} \ln \lambda$.

$$= \inf\{t > 0 \mid D_t < c \wedge H_t \leq 3\varepsilon n\} := T_{c,3\varepsilon}.$$

If $D_t < c$, without loss of generality assume $X_t \geq Y_t$, then $D_t < c$ implies that $n - Y_t \geq n - X_t, X_t - Y_t < c$ and moreover $H_t > 3\varepsilon n$ implies that $n - X_t > 3\varepsilon n - n + Y_t > 3\varepsilon n - n + X_t - c$. This is equivalent to $n - X_t > \frac{3\varepsilon n - c}{2}$. Thus, if $D_t < c$ and $H_t > 3\varepsilon n$, then

$$n - X_t > \frac{3\varepsilon n - c}{2} \text{ and } n - Y_t > \frac{3\varepsilon n - c}{2}.$$

In Lemma 5.4.10, we assume $c = \frac{\frac{1+3\chi}{4} \ln \lambda}{\ln \ln \lambda} = O(\ln \lambda) = o(n)$ where the last equality follows from the assumption in this dissertation $\lambda = \mathrm{poly}(n)$. For sufficiently large n, we have for any $\varepsilon \in (0, 1/3)$,

$$n - X_t > \varepsilon n \text{ and } n - Y_t > \varepsilon n.$$

Now, we meet the conditions of Lemma 5.4.10. Under this setting, by Lemma 5.4.10, $\mathrm{E}_t(H_t - H_{t+1}) \geq 1/2$. By the additive drift theorem He and Yao, 2001,

$$\mathrm{E}(T) \leq \mathrm{E}(T_{c,3\varepsilon}) \leq \frac{2n - 3\varepsilon n}{1/2} = \frac{(2-3\varepsilon)n}{1/2} = 2(2-\delta)n.$$

Now, consider the failure event that a cycle (consecutive steps) fails to be successful at some iteration t. We want to use Lemma 5.4.2 to restart the analysis if such a failure occurs. To clarify our argument, recall from Definition 5.4.3 that a cycle is two consecutive generations of the algorithm. A cycle can be *successful* (see Definition 5.4.3). We call T_c such cycles a T_c-*cycle phase*, and we have a failure in a T_c-cycle phase if any of the T_c cycles is unsuccessful. Note that the event that a successful cycle in Definition 5.4.3 breaks is equivalent to the event that either the search point escapes from the c-tube or does not cross the diagonal during the cycle. By Lemma 5.4.7, the probability that the search point does not cross the diagonal is at most $p_c = 2\lambda^{-1/2e^{\chi}}$. By Lemma 5.4.9, the probability that the search point lies outside the c-tube where c in Lemma 5.4.9 is at most $p_e = 9(\frac{1}{\lambda})^{(1-\chi)/4}$. By Lemma 5.4.2, we have

$$\Pr(\text{The algorithm has } T_c \text{ consecutive successful cycles}) \geq 1 - 2T_c(p_c + p_e)$$

Substituting $p_c = 2(\frac{1}{\lambda})^{1/2e^{\chi}}$ and $p_e = 9(\frac{1}{\lambda})^{(1-\chi)/4}$ gives

$$\geq 1 - 2T_c\left(2(\frac{1}{\lambda})^{1/2e^{\chi}} + 9(\frac{1}{\lambda})^{(1-\chi)/4}\right).$$

Recall that $e^{-\chi} \geq \frac{1-\chi}{2} \geq \frac{1-\chi}{4}$ for $\chi \in (0,1)$. Then, we have

$$\geq 1 - 2T_c \cdot 11(\frac{1}{\lambda})^{(1-\chi)/4}.$$

Picking $T_c = 4(2-\delta)n$ and $\lambda \geq (264(2-\delta)n)^{4/(1-\chi)}$ gives

$$\geq 1 - \frac{4(2-\delta)n \cdot 22}{264(2-\delta)n} = \frac{2}{3}.$$

This means that the algorithm has $4(2-\delta)n$ successful consecutive cycles with probability at least $\frac{2}{3}$. If the algorithm fails to have such a successful consecutive cycles, then we restart

the algorithm. In a successful phase, the algorithm finds a δ-approximation in expected time $2(2-\delta)n$ generations. By Markov's inequality, the algorithm finds a δ-approximation within $4(2-\delta)n$ generations with probability at least $1/2$ (otherwise we consider this a failure). By taking a union bound on the failure events (i.e. an unsuccessful T_c-cycle phase or the algorithm does not find the δ-approximation within the end of the phase), the algorithm finds the δ-approximation in a phase of length T_c cycles with probability $p \geq 1 - 1/3 - 1/2 = 1/6$. Hence, we multiply a factor of 6 to obtain the overall expected runtime is at most $4(2-\delta)n/p \leq 24(2-\delta)n$.

Notice that we only compute the number of generations so far. To obtain the runtime of algorithms (number of function evaluations), we need to multiply λ for each iteration. So we end up with the expected runtime of Algorithm 13 is $4\lambda(2-\delta)n$ for $\lambda = \Omega(n^{\frac{4}{1-\chi}})$.

□

Theorem 5.4.1 shows that a large offspring population size helps the CoEA cross the diagonal consistently with high probability, which is thus favoured during selection and finally leads to the positive drift towards the optimum. Note that we require the algorithm to start with $(0^n, 0^n)$. For uniformly at random initialisation, we conjecture the runtime is asymptotically the same. We simplify the analysis by considering the base case $(0^n, 0^n)$. Lemma 5.4.8 and Lemma 5.4.9 show the necessity of large offspring size. Otherwise, a small number of offspring will let CoEA fall to the flat fitness landscape apart from the tube along the diagonal and then get lost in the random walk around the search space. To summarise, with the proper design of coevolution and large offspring population size, $(1, \lambda)$-CoEA can find an ε-approximation to the optimum of DIAGONAL efficiently.

5.5 Experiments

Our theoretical results only consider approximation, while the experimental results consider finding the exact optimum. To complement our asymptotic results with data for concrete problem sizes, we conduct the following experiments. The main focus of the experiments is to explore the impact of mutation rates χ and how the runtime scales up with the problem size n.

5.5.1 Settings

We conduct experiments with the $(1, \lambda)$-CoEA on the DIAGONAL with $n = \lambda = 1000$ and the initialisation of the algorithm is set up as uniformly at random. We also set different mutation rates in the range of 0.2 to 2.2 in increments of 0.2 (explore the error threshold of mutation rate, often greater than 1.0) and conduct 100 independent runs for each configuration to explore the suitable range of mutation rate for the DIAGONAL problem. The budget for each run is set to be 10^9 function evaluations.

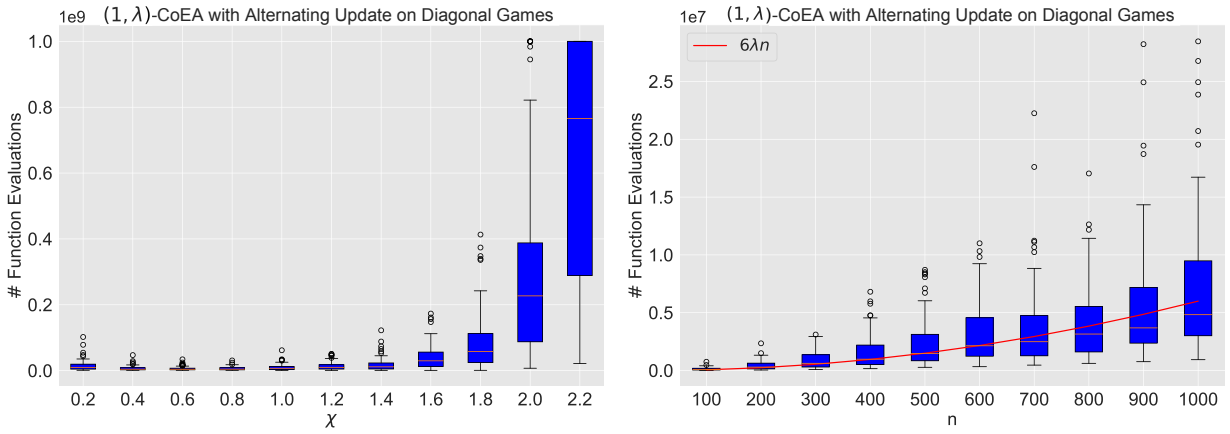


Figure 5.2: Runtime of $(1, \lambda)$-CoEA on DIAGONAL. **Left**: Runtime against different mutation rates under $n = \lambda = 1000$. **Right**: Runtime against specific $\chi = 0.6$. The red curve is $f(n, \lambda) = 6\lambda n$ where in this case we set $n = \lambda$.

Then, as Fig 5.2 shows, $\chi = 0.6$, the expected performance of $(1, \lambda)$-CoEA is the best. Then, we fix such a mutation rate, and run the experiments in the range of $n = 100$ to $n = 1000$ in increments of 100. We conduct 100 independent runs for each configuration. The budget for each run is 10^9 function evaluations.

5.5.2 Results

Fig 5.2 displays the runtime of $(1, \lambda)$-CoEA on DIAGONAL for different mutation rates from 0.2 to 2.2. This data confirms that for the suitable low mutation rates and sufficiently large offspring size, Algorithm 13 finds the optimum efficiently. The higher mutation rate eventually leads to the inefficiency of the algorithm. As we observe in Fig 5.2, under suitable mutation rate $\chi = 0.6$, the empirical average of the runtime is bounded above by $O(\lambda n)$. Notice that our theoretical analysis requires $\lambda = \Omega(n^{4/(1-\chi)})$, while it seems that $\lambda = O(n)$ in the experiments is already sufficient to guarantee a polynomial runtime for $(1, \lambda)$-CoEA on DIAGONAL. This suggests that the current bound may not be tight and our theoretical analysis still has room to improve.

5.6 Discussion and Conclusion

CoEAs exhibit complex dynamics on MAXIMIN optimisation. To the best of our knowledge, this dissertation is the first runtime analysis of CoEAs on binary test-based adversarial optimisation problems. As a starting point, we propose a binary test-based problem called DIAGONAL. We showed that for DIAGONAL, the $(1, \lambda)$-EA get trapped in a binary fitness landscape. Thus, traditional $(1, \lambda)$-EA failed to find any approximation to the optimum in polynomial runtime due to negative drift induced by the flat fitness landscape. However, for the $(1,\lambda)$-CoEA with the alternating update method, if the offspring population is sufficiently large $\lambda = \Omega(n^{4/(1-\chi)})$ with a reasonable constant mutation rate $\chi \in (0, 1)$, it

can find an approximation to optimum efficiently in expected runtime $O(\lambda n)$. We want to highlight the necessity of coevolution and large offspring size in solving these binary problems in which the fitness landscape is very flat and hard to search from these analyses.

On the technical side, this dissertation shows that mathematical runtime analyses are also feasible for the $(1, \lambda)$-CoEA. We are optimistic that our tools will widen the toolbox for future analyses of competitive CoEAs. On the practical side, it brings insight for practitioners that traditional EAs may not be well-suited for DIAGONAL-like problems. We suggest using CoEAs with large offspring population sizes and relatively low mutation rates, when searching for Nash Equilibria on binary problems with large flat regions in the payoff landscape.

For future work, it is interesting to provide a more precise upper bound for the runtime of $(1, \lambda)$-CoEAs on DIAGONAL and a more general analysis by relaxing the deterministic initialisation, since the empirical results suggest our theoretical bound might not be tight. Using more advanced theoretical tools like (Lehre and Lin, 2024a), we can derive a better tail bound of the current runtime for $(1, \lambda)$-CoEA. It is also worth exploring whether there are more efficient ways to capture the interaction between two populations, for example, by combining coevolution with self-adaptation (Hevia Fajardo et al., 2024; Qin and Lehre, 2022) or multi-objective optimisation (Benford et al., 2024). Furthermore, it is exciting to explore the behaviour of CoEAs on a more general class of payoff functions like generalised boundary test-based problems.

Chapter Six

Single-Pair vs Population-Based CoEAs on Sparse Binary Zero-Sum Games

This chapter is based on the following publication:

Towards Runtime Analysis of Population-Based Coevolutionary Algorithms on Sparse Binary Zero-Sum Games (Lehre and Lin, 2025) which is published in Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI'25).

6.1 Background

The maximin problem is a significant class of optimisation problems in machine learning, with applications in adversarial optimisation, training, game playing, and matrix games (Daskalakis and Panageas, 2018; Gidel et al., 2017; Goodfellow et al., 2020; Grand-Clément and Kroer, 2021; Lin et al., 2020; Mertikopoulos et al., 2018; Palaniappan and Bach, 2016; Robey et al., 2024; Wei et al., 2021; Zhang et al., 2022). When gradient information is available, gradient descent ascent (GDA) is a natural approach, especially for bilinear objectives like $f(x, y) = x^T Ay$ (Daskalakis and Panageas, 2018; Gidel et al., 2017). However, when gradients are inaccessible but the problem is convex-concave, zeroth-order methods can provide solutions (Beznosikov et al., 2020; Dvinskikh et al., 2022; Gasnikov et al., 2022). The absence of gradients or a nonconvex-nonconcave structure, however, makes these problems intrinsically more challenging (Colson et al., 2007).

Theoretical analysis of evolutionary algorithms on discrete maximin problems is limited. Recent studies (Fajardo et al., 2023; Lehre, 2022) have conducted rigorous runtime analyses of CoEAs on discrete maximin problems such as Discrete-Bilinear, demonstrating that single-pair CoEAs and population-based CoEAs with pairwise dominance can solve Discrete-Bilinear in polynomial runtime. However, there is a gap in the theoretical understanding of CoEAs on more challenging discrete games with binary outputs (i.e. $g : \mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$), as originated from seminal works on co-evolving sorting networks and test cases by Hillis (1990) and co-evolving players in the game of Backgammon by Pollack and Blair (1998). Lehre and Lin (2024c) showed that $(1, \lambda)$-CoEA can find the approximation of the optimum of the Diagonal game in polynomial runtime with high probability (see Chapter 5). Moreover, another variant of Diagonal called the bigger-number game was considered in (Zhang and Sandholm, 2024) to show the inefficiency of the Double Oracle Algorithm in zero-sum games. However, we hypothesise that games with binary payoffs are challenging because the analytic tool developed in Chapter 2 cannot be

used directly in our scenario. Thus, we adapt the level-based theorem for coevolution to study this problem (see Theorem 6.3.1).

Our Contributions: This chapter tackles open challenges in zero-sum games with sparse binary payoff signals, presenting the first rigorous runtime analysis of CoEAs with pairwise dominance on DIAGONAL game. Specifically, we demonstrate that single-pair CoEAs, such as Random Local Search with Pairwise Dominance (RLS-PD) and (1+1)-CoEA, cannot find the optimum of DIAGONAL in polynomial runtime with high probability. Additionally, we show that a population-based CoEA (PDCoEA) can find the optimum in polynomial runtime under large population sizes and low constant mutation rates. Our experimental results further illustrate the practical implications of our theoretical bounds and examine the error threshold for mutation rates in PDCoEA. These findings highlight the promising potential of coevolution for binary maximin problems and reveal the importance of low mutation rates and large population sizes in maintaining a "coevolutionary arms race", where, in coevolution, when one population changes, the others co-adapt, and repeating this feedback loop yields individuals that perform increasingly better by an external metric (i.e., payoff) as mentioned in Section 2.7.2.

The chapter is organised as follows: Section 6.1 introduces the necessary background for coevolution and adversarial optimisation and the contribution of this chapter. Section 6.2 introduces the necessary preliminaries and formal definitions. Section 6.3 introduces a refined level-based theorem based on the previous level-based theorems for coevolution (Lehre, 2022) and (Fajardo and Lehre, 2024). Section 6.4 demonstrates that single-pair CoEAs, including RLS-PD and (1+1)-CoEA, cannot efficiently solve the DIAGONAL function. Section 6.5 presents a population-based coevolutionary algorithm, PDCoEA, that efficiently solves the same function, highlighting the necessity of a population and a low mutation rate. Section 6.6 provides experimental results to support our theoretical findings. Finally, Section 6.7 concludes the chapter with a discussion of implications and future directions.

6.2 Preliminaries

In this chapter, "with high probability" is abbreviated to "w.h.p." We say an event E_n with problem size $n \in \mathbb{N}$ occurs w.h.p. if $\Pr(E_n) \geq 1 - d/n^c$ for some $c \in \mathbb{N}$ and constant $d > 0$. $o(1)$ of any function $f(n)$ where $n \in \mathbb{N}$ means that $\lim_{n\to\infty} f(n) = 0$. $x \in \text{poly}(n)$ denotes that there exist constants c, d such that $x \leq cn^d$. NE denotes a Nash Equilibrium. We also denote $X_t := |x_t|_1 \in [n] \cup \{0\}$ for $x_t \in \{0,1\}^n$ and $Y_t := |y_t|_1 \in [n] \cup \{0\}$ for $y_t \in \{0,1\}^n$ for any $n \in \mathbb{N}$. We consider the filtration $(\mathcal{F}_t)_{t\geq 0}$ including the σ-algebra of $(X_0, Y_0), \ldots, (X_t, Y_t)$.

6.2.1 Binary Zero-Sum Games

Given a two-player game with strategy spaces $\mathcal{X}$ and $\mathcal{Y}$, the payoff functions $g_1, g_2 : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ are defined for player 1 and player 2, respectively. The game is zero-sum if player 1's gain is exactly equal to player 2's loss (and vice versa), meaning $g_1(x, y) + g_2(x, y) = 0$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. If $g_1(x, y), g_2(x, y) \in \{-1, 1\}$ for all $x, y \in \mathcal{X}$, the game is called binary. Given a binary function $g : \mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$, we refer to the game with payoff functions $g_1(x, y) = g(x, y)$ and $g_2(x, y) = -g(x, y)$ as the *binary zero-sum game* defined by g. Many classical games where the outcomes are win/lose, such as Nim, Backgammon, Chess, or Go (without draw), can be represented by a binary function g by setting $g(x, y) = 1$ if and only if player 1 wins and $g(x, y) = -1$ otherwise.

Sparsity: This chapter adopts the notion of sparsity from (Auger et al., 2015), i.e., they assume that the two vectors of the NE x^* and y^* have support of moderate size k in mixed strategies, i.e. $k << 2^n$ where each player has 2^n pure strategies. Note that this is a different notion from games with a sparse payoff matrix. In this chapter, we consider the sparsity in the sense that the support of mixed strategies (non-zero probability terms) is far fewer than the size of pure strategies each player can take.

DIAGONAL Games: To model the binary maximin optimisation problem originating from Hillis's seminal work on co-evolving sorting networks and test cases (Hillis, 1990), Lehre and Lin (2024c) introduced a class of payoff functions with $\mathcal{X} \times \mathcal{Y}$ as the input and $\{-1, 1\}$ as the output, namely DIAGONAL. Here, $\mathcal{X} = \{0,1\}^n$ represents the set of genotypes corresponding to designs for sorting networks, while $\mathcal{Y} = \{0,1\}^n$ denotes the set of genotypes for test cases. $g(x, y) = 1$ if and only if design x passes test case y. The optimisation problem is to find $(x^*, y^*) \in \mathcal{X} \times \mathcal{Y}$ such that

$$\text{for all } (x, y) \in \mathcal{X} \times \mathcal{Y},\ g(x, y^*) \leq g(x^*, y^*) \leq g(x^*, y).$$

Sparse binary games like DIAGONAL and its variant have been studied in the literature (Auger et al., 2015; Benford and Lehre, 2024b; Lehre and Lin, 2024c; Zhang and Sandholm, 2024).

Definition 6.2.1 (Lehre and Lin, 2024c)**.** *Given $n \in \mathbb{N}$, $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$, the payoff function DIAGONAL : $\mathcal{X} \times \mathcal{Y} \to \{-1, 1\}$ is*

$$\textsc{Diagonal}(x, y) := \begin{cases} 1 & |y|_1 \leq |x|_1 \\ -1 & \textit{otherwise.} \end{cases}$$

For notational brevity, we will denote the function DIAGONAL by g in this chapter. A payoff of 1 indicates that design x passes the test cases y; otherwise, the payoff is -1. In the DIAGONAL game, $y^* = 1^n$ is the hardest test case, which only the solution $x^* = 1^n$ can pass (i.e., $g(x^*, y^*) = 1$). Thus, (x^*, y^*) is the unique maximin optimum, where neither the design nor the test case deviates without affecting their payoff $g(x, y)$. This aligns with the NE in zero-sum games. This chapter investigates whether CoEAs can efficiently find the NE of DIAGONAL.

To distinguish DIAGONAL from the commonly used ONEMAX benchmark in the theory of evolutionary computation (Doerr and Neumann, 2019), we highlight key differences. Unlike ONEMAX, DIAGONAL is more challenging. ONEMAX uses a linear fitness function

(payoff independent of the opponent), while DIAGONAL is a game where the payoff depends on the opponent. DIAGONAL provides only 1 bit of information per evaluation, compared to $\log(n)$ bits in ONEMAX. Additionally, ONEMAX has an ascending fitness landscape, making it easy for algorithms like (1+1)-EA or RLS to follow, whereas the binary payoff landscape in DIAGONAL is flat, rendering single-pair CoEAs inefficient.

6.2.2 Coevolutionary Algorithms and Characteristic Lemma of Dominance Relation

A broad class of coevolutionary processes, defined by Algorithm 5 update two populations P, Q. At each iteration, a new pair of solutions is sampled from P, Q via the interaction of two populations. Such an interaction includes variation, evaluation, selection, etc. The population is then updated with this new pair of solutions, and the process continues until the termination criteria are met. For single-pair CoEAs, we denote the current pair in generation t by (x_t, y_t). These stochastic processes track the dynamics of coevolutionary algorithms, and we will analyse their first hitting time on the target set. Next, we introduce the concept of pairwise dominance, which is key to characterising the operator $\mathcal{D}$ of CoEAs.

Lehre (2022) demonstrated that a population-based CoEA called Pairwise Dominance CoEA (PDCoEA) can solve some instances of BILINEAR (a bi-linear payoff function with real-valued rather than binary payoffs) in polynomial runtime with high probability. Fajardo et al. (2023) shows random local search with pairwise dominance (RLS-PD) solves some instances of BILINEAR in expected polynomial runtime (as summarised by the Table in the Appendix). Therefore, we aim to explore whether such a pairwise dominance method can also work for our binary maximin optimisation problem. Here, we provide a definition of the pairwise dominance relation (c.f. Definition 6.2.2).

Definition 6.2.2 (Pairwise Dominance (Lehre, 2022))**.** *Given a function* $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$

and two pairs $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$, (x_1, y_1) *dominates* (x_2, y_2) *with respect to* g, *denoted* $(x_1, y_1) \succeq_g (x_2, y_2)$, *if and only if* $g(x_1, y_2) \geq g(x_1, y_1) \geq g(x_2, y_1)$.

In order to understand how the search point will move under the pairwise dominance relation, we characterise the relationship among search points in DIAGONAL games.

Definition 6.2.3. *Given a pair of search points* $(x, y) \in \mathcal{X} \times \mathcal{Y}$, *we say* (x, y) *lies above the diagonal if* $|x|_1 < |y|_1$ *and* (x, y) *lies below the diagonal if* $|x|_1 \geq |y|_1$.

Next, we derive sufficient and necessary conditions for the dominance relation of DIAGONAL games. We defer all the proofs to the appendix.

Lemma 6.2.1. *Given a* DIAGONAL *game denoted by* g, $(x, y) \succeq_g (u, v)$ *if and only if* $(x, y), (u, v)$ *satisfy one of the following:*

(1) $|x|_1 < |v|_1, |x|_1 < |y|_1$ *and* $|u|_1 < |y|_1$;

(2) $|x|_1 \geq |v|_1, |x|_1 < |y|_1$ *and* $|u|_1 < |y|_1$;

(3) $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ *and* $|u|_1 < |y|_1$;

(4) $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ *and* $|u|_1 \geq |y|_1$.

6.3 A Refined Level-Based Theorem

To analyse the runtime of population-based evolutionary algorithms, we define runtime.

Definition 6.3.1. *For any instance* A *of Algorithm 5 and* $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$, *define* $T_{A,\mathcal{S}} := \min\{t\lambda \in \mathbb{N} \mid (P_t \times Q_t) \cap \mathcal{S} \neq \emptyset\}$.

Fajardo et al. (2023) and Lehre (2022) have analysed CoEAs on BILINEAR, as mentioned in the introduction, DIAGONAL and other binary maximin problems exhibit a more challenging and sparse payoff landscape, causing the original methods to be inapplicable. The

original coevolutionary level-based theorem does not make any assumption about the initialisation of the populations. Here, we will require a variant of the theorem where the populations are assumed to be initialised at a given level. The proof is almost identical. A similar modification can be found in (Fajardo and Lehre, 2024).

Theorem 6.3.1. *Given subsets $A_j \subseteq \mathcal{X}, B_j \subseteq \mathcal{Y}$ for $j \in [m]$ where $A_1 = \mathcal{X}$ and $B_1 = \mathcal{Y}$, define $T := \min\{t\lambda \mid (P_t \times Q_t) \cap (A_m \times B_m) \neq \emptyset\}$, where for all $t \in \mathbb{N}, P_t \in \mathcal{X}^\lambda$ and $Q_t \in \mathcal{Y}^\lambda$ are the populations of Algorithm 5 in generation t. Denote the current level of the population by $j := \max\{\ i \in [m-1] \mid |(P \times Q) \cap (A_i \times B_i)| \geq \gamma_0\lambda^2\}$ where $\gamma_0 \in (0,1)$.*

Given $m_2 \in \mathbb{N}$ where $m_2 < m$, suppose $P_0 \times Q_0$ is initialised with the current level $j \geq m_2$. If there exist $z_{m_2}, \ldots, z_{m-1}, \delta \in (0,1)$ such that for any populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ with the current level $j \geq m_2$,

(G1) for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq z_j$;

(G2a) for all $\gamma \in (0, \gamma_0)$, if $|(P \times Q) \cap (A_{j+1} \times B_{j+1})| \geq \gamma\lambda^2$, then for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq (1+\delta)\gamma$;

(G2b) for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_j) \Pr(y \in B_j) \geq (1+\delta)\gamma_0$;

(G3) and the population size $\lambda \in \mathbb{N}$ satisfies $\lambda \geq c' \log(m/z_*)$ for a sufficiently large constant c', where $z_: = \min_{m_2 \leq i \leq m-1} z_i$;

then there exists a constant $c'' > 0$ such that any $r > 0$,

$$\Pr\left(T \geq r\frac{c''\lambda}{\delta}\left(\lambda^2(m-m_2) + \sum_{i=m_2}^{m-1}\frac{1}{z_i}\right)\right) \leq \frac{1+o(1)}{r}.$$

6.4 Exponential Lower Bounds for RLS-PD and (1+1)-CoEA on Diagonal Game

In this section, we focus on the single-pair CoEA with pairwise dominance.

Algorithm 14 Single-Pair CoEA with Pairwise Dominance

Require: Payoff function, $g : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$.

Require: Mutation operator, $\texttt{mutate} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$.

1: Sample x, y uniformly at random from $\{0,1\}^n$;

2: **for** $t \in \{1, 2, \dots\}$ **do**

3: $\quad (x', y') = \texttt{mutate}(x, y)$;

4: $\quad$ **if** $(x', y') \succeq_g (x, y)$ **then** $(x, y) := (x', y')$;

In RLS-PD, the local mutation operator `mutate` creates x', y' by copying x and y, and flipping exactly one bit chosen uniformly at random from either x or y. In (1+1)-CoEA, the bit-wise mutation operator `mutate` creates x' and y' by copying x, y and flipping each bit independently with probability χ/n, where $\chi \in (0, n)$.

6.4.1 RLS-PD on Diagonal

The first competitive CoEA we consider is RLS-PD. We show that it fails to find the maximin optimum of Diagonal games in polynomial runtime with overwhelming high probability.

RLS-PD samples both the design x and the test case y uniformly at random. Then, until the termination condition is met, either x or y is mutated by the local mutation operator. After that, RLS-PD selects the pair of designs and test cases based on pairwise dominance as defined in Definition 6.2.2. Unlike the success of RLS-PD on Bilinear (Fajardo et al.,

2023), it fails to find the maximin optimum of DIAGONAL in polynomial runtime.

Theorem 6.4.1. *Consider RLS-PD on DIAGONAL and problem size $n \in \mathbb{N}$. The runtime of RLS-PD for finding the maximin optimum of DIAGONAL is $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.*

Theorem 6.4.1 shows that the algorithm performs a random walk before reaching the optimum. In this case, the selection mechanism fails to distinguish which individual is fitter if search points share the same payoff. Theorem 6.4.1 also suggests that DIAGONAL is a challenging and non-trivial maximin problem.

There are various simple and effective mutation operators used in evolutionary algorithms (Doerr and Neumann, 2019). RLS-PD employs the local mutation operator, which is a commonly used mutation operator in evolutionary algorithms. This operator picks any bit position uniformly at random and flips the selected bit. Another well-known mutation operator is the bit-wise mutation operator, which flips each bit in the bit-string with probability χ/n where the mutation rate $\chi \in (0, n)$ is some constant. The bit-wise mutation performs a more diverse search in the Hamming neighbourhood of the given search point. What if we modify the mutation operator from local mutation to bit-wise mutation? The bit-wise mutation operator potentially allows the search point to make more significant jumps in the search space at one iteration. Can the algorithm then find the optimum more efficiently? To answer these questions, we analyse the runtime of (1+1)-CoEA on DIAGONAL.

6.4.2 (1+1)-CoEA and DIAGONAL Game

We consider (1+1)-CoEA with the pairwise dominance relation. We want to explore the impact of changing the mutation operator compared to RLS-PD and whether this new variant can find the optimum more efficiently.

(1+1)-CoEA samples both the design x and the test case y uniformly at random. Then, until the termination criteria are met, unlike RLS-PD, both x and y are mutated by a bit-wise mutation operator. After that, (1+1)-CoEA selects the pair of designs and test cases based on the pairwise dominance relation from Definition 6.2.2. We use Lemma 6.2.1 to show condition (2) in Negative Drift Theorem (Doerr and Neumann, 2019; Oliveto and Witt, 2012; Rowe and Sudholt, 2012).

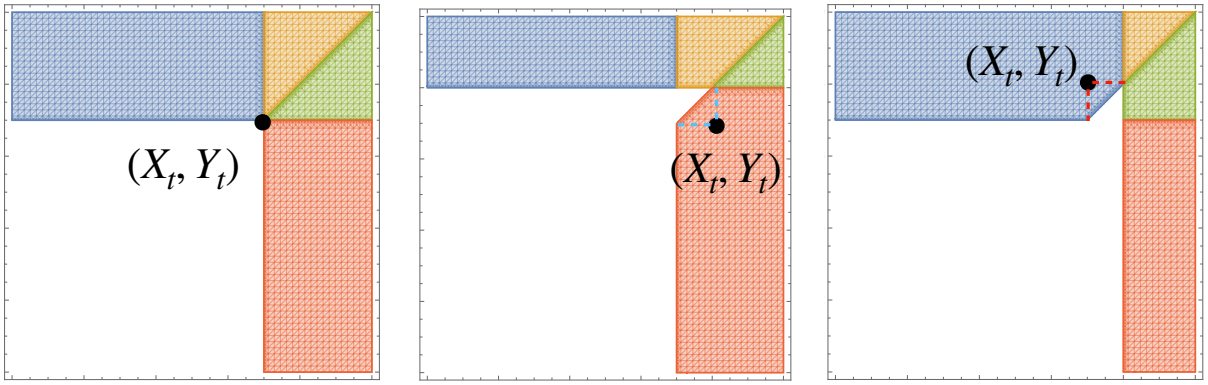
Recall that X_t, Y_t denote the number of 1-bits in both predator and prey bitstrings.

Lemma 6.4.1. *Consider (1+1)-CoEA with pairwise dominance relation on* DIAGONAL *and a constant mutation rate $\chi = O(1)$. Suppose we have a search point $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ in iteration $t \in \mathbb{N}$ and define $M_t := 2n - (X_t + Y_t)$. For any t, if $\varepsilon n < M_t \leq n/4$ for some constant $\varepsilon > 0$, then there exist some $r, \eta > 0$ s.t. for any $j \geq 0$, $\Pr(|M_t - M_{t+1}| \geq j \mid \mathcal{F}_t) \leq r/(1+\eta)^j$.*

Lemma 6.4.1 shows that (1+1)-CoEA makes big jumps with exponential decay probability, which satisfies the step size condition in the Negative Drift Theorem. To see how to bound the negative drift from (1+1)-CoEA, we visualise Lemma 6.2.1. In Figure 6.1, the coloured regions represent the potential areas where the search point may move in the next iteration under the pairwise dominance relation. We consider the Manhattan distance between the current search point and the optimum, denoted as $M_t = 2n - (X_t + Y_t)$. Negative drift occurs when the search point moves to the blue and red regions. Positive drift occurs when the search point moves to the orange and green regions.

Lemma 6.4.2. *With the same setting as Theorem 6.4.2 and sufficiently large $n \geq 1$, for any t and $M_t := 2n - X_t - Y_t$, if $\varepsilon n < M_t \leq 2\varepsilon n$ for any constant $\varepsilon \in (0, 1/20]$, then there exists $\delta > 0$ such that $\mathrm{E}_t(M_t - M_{t+1}) \leq -\delta$.*

Once we show the existence of negative drift in Lemma 6.4.2, we can use the Negative Drift Theorem with Lemma 6.4.1 to derive Theorem 6.4.2.



(a) $X_t = Y_t$ (b) $X_t > Y_t$ (c) $X_t < Y_t$

Figure 6.1: Visualisation of Lemma 6.2.1. The colour area represents the search point $(x, y) \in \mathcal{X} \times \mathcal{Y}$ s.t. $(x, y) \succeq_g (x_t, y_t)$ where $X_t = |x_t|_1, Y_t = |y_t|_1$. The blue region corresponds to case (1), the orange region corresponds to case (2), the green region corresponds to case (3) and the red region corresponds to case (4) in Lemma 6.2.1.

Theorem 6.4.2. *Consider (1+1)-CoEA with pairwise dominance relation on* DIAGONAL*, constant mutation parameter* $\chi \in (0, 1]$*, and problem size* $n \in \mathbb{N}$*. The runtime of (1+1)-CoEA for finding the maximin optimum of* DIAGONAL *is* $e^{\Omega(n)}$ *with probability* $1 - e^{-\Omega(n)}$*.*

Theorem 6.4.2 shows that (1+1)-CoEA also fails to find the desired optimum for DIAGONAL in polynomial time. The proof for Theorem 6.4.2 is more complex than that of Theorem 6.4.1 because the current search point of the (1+1)-CoEA can potentially make a significant jump in the search space, while RLS-PD only makes a 1-bit-step local search around the given search point.

From our analysis of RLS-PD and (1+1)-CoEA on DIAGONAL, Theorem 6.4.1 and Theorem 6.4.2 both show that with overwhelmingly high probability (i.e., $1 - e^{-\Omega(n)}$), single-pair CoEAs with pairwise dominance cannot find the maximin optimum of DIAGONAL efficiently, regardless of which mutation operators the algorithms use. DIAGONAL consists only of binary values, resulting in a very flat payoff landscape in the search space, unlike

problems with real-valued payoff functions, such as BILINEAR (Fajardo et al., 2023; Hevia Fajardo and Lehre, 2023). Such a payoff landscape is challenging for CoEAs.

6.5 PDCoEA Finds the NE of Diagonal Efficiently

6.5.1 Pairwise Dominance CoEA

From our previous analysis, we know that (1+1)-type CoEAs with pairwise dominance cannot solve DIAGONAL efficiently. We now ask whether a competitive CoEA can solve this problem by changing its selection mechanism or increasing its population size. To address this question, we analyse the runtime of PDCoEA on DIAGONAL in the following section.

Algorithm 15 samples both the designs P_0 and the test cases Q_0 uniformly at random. Before mutation, Algorithm 15 uses 2-tournament selection to compare the pairs of designs and test cases. First, two pairs are sampled uniformly at random (lines 6-7), and then they are compared based on the pairwise dominance relation from Definition 6.2.2 (lines 8-9). Finally, both $x \in P_t$ and $y \in Q_t$ are mutated by a bit-wise mutation operator.

In this section, we use Theorem 6.3.1 to prove that PDCoEA can efficiently find the optimum of the DIAGONAL games. In general, level-based analysis partitions the whole search space into several levels and analyses the upgrade among different levels. Theorem 6.3.1 enables us to track the state of the algorithm and provides the tail bound on the number of function evaluations until the current population contains an individual hitting a target set (set to be our final level), given that conditions (G1)-(G3) hold. We defer the proofs of (G1)-(G3) conditions to the appendix.

Before showing our main result (Theorem 6.5.1), we define the following pixels and par-

Algorithm 15 Pairwise Dominance CoEA (Lehre, 2022)

Require: Payoff function $g : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$.

Require: Population size λ and mutation rate $\chi \in (0, n]$.

for $i \in [\lambda]$
  Sample $P_0(i)$ uniformly at random from $\{0,1\}^n$;
  Sample $Q_0(i)$ uniformly at random from $\{0,1\}^n$;
for $t \in \{0, 1, 2, \dots\}$ until termination criterion met **do**
  for $i \in [\lambda]$ **do**
    Sample $(x_1, y_1) \sim \text{Unif}(P_t \times Q_t)$;
    Sample $(x_2, y_2) \sim \text{Unif}(P_t \times Q_t)$;
    if $(x_1, y_1) \succeq_g (x_2, y_2)$ **then**$(x, y) := (x_1, y_1)$
    else$(x, y) := (x_2, y_2)$
    Obtain x' by flipping each bit in x with prob. $\frac{\chi}{n}$;
    Obtain y' by flipping each bit in y with prob. $\frac{\chi}{n}$;
    Set $P_{t+1}(i) := x'$ and $Q_{t+1}(i) := y'$.

titions that we will use in Theorem 6.3.1. We define the pixels: $j \in \{1, \dots, n\}$,

$$\begin{aligned} A_j &:= \{x \in \{0,1\}^n \mid |x|_1 = j\} \text{ and} \\ B_j &:= \{y \in \{0,1\}^n \mid |y|_1 = j\}. \end{aligned} \tag{6.1}$$

Next, we define the following partitions that we will use in Theorem 6.3.1. Given $(U_0 \times V_0), \dots, (U_n \times V_n)$, we define $U_0 = \mathcal{X}$ and $V_0 = \mathcal{Y}$ and for $j \in \{1, \dots, n\}$, where we suppose $n = 4k$ (and thus $n/2 = 2k$ to simplify the notation during our calculation. One can use a different alternating partition for odd n)

$$\begin{aligned} U_{2j} &:= A_{2j} \text{ and } U_{2j+1} := A_{2j+1} \cup A_{2j+2}; \\ V_{2j} &:= B_{2j} \cup B_{2j+1} \text{ and } V_{2j+1} := B_{2j+1}. \end{aligned} \tag{6.2}$$

Given two populations of solutions $P_t \in \mathcal{X}^\lambda, Q_t \in \mathcal{Y}^\lambda$, we define the current level by $j := \max\{i \in [n] \cup \{0\} \mid |(P_t \times Q_t) \cap (U_i \times V_i)| \geq \gamma_0 \lambda^2\}$ where $\gamma_0 \in (0,1)$ is constant and

$\lambda \in \mathbb{N}$ is the population size. To estimate the runtime of a population-based evolutionary algorithm, we consider the number of function evaluations until the population contains at least one individual arriving at the final level (i.e. the first hitting time of the population of solution hitting the final level multiplied by the population size λ): $T := \inf\{\lambda t > 0 \mid (P_t \times Q_t) \cap (U_n \times V_n) \neq \emptyset\}$.

6.5.2 Technical Lemmas for Level-Based Analysis

In order to use Theorem 6.3.1, we need to satisfy (G1), (G2a), (G2b) and (G3). (G3) is easily satisfied by setting $\lambda = \Omega(\log n)$. So the rest of this section is to show conditions (G1), (G2a) and (G2b)[1].

6.5.3 Ensuring Condition (G1)

In this section, we compute the upgrade probability in condition (G1). It would be hard for the algorithm to upgrade from $A \times B$ to $A_1 \times B_1$ when the population falls off the diagonal (i.e., None of individual in the population has $X_t = Y_t$). To resolve this, we assume the algorithm initialises the population at level $n/2$ and the population will keep hill-climbing along the diagonal from level $n/2$ to level n. It is also worth noting that the original tool (Level-Based Theorem Lehre, 2022) does not consider the population's initialisation and thus cannot be directly applied to our problem.

Lemma 6.5.1. *Given two populations P and Q in PDCoEA with population size $\lambda \in$ poly(n) and constant mutation rate $\chi \in (0, \ln 2/2)$ on* Diagonal *with problem size $n \in \mathbb{N}$, assume PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$. For the*

[1]We provide a visualisation of level partitions for PDCoEA on Diagonal in the appendix.

current level $j \geq n/2$, there exists $z_{n/2}, \ldots, z_{n-1} \in (0,1)$ such that for $(x,y) \sim \mathcal{D}(P,Q)$,

$$\Pr(x \in U_{j+1}) \cdot \Pr(y \in V_{j+1}) \geq z_j.$$

Lemma 6.5.1 shows that if the populations in PDCoEA are initialised at level $j = n/2$ and the current level is above $n/2$, then we can obtain a positive upgrade probability.

6.5.4 Ensuring Condition (G2a)

For PDCoEA, recall that $\mathcal{D}(P,Q)$ in Algorithm 5 is $\mathcal{D} = \textsc{Mutate} \circ \textsc{Select}$, where SELECT corresponds to lines 6-9, and MUTATE corresponds to lines 10-12 in Algorithm 15. We denote the SELECT probability by $\Pr_{\text{select}}$ and the MUTATE probability by $\Pr_{\text{mutate}}$. Note that (G2a) in Theorem 6.3.1 keeps track of the multiplicative growth on the next current level. In the following analysis, without loss of generality, we assume the current level j is even to simplify the calculation. Before showing the next lemma, we need to make some assumptions: before reaching the NE in each iteration t, for the current level j, there exists a constant (with respect to χ) $\alpha \in (0,1)$ such that

$$(\text{A} \mid \text{G2a}): \quad \Pr_{y \sim \text{Unif}(Q_t)}(y \in B_j) \geq \alpha.$$

This means that in each iteration, when the population is at the current level, there exists a constant fraction of the population in $A_j \times B_j$. Our current analysis works when assuming (A | G2a) holds. The other case where (A | G2a) does not hold is left as an open problem.

Lemma 6.5.2. *Given two populations P and Q in PDCoEA with population size $\lambda \in$ poly(n) and a sufficiently small constant mutation rate $\chi \in (0,1)$ on* DIAGONAL *with problem size n, assume (A | G2a) holds for a constant $\gamma_0 \geq \left(-1 + \sqrt{4\alpha(1+\delta)+1}\right)/2\alpha$ where $\delta \in (0,\alpha)$. For all $\gamma \in (0,\gamma_0)$, if $|(P \times Q) \cap (U_{j+1} \times V_{j+1})| \geq \gamma\lambda^2$, then for a sample $(x,y) \sim \mathcal{D}(P,Q)$, there exists a constant $\delta_0 > 0$ such that for all current levels $j \in [n/2, n]$, $\Pr(x \in U_{j+1}) \Pr(y \in V_{j+1}) \geq (1+\delta_0)\gamma$.*

Lemma 6.5.2 shows that we obtain multiplicative growth on the next level if the mutation rate is relatively low.

6.5.5 Ensuring Condition (G2b)

(G2b) in Theorem 6.3.1 keeps track of the multiplicative growth on the current level. Before showing (G2b) condition, we need to show that if the population starts at level $n/2$ (as stated in Theorem 6.5.1, with the assumption (A | G2b) holds, we can have the multiplicative growth in the current level. (A | G2b) means that there exists a certain fraction of the population in previous pixels for $x \in A_{<j}$ and $y \in B_{<j} \cup B_j$. To proceed with this, we show Lemma 6.5.3.

Lemma 6.5.3. *Let $\mathcal{D}$ be the operator associated to Algorithms 15 (PDCoEA) on* Diagonal. *Let P and Q be populations of PDCoEA at current level $j \in [n/2, n]$ of* Diagonal *with respect to the partitions U, V and $\gamma_0 \in (0, 1)$. There exists a constant $\delta_0 > 0$ and sufficiently small mutation rate χ, such that if for some constant $\delta > 0$, (A | G2b):*

$$\Pr_{x \sim \mathrm{Unif}(P)}(x \in A_{<j}) \left(\Pr_{y \sim \mathrm{Unif}(Q)}(y \in B_{<j} \cup B_j)^2 + 1 \right) \geq \frac{1 - \gamma_0^2 + \delta}{\gamma_0} \text{ holds,}$$

then for $(x, y) \sim \mathcal{D}(P, Q)$, $\Pr(x \in U_j) \Pr(y \in V_j) \geq (1 + \delta_0)\gamma_0$.

Next, we show that the condition (A | G2b) holds with high probability.

Lemma 6.5.4. *Suppose PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$ with the runtime of Algorithm 15 finding the optimum $T \in \mathbb{N}$ and $(x_t, y_t) \sim \mathcal{D}(P_t, Q_t)$ at iteration $t \leq T$. With the same setting of Lemma 6.5.3, there exists $1 \geq \gamma_0 \geq \sqrt{(1 + \delta)/(\chi e^{-\chi}/7 + 1)}$ such that (A | G2b) holds for some constant $\delta \in (0, \chi e^{-\chi}/7)$ with probability $1 - e^{-\Omega(n)}$.*

Lemma 6.5.4 shows that the condition (A | G2b) holds with overwhelming high probability. Then, we can now divide the computation of G2b into two cases: either (A | G2b)

holds or not. If it holds, then G2b is easy. If not, then we can use Lemma 6.5.4 to show that this happens with extremely small probability.

Lemma 6.5.5. *Given two populations from PDCoEA, P and Q with population size $c\log n \leq \lambda \in poly(n)$ where $c > 0$ is some constant and a constant mutation rate χ on* DIAGONAL *with problem size n, assume constant $\gamma_0 \geq \sqrt{(1+\delta)/\left(\chi e^{-\chi}/7+1\right)}$ with constant $\delta \in (0, \chi e^{-\chi}/7)$ and sufficiently large n. Suppose PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$ with the runtime of Algorithm 15 finding the optimum $T \in \mathbb{N}$ and $(x_t, y_t) \sim \mathcal{D}(P_t, Q_t)$ at iteration $t \leq T$. There exists a constant $\delta > 0$ such that for all current levels $j \in [n/2, n]$, $\Pr(x_t \in U_j)\Pr(y_t \in V_j) \geq (1+\delta)\gamma_0$.*

Finally, Lemma 6.5.5 states that if the mutation rate is set low, γ_0 large enough and furthermore the population is initialised at level $A_{n/2} \times B_{n/2}$, then we can obtain the multiplicative growth in our current level. Next, we conclude everything in the level-based argument using Theorem 6.3.1 with Lemma 6.5.1, 6.5.2 and 6.5.5.

Theorem 6.5.1. *Assume the condition (A | G2a) holds with $\gamma_0 \geq \left(\sqrt{4\alpha(1+\delta)+1}-1\right)/2\alpha$ where $\delta \in (0, \alpha)$ and $\alpha > 0$ is some constant with respect to χ. Assume that for a sufficiently large constant c, it holds $c\log n \leq \lambda \in poly(n)$ and Algorithm 15 initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$. If χ is a sufficiently low constant, then there exists a constant (w.r.t n and λ) $d > 0$ such that the runtime of Algorithm 15 is at most $rd\lambda n\left(\lambda^2 + n^2\right)$ with probability at least $1 - 1/r$.*

Proof. To employ Theorem 6.3.1, we choose the partition such that

$$A_0 \times B_0 := \mathcal{X} \times \mathcal{Y}$$

Moreover, for $j \in \{n/2, \ldots, n\}$, recall that

$$A_j \times B_j := \{x \in \{0,1\}^n \mid |x|_1 = j\} \times \{y \in \{0,1\}^n \mid |y|_1 = j\}.$$

Given Lemma 6.5.2 and Lemma 6.5.5, we only need to show (G1) and (G3) now to apply Theorem 6.3.1. For (G1), by using Lemma 6.5.1, notice that we choose constant $\gamma_0 > (e^{2\chi}/2)^{1/3}$, for $j \geq 1$,

$$\begin{aligned}\Pr_{(x,y)\sim\mathcal{D}(P,Q)}(x \in A_{j+1}) \Pr_{(x,y)\sim\mathcal{D}(P,Q)}(y \in B_{j+1}) &\geq \left(\gamma_0 \frac{\chi}{n}(1-\frac{\chi}{n})^{n-1}\right)^2 \\ &\geq \frac{\gamma_0^2 \chi^2 e^{-2\chi}\,(1-o(1))}{n^2} := z_j.\end{aligned}$$

For (G3), since we choose $\lambda \in \text{poly}(n)$ and $\lambda = \Omega\left(\log(n/\min_{i\in[n]}\{z_i\})\right)$, we can automatically satisfy (G3). Thus, the runtime is bounded by

$$\Pr\left(T \leq c'' r\lambda\left(\lambda^2 n + e^{2\chi} + \sum_{j=1}^{n/2} 1/z_i\right)\right) \leq \frac{1}{r}(1+o(1)).$$

By substituting the values gives

$$\Pr\left(T \leq c'' r\lambda\left(\lambda^2 n + e^{2\chi} + \frac{n^3}{2\gamma_0^2\chi^2 e^{-2\chi}\,(1-o(1))}\right)\right) \leq \frac{1}{r}(1+o(1)).$$

This is equivalent to the following:

$$\Pr\left(T \leq r \cdot O(\lambda n(\lambda^2+n^2)\right) \leq \frac{1}{r}(1+o(1)).$$

□

Theorem 6.5.1 shows that if we set a low constant mutation rate and large population size, then PDCoEA can find the optimum of DIAGONAL in polynomial runtime with high probability. To the best of our knowledge, although various empirical studies demonstrate the existence of a "coevolutionary arms race" (Ficici and Pollack, 1998b; Gomes et al., 2014a; Hillis, 1990; Popovici et al., 2012), including recent applications of coevolutionary algorithms (Hemberg et al., 2021; Xue et al., 2023), this is the first provable coevolutionary "arms race" for solving binary maximin optimisation problems. This result not only demonstrates the promising potential of coevolution with pairwise dominance but also rigorously proves the necessity of large population sizes and low mutation rates to maintain a "coevolutionary arms race" on binary maximin optimisation problems.

In addressing the technical constraints of Theorem 6.5.1, it is important to note that generalising the result without the initialisation constraint is challenging. Proving Theorem 6.5.1 with random initialisation introduces a random walk phase, and it remains unclear whether existing tools can effectively address genetic drift toward the diagonal. The primary objective of this analysis is to rigorously demonstrate the "coevolutionary arms race," and for this purpose, the current Theorem 6.5.1 sufficiently illustrates this point. We consider Theorem 6.5.1 as a foundational step for future advancements.

6.6 Experiments

To complement our asymptotic results with data for concrete problem sizes, we conduct the following experiments.

Settings: We conduct experiments with all CoEAs on the DIAGONAL problem with three different problem sizes: $n = 100$, 500, and 1000. Firstly, we conduct experiments of all CoEAs on DIAGONAL with $n = 100$, generating heatmaps for the empirical mean of runtimes under different configurations. Secondly, we explore the best possible configurations of χ and λ for PDCoEA with problem sizes $n = 100, 500$, and 1000, providing detailed statistics, including tables for the empirical mean of runtimes, p-values from Wilcoxon rank-sum tests for runtimes, and boxplots for runtime distribution with respect to low mutation rates $\chi \in [0.1, 0.4]$. Finally, we pick $\chi = 0.3$ (The following experiments, i.e., grid search for mutation rates from 0.1 to 0.7 and population size from $0.2n$ to n, in Figure 6.2 show that $\chi = 0.3$ seems a reasonable mutation rate to start with. For example, $\chi = 0.5$ and $\chi = 0.7$ are too high; $\chi = 0.1$ is too low.) and $\lambda = c \log n$ where $c = 10, 20, 30, 40, 50$, and 60 and conduct the experiments under problem sizes $n = 100$-1000. The budget for each run is set to 10^8 function evaluations for $n = 100$ and 10^9 function evaluations for $n = 500$ and 1000. For each configuration, we conduct 100 independent runs.

Results: We defer other empirical results to the appendix and present part of our findings

here. As shown in Figure 6.2, with respect to different problem sizes, once the population size reaches certain thresholds (i.e., $\lambda \geq 0.4n$ for $n = 100$ and $\lambda \geq 0.2n$ for $n = 500, 1000$) under a low mutation rate $\chi \in [0.1, 0.4]$. Although PDCoEA can solve DIAGONAL under certain configurations, a high mutation rate above $\chi = 0.5$ results in the inefficiency of PDCoEA. The data (in the appendix) shows that the best empirical mean of runtime, for $n = 100, 500, 1000$, is approximately 0.12×10^6, 1.05×10^6 and 2.71×10^6, respectively, under a certain low mutation rate $\chi = 0.3$. Figure 6.3 suggests that although our theoretical bound is correct, it might not be tight enough in terms of asymptotic order. While our bounds are sufficient to show a distinct separation between CoEAs with a large population and those with a small population, our theoretical bounds can be improved.

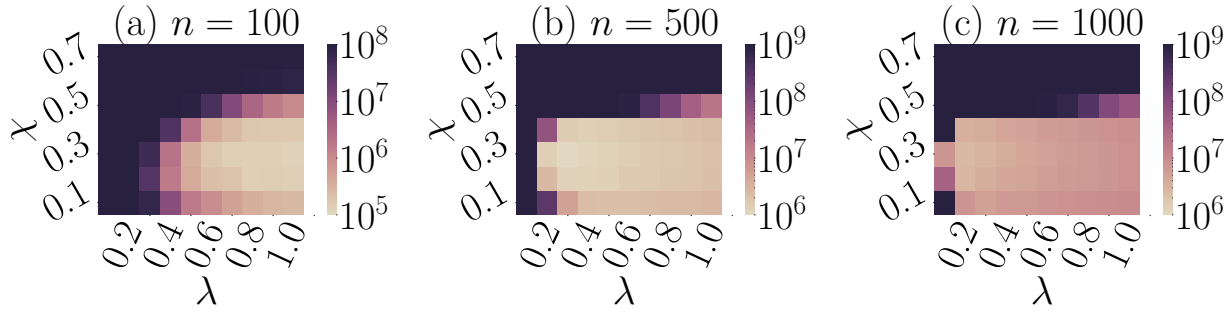


Figure 6.2: Heatmaps for the mean runtime of PDCoEA on DIAGONAL against different mutation rates and each pixel in the plot represented the empirical mean of runtime among 100 independent runs with respect to different mutation rates and population sizes. The mutation rate χ in the vertical axis ranges from 0.1 to 0.7, and the population size λ in the horizontal axis ranges from $0.1n$ to n.

6.7 Conclusion

Discrete maximin optimisation is challenging. This chapter presents the first runtime analysis of CoEAs with pairwise dominance on non-trivial binary two-player zero-sum games. We show that for the DIAGONAL game, with overwhelmingly high probability, RLS-PD

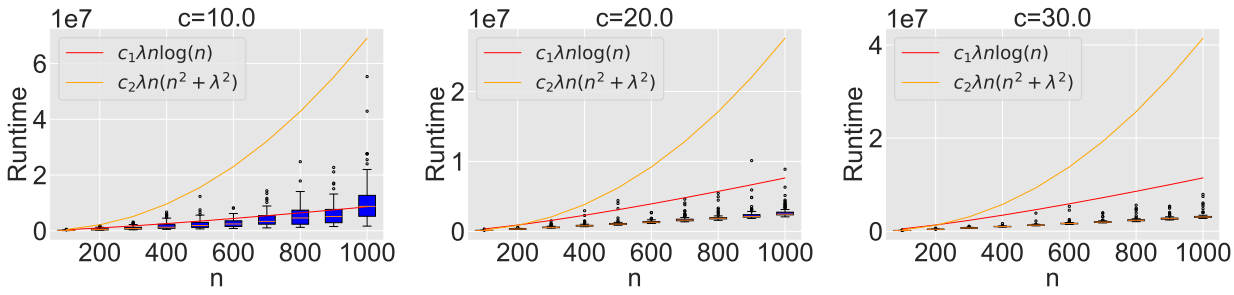


Figure 6.3: Boxplots for runtime of PDCoEA ($\chi = 0.3$) on DIAGONAL with respect to different c and problem size n.

and (1+1)-CoEA require exponential time to find the optimum due to negative drift, highlighting the inefficiency of CoEAs without sufficiently large population sizes in flat payoff landscapes. Note also that the low success probability implies that even strategies for restarting the algorithm will be unsuccessful (Case and Lehre, 2020c; Friedrich et al., 2018). However, PDCoEA with a low mutation rate and large population size $\lambda = \Omega(\log n)$ can efficiently find the optimum with high probability.

This chapter presents a clear separation among various CoEAs on binary two-player zero-sum games, emphasising the need for large populations, as opposed to previous studies on BILINEAR (Fajardo et al., 2023) (while, Fajardo et al. (2023) only considered a simple special case when the optimum at $n/2$). It demonstrates the feasibility of runtime analysis for population-based competitive CoEAs, suggesting that these tools can broaden the scope of future analyses. Practically, it shows that single-pair CoEAs like RLS-PD and (1+1)-CoEA may be unsuitable for such problems, recommending instead CoEAs with large populations and low mutation rates for challenging, flat payoff landscapes.

This work marks a significant first step in the rigorous runtime analysis of population-based CoEAs on DIAGONAL and introduces a variant of the level-based theorem for co-evolution, which assumes specific initialisation of the population. While our analysis is focused on a specific class of sparse binary games, we believe the level-based theorem has

broader applicability to general coevolutionary processes. Extending this tool to more general partitions beyond the Cartesian product of search spaces $\mathcal{X}$ and $\mathcal{Y}$ would be valuable. Although our empirical results imply that the current bound may not be tight, we conjecture that the runtime of PDCoEA could be $O(\lambda n \log n)$ under certain conditions. Future research should explore a wider range of binary games, develop additional coevolutionary algorithms, and refine the runtime bounds for PDCoEA on DIAGONAL.

Chapter Seven

Conclusion

7.1 Summary

Learning in games presents a fundamental challenge in machine learning and artificial intelligence, with numerous applications from board games to robust optimisation and adversarial training. These scenarios often involve strategic interactions among agents, particularly in adversarial optimisation, where success depends on competing against adaptive opponents. Coevolutionary algorithms provide a promising framework for addressing such problems, particularly in black-box query oracle settings that lack gradient information. Despite their practical success, theoretical analysis, of competitive CoEAs remain underexplored. Previous work has focused primarily on cooperative settings, while competitive dynamics have been less analysed rigorously from a runtime perspective. This thesis addresses this gap by investigating the runtime and regret properties of competitive CoEAs on binary games, providing new theoretical tools and results that clarify when and why these algorithms succeed in solving adversarial optimisation problems.

To address the first research question in Chapter 1, whether we can develop suitable theoretical tools to analyse the runtime of competitive CoEAs, we introduced several novel drift theorems. Standard drift analysis fails in competitive settings such as the bilinear

problem, where the dynamics exhibit negative drift near the optimum. In Chapter 3, we proposed the variance drift theorem with a tail bound version, which provides meaningful runtime guarantees even when the expected progress is negative under certain conditions. This new tool allows us to rigorously analyse the runtime behaviour of competitive CoEAs in settings where traditional methods break down. Our results demonstrate that, despite the adversarial nature of the problem and the challenges posed by cyclic behaviour, it is still possible to establish polynomial runtime bounds under appropriate parameter settings and interaction models. Moreover, these theorems can be generally applied to different problems, including bandit learning, 2-SAT and graph colouring problems. This contributes a foundational step toward a general theory of runtime analysis for competitive coevolution and regret analysis of bandit learning algorithms.

To answer the second research question in Chapter 1, we examined whether any algorithm can consistently outperform others in adversarial optimisation by introducing a black-box model for adversarial optimisation, particularly when the solution concept is the Nash Equilibrium (NE). Our investigation builds upon the No Free Lunch (NFL) Theorems, which state that all black-box algorithms on single-objective optimisation perform equally when the performance is averaged over all problems. However, adversarial optimisation introduces a more complicated structure through solution concepts like NE. In Chapter 4, we formalised and proved an Adversarial No Free Lunch Theorem, demonstrating that even when restricting attention to the NE as the desired solution concept, no general algorithm can outperform others across all adversarial problem instances. This result extends the classical NFL perspective to adversarial domains, revealing intrinsic computational hardness when using problem-independent, possibly randomised, search heuristics. Our findings clarify that while specific algorithms may excel on particular problems, there exists no universally superior method in the black-box setting for finding Nash Equilibria, underlining the importance of tailoring algorithms to specific problem structure in adversarial optimisation.

To address the third research question in Chapter 1, we investigated whether competitive coevolutionary algorithms (CoEAs) can be provably more efficient in finding the Nash Equilibrium than traditional evolutionary algorithms (EAs) using the fixed-pairing approach in certain classes of adversarial optimisation problems (i.e., DIAGONAL and BILINEAR). While empirical evidence has long suggested the potential of CoEAs in domains like sorting networks and spatial games, rigorous runtime analysis has been lacking. In Chapter 5, we provided a rigorous comparison between a standard EA and its coevolutionary variant on an adversarial benchmark, namely DIAGONAL. Our analysis demonstrated that a traditional EA using fixed pairings to evaluate individuals requires at least exponential time $e^{\Omega(n)}$ to find the Nash Equilibrium with overwhelmingly high probability. In contrast, the $(1, \lambda)$-CoEA, by continuously co-adapting both populations, was shown to reach the equilibrium in expected polynomial time $24(2-\delta)\lambda n$. This separation result provides the first theoretical evidence that competitive coevolution can yield a significant computational advantage in certain adversarial settings, thereby establishing the necessity of CoEAs in such cases.

Following the result of Research Question 3, which establishes the necessity of competitive coevolution, Research Question 4 turns to examine the performance of specific CoEAs, including $(1, \lambda)$-CoEA and PDCoEA. Specifically, we analysed how key design choices, such as population size and mutation rate, affect the runtime of CoEAs in adversarial optimisation. In Chapter 6, our analysis showed that not all CoEAs are equally effective: small populations or improperly tuned mutation rates can lead to exponential runtime, while sufficiently large populations and appropriately scaled mutation rates can ensure polynomial-runtime performance $O(\lambda n\,(\lambda^2 + n^2))$. These findings highlight that the runtime of coevolutionary algorithms depends strongly on the selection of parameters, providing practical guidelines and theoretical understanding for designing effective CoEAs in adversarial environments.

7.2 Future Work

This thesis opens several promising avenues for future research. On the theoretical side, tighter runtime and tail-bound estimates are needed, particularly for CoEAs like (1, λ)-CoEA and PDCoEA on benchmarks such as Diagonal. The development of stronger drift theorems and extensions of level-based analysis to more general coevolutionary dynamics and partition structures would enhance our analytical toolkit. It is also important to explore black-box complexity in more general settings, such as non-zero-sum games, multiple Nash equilibria, and alternative solution concepts beyond NE. On the practical side, future work should focus on designing more stable and efficient CoEAs through principled choices of population size and mutation rates, potentially incorporating techniques like self-adaptation or multi-objective optimisation. Extending runtime and regret analysis to broader classes of coevolutionary and learning algorithms remains a largely unexplored but critical direction. There is also significant potential to apply concentration bounds from drift analysis to reinforcement learning and bandit problems, particularly for non-stationary or stochastic settings.

Appendix One

A.1 Basic Probability Theory

A.1.1 Sigma-algebra and Measure

Definition A.1.1 (Algebra and σ-algebras (Adams, 2021))**.** *Let Ω be a set and $\mathcal{A} \subset \mathcal{P}(\Omega)$ be a collection of subsets of Ω.*

(1) $\mathcal{A}$ is an algebra if $\Omega \in \mathcal{A}$; $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$; and $A, B \in \mathcal{A} \Rightarrow A \cup B \in \mathcal{A}$.

(2) $\mathcal{A}$ is an σ-algebra if $\mathcal{A}$ is an algebra and $\cup_{n=1}^{\infty} A_n \in \mathcal{A}$ for all $(A_n)_{n\geq 1} \in \mathcal{A}$.

Definition A.1.2. *(Measure space (Cohn, 2013)) A measure space is a triple $(\Omega, \mathcal{F}, \mu)$ where Ω is a set, $\mathcal{F}$ is a σ-algebra on Ω, $\mu : \mathcal{F} \to [0, +\infty]$ is a measure. We define μ is a measure if μ is a pre-measure on σ-algebra $\mathcal{F}$.*

Definition A.1.3 (Type of measure space (Chleboun, 2021))**.** *Let $(\Omega, \mathcal{F}, \mu)$ be a measure space,*

(1) μ is called finite if $\mu(\Omega) < \infty$;

(2) μ is called σ-finite if there exists a sequence of sets $(E_n)_{n\geq 1}$ in $\mathcal{F}$ such that $\mu(E_n) < \infty$ for each $n \in \mathbb{N}$ and $\cup_{n=1}^{\infty} E_n = \Omega$.

(3) μ is called a probability measure if $\mu(\Omega) = 1$ and $(\Omega, \mathcal{F}, \mu)$ is called a probability space, usually we use $(\Omega, \mathcal{F}, \Pr)$ to denote it.

Definition A.1.4 (Distribution of Pr (Cohn, 2013))**.** *Let* Pr *be a probability measure on* $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$*. The distribution is* $F : \mathbb{R} \to [0, +\infty]$*,* $F(x) = \Pr((-\infty, x]))$*.*

A.1.2 Random variables and Filtration

Definition A.1.5 ((Cohn, 2013))**.** *Suppose* $(\Omega, \mathcal{F}, \Pr)$ *is a probability space and* $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ *is a measurable space. A function* $X : \Omega \to \mathbb{R}$ *is a random variable if*

$$\forall G \in \mathcal{G} := \mathcal{B}(\mathbb{R}) \Rightarrow X^{-1}(G) \in \mathcal{F}.$$

In other words, the preimage of every $\mathcal{G}$-measurable set is $\mathcal{F}$-measurable.

Definition A.1.6 (σ-algebra generated by random variables (r.v.s) (Chleboun, 2021))**.** *Suppose* $(\Omega, \mathcal{F}, \Pr)$ *is a probability space and X is a $\mathcal{F}$-measurable r.v.. Then we define the σ-algebra generated by X to be*

$$\sigma(X) = X^{-1}(\mathcal{B}(\mathbb{R})) = \{X^{-1}(A) : A \in \mathcal{B}(\mathbb{R})\} = \sigma(\{X \leq t : t \in \mathbb{R}\}).$$

Moreover, if $(X_i : i \in I)$ is a collection of r.v.s, then the σ-algebra generated by $(X_i : i \in I)$ is

$$\sigma(X_i : i \in I) = \sigma\big(\cup_{i \in I} \sigma(X_i)\big) = \sigma(\{X_i \leq t : i \in I, t \in \mathbb{R}\}).$$

Now we define an important concept which will play a role in some later proof.

Definition A.1.7 (Filtration (Cohn, 2013))**.** *A filtration of σ-algebra on a probability space* $(\Omega, \mathcal{F}, \Pr)$ *is an increasing sequence of σ-algebra* $(\mathcal{F}_n)_{n \geq 0}$ *such that* $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \ldots \subseteq \mathcal{F}_n \subseteq \ldots \mathcal{F}$*. In this case,* $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n \geq 0}, \Pr)$ *is called a filtered space.*

Definition A.1.8 (Adapted Stochastic Processes (Chleboun, 2021))**.**

- *A discrete-time stochastic process is a sequence* $(X_n)_{n\geq 0}$ *of r.v.s. on a probability space* $(\Omega, \mathcal{F}, \Pr)$.

- *A stochastic process is integrable if for each* $n \in \mathbb{N}_{\geq 0}$, *we have* $X_n \in \mathcal{L}(\Omega, \mathcal{F}, \Pr)$ *(i.e.* $\mathbb{E}[X_n] < \infty$ *).*

- *A stochastic process* $(X_n)_{n\geq 0}$ *is adapted to the filtration* $(\mathcal{F}_n)_{n\geq 0}$ *if for each* $n \in \mathbb{N}_{\geq 0}$, *the r.v.* X_n *is* $\mathcal{F}_n$*-measurable.*

Definition A.1.9 (Natural filtration (Cohn, 2013))**.** *The natural filtration,* $(\mathcal{G}_n)_{n\geq 0}$ *associated with the stochastic process* $(X_n)_{n\geq 0}$ *on* $(\Omega, \mathcal{F}, \Pr)$ *is given by* $\mathcal{G}_n = \sigma(X_0, \ldots, X_n) = \sigma(\cup_{i=1}^{n} \sigma(X_i))$ *for* $n \geq 0$.

Example A.1.1. *For instance, if* $X_1, X_2, \ldots$ *is a random walk, then we might have* $\mathcal{F}_n = \sigma(X_0, \ldots, X_n)$. *So we have* $\mathcal{F}_m \subseteq \mathcal{F}_n$ *for* $\forall m \leq n$. *This equivalently means that at time* n, *we "know more" about what the random walk has done. Then we did at time* $m \leq n$.

In this thesis, we mainly work on probability spaces $(\Omega, \mathcal{F}, \Pr)$, where $\mathcal{F}$ is a natural filtration.

A.1.3 Martingales

We provide a formal definition of martingales, super-martingales and sub-martingales.

Definition A.1.10 (Martingales (Williams, 1991))**.** *A sequence of random variables* $X_0, X_1, \ldots$ *is a martingale with respect to the filtration* $\mathcal{F}_0, \mathcal{F}_1, \ldots$ *if for all* $t \geq 0$, *the following conditions hold:*

(1) $(X_t)_{t\geq 0}$ *is adapted to* $(\mathcal{F}_t)_{t\geq 0}$,

(2) $\mathrm{E}(|X_t|) < \infty$

(3) $\mathrm{E}(X_t \mid \mathcal{F}_{t-1}) = X_{t-1}$ *for* $t \geq 1$.

Moreover, a super-martingale with respect to $(\mathcal{F}_t)_{t\geq 0}$ *is defined similarly, except that condition (3) is replaced by for* $t \geq 1$, $\mathrm{E}(X_t \mid \mathcal{F}_{t-1}) \leq X_{t-1}$.

A sub-martingale with respect to $(\mathcal{F}_t)_{t\geq 0}$ *is defined with condition (3) replaced by for* $t \geq 1$, $\mathrm{E}(X_t \mid \mathcal{F}_{t-1}) \geq X_{t-1}$.

Definition A.1.11 (Stopping time (Chleboun, 2021))**.** *Let* $(\Omega, \mathcal{F}, \mathcal{F}_t, \Pr)$ *be a filtered space. A map* $\tau : \Omega \to \mathbb{N}_0 \cup \{\infty\}$ *is called a stopping time, with respect to* $(\mathcal{F}_t)_{n\geq 0}$*, if*

$$\{\tau \leq n\} \in \mathcal{F}_t \quad \textit{for each } t \geq 0.$$

A.2 Useful Inequalities and Lemmas

Theorem A.2.1 (Union Bound (Cohn, 2013))**.** *Let* $(\Omega, \mathcal{F}, \Pr)$ *be a probability space,* $\Pr\left(\cup_{n=1}^{\infty} A_n\right) \leq \sum_{n=1}^{\infty} \Pr(A_n)$ *for all* $A_n \in \mathcal{F}$.

Theorem A.2.2 (Markov's Inequality (Cohn, 2013; Krejca, 2019))**.** *Let* $(\Omega, \mathcal{F}, \Pr)$ *be a probability space, for all* $a > 0$, $\Pr(X \geq a) \leq \mathrm{E}(X)/a$.

Theorem A.2.3 (Chernoff's Bound - Two-Sided)**.** *Given* $X := \sum_{i=1}^{n} X_i$ *where* $X_i \sim Ber(p_i)$ *where* $Ber(p_i)$ *is a Bernoulli random variable with probability* p_i *and all* X_i *are independent. Let* $\mu := \mathrm{E}(X) = \sum_{i=1}^{n} p_i$*. Then, for* $\delta \in (0, 1)$*,*

$$\Pr(|X - \mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}.$$

Theorem A.2.4 (Chernoff's Bound - One-Sided, Right-Tail)**.** *Given* $X := \sum_{i=1}^{n} X_i$ *where* $X_i \sim Ber(p_i)$ *where* $Ber(p_i)$ *is a Bernoulli random variable with probability* p_i *and all* X_i *are independent. Let* $\mu := \mathrm{E}(X) = \sum_{i=1}^{n} p_i$*. Then, for* $\delta \geq 0$*,*

$$\Pr(X \geq (1 + \delta)\mu) \leq e^{-\mu\delta^2/(2+\delta)}.$$

Theorem A.2.5 (Chernoff's Bound - One-sided, Left-Tail)**.** *Given* $X := \sum_{i=1}^{n} X_i$ *where* $X_i \sim Ber(p_i)$ *where* $Ber(p_i)$ *is a Bernoulli random variable with probability* p_i *and all* X_i *are independent. Let* $\mu := \mathrm{E}(X) = \sum_{i=1}^{n} p_i$*. Then, for* $\delta \in (0, 1)$*,*

$$\Pr\left(X \leq (1-\delta)\mu\right) \leq e^{-\mu\delta^2/2}.$$

A.3 Optional Stopping Time Theorems

Theorem A.3.1 (Doob's Optional Stopping Time (Doob, 1971; Williams, 1991))**.** *We consider the following conditions. Let* X_n *be a stochastic process in* $\mathbb{R}$ *and* T *be its stopping time.*

(1) T *is bounded (i.e. for some* $N \in \mathbb{N}$*,* $T(\omega) \leq N, \forall\omega$*);*

(2) X *is bounded (i.e. for some* $K \in \mathbb{R}_{\geq 0}$*,* $|X_t(\omega)| \leq K$ *for every* t *and every* ω *and* T *is a.s. finite);*

(3) $\mathrm{E}(T) < \infty$*, and, for some* $K \in \mathbb{R}_{\geq 0}$*,*

$$|X_t(\omega) - X_{t-1}(\omega)| \leq K \quad \forall(t, \omega).$$

In particular, let X_n *be a super-martingale and* τ, σ *be stopping times. Then* X_τ, X_σ *are integrable and for* $\sigma \leq \tau$*,* $\mathrm{E}(X_\tau \mid \mathcal{F}_\sigma) \leq X_\sigma$ *a.s.. In particular, for* $\sigma = 0, \tau = T$*,* $\mathrm{E}(X_T) \leq \mathrm{E}(X_0)$ *in each of the above situations (1)-(3) holds.*

Let X_n *be a sub-martingale. Then* X_τ, X_σ *are integrable and for* $\sigma \leq \tau$*,* $\mathrm{E}(X_\tau \mid \mathcal{F}_\sigma) \geq X_\sigma$ *a.s.. In particular, for* $\sigma = 0, \tau = T$*,* $\mathrm{E}(X_T) \geq \mathrm{E}(X_0)$ *in each of the above situations (1)-(3) holds.*

Notice that in condition (3), the theorem still assumes a fixed step size. Then, we need

an extended version of the Optional Stopping Theorem (Bhattacharya and Waymire, 2007; Grimmett and Stirzaker, 2001).

Theorem A.3.2 (Extended Doob's Optional Stopping Time)**.** *We consider the following conditions. Let X_t be a stochastic process in $\mathbb{R}$ and T be its stopping time.*

(1) T is bounded (i.e. for some $N \in \mathbb{N}$, $T(\omega) \leq N, \forall \omega$);

(2) X is bounded (i.e. for some $K \in \mathbb{R}_{\geq 0}$, $|X_t(\omega)| \leq K$ for every t and every ω and T is a.s. finite);

(3) $\mathrm{E}(T) < \infty$*, and, for some $K \in \mathbb{R}_{\geq 0}$,*

$$|X_t(\omega) - X_{t-1}(\omega)| \leq K \quad \forall (t, \omega).$$

(4) The stopping time T has a finite expectation, and the conditional expectations of the absolute value of the martingale increments are almost surely bounded. More precisely, $E(T) < \infty$, and there exists a constant $c > 0$ such that

$$\mathrm{E}\big(|X_{t+1} - X_t| \,\big|\, \mathcal{F}_t\big) \leq c$$

almost surely on the event $\{T > t\}$ for all $t \in \mathbb{N}_0$.

Theorem A.3.1 holds in each of the above situations (1)-(4).

A.4 Supplementary Materials of Chapter 3

A.4.1 Pseudo-Code of Algorithms

We consider the pseudocode from (Mitzenmacher and Upfal, 2005).

Algorithm 16 2-SAT Algorithm

1: Start with an arbitrary truth assignment.

2: **for** $t = 0, 1, 2, \ldots, 2mn^2$ until all clauses are satisfied **do**

3: Choose an arbitrary clause that is not satisfied;

4: Choose uniformly at random one of the literals in the clause and switch the value of its variable

5: **if** a valid truth assignment has been found **then** return it.

6: Otherwise, return that the formula is unsatisfied.

We consider the pseudocode from (McDiarmid, 1993).

Algorithm 17 Recolour

1: Start with an arbitrary 2-colouring of the points

2: **while** SEEK returns a monochromatic edge E **do**

3: pick a random point in E and change its colour

We consider the pseudocode from (Hevia Fajardo et al., 2023).

Algorithm 18 RLS-PD: Randomised Local Search with Pairwise Dominance

Require: Maximin-objective function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$

1: Sample $x_1 \sim \text{Unif}(\{0,1\}^n)$

2: Sample $y_1 \sim \text{Unif}(\{0,1\}^n)$

3: **for** $t \in \{1, 2, \ldots\}$ **do**

4: Create $x', y' \in \{0,1\}^n$ by copying x_t and y_t and flipping exactly one bit chosen uniformly at random from either x_t or y_t.

5: **if** $(x', y') \succeq_g (x_t, y_t)$ **then** $(x_{t+1}, y_{t+1}) := (x', y')$

A.4.2 Supplementary material for the analysis of coevolutionary algorithms

We defer some important definitions and lemmas used in the analysis of the RLS-PD algorithm here. Following the definitions used in (Hevia Fajardo et al., 2023), we define

Definition A.4.1. *Let $M(x,y)$ be the Manhattan distance from a search point $(x,y) \in \mathcal{X} \times \mathcal{Y}$ to the optimum, that is, $M(x,y) := |n\beta - |x|_1| + |n\alpha - |y|_1|$. Let $M_t := M(x_t, y_t)$. For all $t \in \mathbb{N}$ we define:*

$$p^+_{x,y} := \Pr\left(M_{t+1} > M_t \mid M_t = M(x,y)\right);$$
$$p^-_{x,y} := \Pr\left(M_{t+1} < M_t \mid M_t = M(x,y)\right).$$

Definition A.4.2. *During our analysis, we divide the search space into four quadrants. We say that a pair of search points (x,y) is in:*

- *the first quadrant if $0 \leq |x|_1 < \beta n \wedge \alpha n \leq |y|_1 \leq n$,*
- *the second quadrant if $\beta n \leq |x|_1 \leq n \wedge \alpha n < |y|_1 \leq n$,*
- *the third quadrant if $\beta n < |x|_1 \leq n \wedge 0 \leq |y|_1 \leq \alpha n$, and*
- *the fourth quadrant if $0 \leq |x|_1 < \beta n \wedge 0 \leq |y|_1 < \alpha n$.*

Lemma A.4.1. *(Hevia Fajardo et al., 2023) Consider RLS-PD on* Bilinear *as in Theorem 3.5.1. Define $T := \inf\{t \mid M_t = 0\}$, then for every generation $t < T$*

$$\mathrm{E}\left(M_t - M_{t+1} - \frac{M_t - (A+B)\sqrt{n}}{2n}; t < T \mid M_t\right) \geq 0.$$

A.4.3 Supplementary material for the analysis of RWAB algorithm

Before showing the main theorem, we need some lemma to proceed.

Lemma A.4.2. *Given $X \sim Geo(p), Y \sim Geo(q)$ where $p, q \in (0,1)$, if $p \geq q$, then Y stochastic dominates X.*

Lemma A.4.2 shows that given two Geometric distributed random variables, the sufficient condition for one stochastic dominating the other is simply with higher "success" probability.

The proof consists of four cases. We denote the accumulated regret for each case by $\mathcal{R}_i$ for $i \in [4]$. Next, following the definitions in (Larcher et al., 2023), we define $\mathcal{R}_i$ for $i \in [4]$ formally.

Definition A.4.3. *We denote the optimal arm by a^*, the action value of S in* CHALLENGE *by S_t and the expected regret at iteration t by Δ. Let $s = \sqrt{T/L}$ where T is the time horizon and L is the number of changes. Then, we define*

- $\mathcal{R}_1$*: "The accumulated regret when starting* CHALLENGE *with correct arm $a^+ = a^*$ until a change occurs or the* CHALLENGE *ends";*
- $\mathcal{R}_2$*: "The accumulated regret when a change occurs during an ongoing* CHALLENGE *until the next change occurs or the end of this ongoing* CHALLENGE *no matter if we have $a^+ = a^*$ or $a^+ \neq a^*$";*
- $\mathcal{R}_3$*: "The accumulated regret when $a^+ \neq a^*$ with no ongoing* CHALLENGE*";*
- $\mathcal{R}_4$*: "The accumulated regret when starting* CHALLENGE *with incorrect arm $a^+ \neq a^*$ until a new change occurs or the end of* CHALLENGE *in which the action value in* CHALLENGE*, denoted by S_t, hits $-s$ to trigger a swap".*

Proof of Sketch for Theorem 3.6.1. Here we provide a sketch of the proof. We divide the analysis into four cases covered by $\mathcal{R}_i$ for $i \in [4]$ (as shown in Figure A.1). Once we obtain the tail bound for these $\mathcal{R}_i$, we can derive the tail bound for total regret $\mathcal{R}$ by using $\mathcal{R} \leq \sum_{i=1}^{4} \mathcal{R}_i$. □

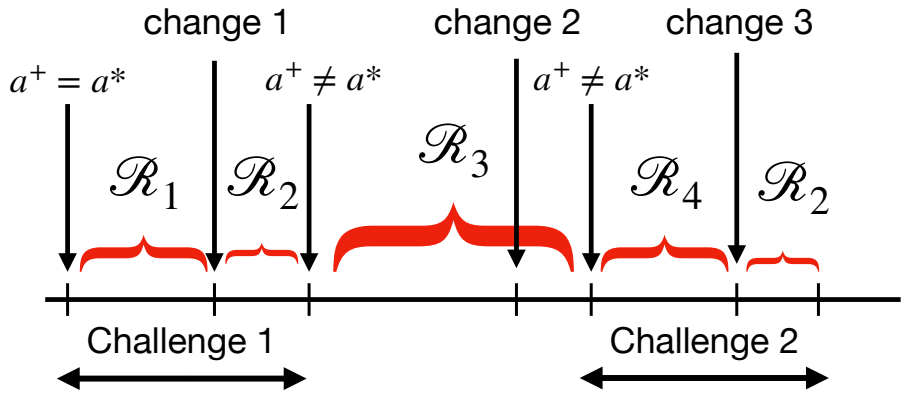


Figure A.1: Classes of accumulated Regret along time horizon.

A.5 Supplementary Materials of Chapter 4

A.5.1 Two Other Solution Concepts

Definition A.5.1 (Maximisation Over All Test Cases (Service and Tauritz, 2008)). *Suppose a set of candidate solutions, $\mathcal{X}$, and a set of test cases, $\mathcal{Y}$. Given an interaction function or payoff function, $g : \mathcal{X} \times \mathcal{Y} \to O$, where O is an ordered set which determines the outcome of candidate solution $\mathcal{X}$ on test $\mathcal{Y}$, the solution concept is defined by:* $S = \{x^* \in \mathcal{X} \mid \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, g(x^*, y) \geq g(x, y)\}$.

Next, we define the maximin solutions are those solution configurations which maximise the minimum outcome over all test cases.

Definition A.5.2 (Maximin (Wolpert and Macready, 2005)). *Suppose a set of candidate solutions, $\mathcal{X}$, and a set of test cases, $\mathcal{Y}$. Given an interaction function or payoff function, $g : \mathcal{X} \times \mathcal{Y} \to O$, where O is an ordered set which determines the outcome of candidate solution $\mathcal{X}$ on test $\mathcal{Y}$, the solution concept is defined by:*

$$S = \{x^* \in \mathcal{X} \mid \forall x \in \mathcal{X}, \min_{y \in \mathcal{Y}} g(x^*, y) \geq \min_{y \in \mathcal{Y}} g(x, y)\}.$$

Notice that in Definition A.5.1, this solution concept considers a candidate solution x with respect to all test cases y. Definition A.5.2 considers a candidate solution with respect to the worst-case scenario. If $\mathcal{Y}$ is compact, then these two solution concepts are essentially the same. Compared with Nash Equilibrium in Definition 2.3.1, these two solution concepts only focus on finding the best candidate solution and the algorithm is not required also to produce an opposing "optimal" solution y^*.

A.5.2 Technical Lemmas for Games with Unique Nash Equilibrium

We adopt the notation in (Maiti et al., 2023).

Definition A.5.3. *Given a two-player zero-sum game on a payoff matrix $P \in \mathbb{R}^{n\times n}$. Let Δ_n denote the n-dimensional probability simplex (i.e. for $p \in \Delta_n, \sum_{i=1}^n p_i = 1$ and $p_i \geq 0$). A pair (p^*, q^*) is a (mixed strategy) Nash Equilibrium of an input matrix $P \in \mathbb{R}^{n\times n}$ if and only if the following holds for all $(p, q) \in \Delta_n \times \Delta_n$:*

$$\langle p, Pq^* \rangle \leq \langle p^*, Pq^* \rangle \leq \langle p^*, Pq \rangle.$$

In this paper, we are only interested in Pure Strategy Nash Equilibrium.

Definition A.5.4. *For any payoff matrix $P \in \mathbb{R}^{n\times n}$, let $V_P^\star := \max_{p\in\Delta_n} \min_{q\in\Delta_n} \langle p, Pq \rangle$. In particular, we say an entry $(i^*, j^*) \in [n] \times [n]$ is a unique pure strategy Nash Equilibrium if $(e_{i^*}, e_{j^*}) \in \Delta_n \times \Delta_n$ is a unique Nash Equilibrium of P where e_{i^*}, e_{j^*} are the unit probability vector.*

We refer to the following lemma from (Shapley et al., 1950).

Lemma A.5.1 (Lemma 1 in (Maiti et al., 2023))**.** *Consider a matrix $A \in \mathbb{R}^{n\times n}$ with a unique Nash equilibrium $(p^\star, q^\star)$. The following conditions hold:*

1. *The support sizes of $p^\star$ and $q^\star$ are equal (i.e., $|supp(p^\star)| = |supp(q^\star)|$).*

2. *$V_A^\star = \langle e_i, Aq^\star \rangle$ for all $i \in supp(p^\star)$, and $V_A^\star > \langle e_i, Aq^\star \rangle$ for all $i \notin supp(p^\star)$.*

3. *$V_A^\star = \langle p^\star, Ae_j \rangle$ for all $j \in supp(q^\star)$, and $V_A^\star < \langle p^\star, Ae_j \rangle$ for all $j \notin supp(q^\star)$.*

Lemma A.5.1 allows us to derive the following result, which is proved in Appendix B.1 in (Maiti et al., 2023).

Lemma A.5.2 (Lemma 2 in (Maiti et al., 2023))**.** *Consider a matrix $P \in \mathbb{R}^{n \times n}$ with a unique Nash equilibrium $(p^\star, q^\star)$. Consider a submatrix M of P with row index set I_X and column index set I_Y. If $\textsc{Supp}(p^\star) \subseteq I_X$ and $\textsc{Supp}(q^\star) \subseteq I_Y$, then $(\hat{p}, \hat{q})$ is the unique Nash equilibrium of M where $(\hat{p})_i = (p^\star)_i$ for all $i \in I_X$ and $(\hat{q})_j = (q^\star)_j$ for all $j \in I_Y$. Moreover, $V_M^\star = V_P^\star$.*

A.5.3 Analytic Tools from Probability Theory and Analysis of Randomised Algorithms

A.5.3.1 Yao's Principle

In this section, we provide the mathematical tool that we use in the rest of the analysis. Yao's Principle (Yao, 1977) can be used to give lower bounds of a class of randomised algorithms.

Theorem A.5.1 (Yao's Principle (Yao, 1977))**.** *Let Π be a problem with a finite set $\mathcal{I}$ of input instances (of a fixed size) permitting a finite set $\mathcal{A}$ of deterministic algorithms. Let p be a probability distribution over $\mathcal{I}$ and q be a probability distribution over $\mathcal{A}$. Then,*

$$\min_{A \in \mathcal{A}} \mathrm{E}(T(I_p, A)) \leq \max_{I \in \mathcal{I}} \mathrm{E}(T(I, A_q))$$

where I_p denotes a random input chosen from $\mathcal{I}$ according to p, A_q a random algorithm chosen from $\mathcal{A}$ according to q and $T(I, A)$ denote the runtime of algorithm A on input I.

A.5.4 Further Explanations for Sub-Problem Class

We provide an example to illustrate the concept of the sub-problem class (as in Definition 4.3.2).

Definition 4.3.2. *Let $\mathcal{G} = (\mathcal{X}, \mathcal{Y}, \mathcal{F}, \text{NASH})$ be a class of two-player zero-sum games, where $O \subseteq \mathbb{R}$ and $\mathcal{F}$ be any subset of the set of payoff functions in these games $g : \mathcal{X} \times \mathcal{Y} \to O$ such that $\mathcal{F}$ is closed under permutation. For any given $(x_1, y_1) \in \mathcal{X} \times \mathcal{Y}$, any function $b_1 : \mathcal{Y} \to O$, and any function $b_2 : \mathcal{X} \to O$, we define a sub-problem class $\mathcal{F}((x_1, y_1), (b_1, b_2))$ with respect to $\mathcal{F}$ as follows: $f \in \mathcal{F}((x_1, y_1), (b_1, b_2))$ if and only if there exists a function $g \in \mathcal{F}$ such that*

(1) $g(x_1, y) = b_1(y)$ for all $y \in \mathcal{Y}$;

(2) $g(x, y_1) = b_2(x)$ for all $x \in \mathcal{X}$;

(3) f is a restriction of g on $(\mathcal{X} \setminus \{x_1\}) \times (\mathcal{Y} \setminus \{y_1\})$.

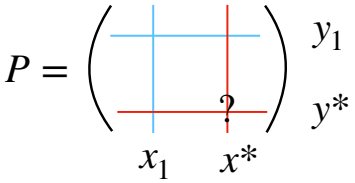


Figure A.2: Example of a sub-problem class. Given a payoff matrix P, the blue entries mean that the actual values are already assigned to $g(x_1, \cdot)$ and $g(\cdot, y_1)$. (x^*, y^*) is the Nash Equilibrium we search for. The sub-problem class here means that the problem consists of different payoff matrices with given blue entries.

Figure A.2 is an example of a sub-problem class. When querying (x_1, y_1), we also check the corresponding row and column to verify whether the unique Nash Equilibrium lies on this blue cross or not. If the answer is negative, then we restrict to the sub-matrix by removing the blue entries in P.

A.5.5 Further Explanations for the role of Corollary 4.3.1 and Lemma 4.3.1 in Theorem 4.3.1

In the main proof of Theorem 4.3.1, we rely on proof by induction: assume the theorem holds for a search space of size N, we want to show it also holds for $N+1$. Corollary 4.3.1 and Lemma 4.3.1 are crucial since they ensure that for every payoff function $g \in \mathcal{F}$, there is a corresponding function g' where the NE (x^*, y^*) is switched to $(\sigma(x^*), \tau(y^*))$ (i.e. σ, τ are two permutations on $\mathcal{X}, \mathcal{Y}$ respectively) and the NE is still unique in the game defined by g'. This means that the inductive hypothesis can be applied to g' due to the closure under permutation. In short, Corollary 4.3.1 and Lemma 4.3.1 establish a property of the problems or payoff functions under consideration that is preserved under permutations. This property helps to ensure that the induction step can be applied successfully.

A.5.6 Further Explanations for bitwise exclusive or in Black-Box Complexity Analysis

We define $\oplus$ as bitwise exclusive or. We introduce bitwise exclusive or here to generate problem instances with the same structure. For example, $\textsc{Diagonal}(x, y)$ is only one possible problem instance in binary voting games and to find its optimum, we only need to design the algorithm querying $(1^n, 1^n)$ to reach its optimum. And its black-box complexity is trivially $\Theta(1)$. Using bitwise exclusive or $\oplus$ (i.e. $\textsc{Diagonal}(u \oplus x, v \oplus y)$ where $(u, v) \in \mathcal{U} \times \mathcal{V}$ is sampled uniformly at random), we can generate a set of problem instances which have the same structure as $\textsc{Diagonal}(x, y)$ and the goal of black-box algorithms is to find (x, y) such that $(u \oplus x, v \oplus y) = (1^n, 1^n)$. (Note that if $(u, v) = (0^n, 0^n)$, then $\textsc{Diagonal}(0^n \oplus x, 0^n \oplus y)$ is reduced to the vanilla $\textsc{Diagonal}(x, y)$.) This bitwise exclusive generator can avoid the trivial black-box complexity mentioned above and reflect the true difficulty of the class Diagonal. We refer to (Doerr et al., 2015; Droste et al., 2006) for more detail about

black-box complexity theory.

A.6 Supplementary Materials of Chapter 5

A.6.1 Proof Idea for Theorem 5.3.1

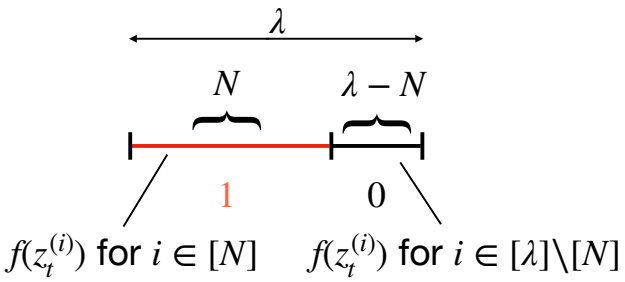


Figure A.3: Sketch of proof idea for Theorem 5.3.1. The offspring are sorted according to fitness. We denote the number of offspring with fitness 1 by N.

A.6.2 Proofs of Technical Lemmas for Section 5.4.1

Proof of Lemma 5.4.1. (1) If $|x^{(1)}| \geq |x^{(2)}|$, then we rewrite the fitness in terms of sum indicator functions:

$$f(x^{(1)}) := g(x^{(1)}, y) = \mathbb{1}_{\{x^{(1)} \geq y\}}$$

Notice that if $|x^{(2)}| \geq |y|$, then $|x^{(1)}| \geq |x^{(2)}| \geq |y|$. Then we have the event inclusion:

$$\{|x^{(2)}| \geq |y|\} \subseteq \{|x^{(1)}| \geq |y|\}$$

This implies that

$$\mathbb{1}_{\{|x^{(2)}| \geq |y|\}} \leq \mathbb{1}_{\{|x^{(1)}| \geq |y|\}}$$

Then, we have

$$f(x^{(2)}) = \mathbb{1}_{\{|x^{(2)}| \geq |y|\}} \leq \mathbb{1}_{\{x^{(1)} \geq y\}} = f(x^{(1)}).$$

(2) If $|y^{(1)}| \geq |y^{(2)}|$, then we rewrite the fitness in terms of sum indicator functions:

$$h(y^{(1)}) := g(x, y^{(1)}) = \mathbb{1}_{\{x \geq y^{(1)}\}}$$

Notice that if $|x| \geq |y^{(1)}|$, then $|x| \geq |y^{(1)}| \geq |y^{(2)}|$. Then we have the event inclusion:

$$\{|x| \geq |y^{(1)}|\} \subseteq \{|x| \geq |y^{(2)}|\}$$

This implies that

$$\mathbb{1}_{\{|x| \geq |y^{(1)}|\}} \leq \mathbb{1}_{\{|x| \geq |y^{(2)}|\}}$$

Then, we have

$$h(y^{(1)}) = \mathbb{1}_{\{|x| \geq |y^{(1)}|\}} \geq \mathbb{1}_{\{|x| \geq |y^{(2)}|\}} = h(y^{(2)}).$$

□

A.6.3 Proofs of Technical Lemmas for Section 5.4.2

Proof of Lemma 5.4.2. In odd iterations $t = 1, 3, \ldots$, Algorithm 13 updates y and in even iterations $t = 2, 4, \ldots$, Algorithm 13 updates x.

If we have $Y_{2t} > X_{2t}$ in iteration $2t$, then the next search point does not cross the diagonal if $Y_{2t+1} > X_{2t+1}$ or $X_{2t+2} \geq Y_{2t+2}$ in one cycle. And if $Y_{2t+1} + c \leq X_{2t+1}$ or $X_{2t+2} + c \leq Y_{2t+2}$, then the search point lies outside the c-tube during the consecutive steps. Thus, a successful cycle in Definition 5.4.3 breaks. Note that a successful cycle in Definition 5.4.3 breaks is equivalent to the event that either the search point escapes from the c-tube or does not cross the diagonal during the cycle. Then, we use the union bound to give an upper bound of the probability that a cycle fails to be successful. Before that, we provide a more precise

union bound as follows. Given events A, B, E,

$$\begin{aligned}\Pr(A \cup B \mid E) &= \Pr(A \mid E) + \Pr(B \setminus A \mid E) \\ &\leq \Pr(A \mid E) + \Pr(B \mid E, A^c) \Pr(A^c \mid E) \\ &\leq \Pr(A \mid E) + \Pr(B \mid E, A^c) \end{aligned} \tag{A.1}$$

Next, we denote the following events.

$$\begin{aligned} E_0 &:= \{Y_{2t} > X_{2t} \cap D_{2t} < c\} \\ E_1 &:= \{X_{2t+1} \geq Y_{2t+1}\} \\ E_2 &:= \{Y_{2t+2} > X_{2t+2}\} \\ F_1 &:= \{D_{2t+1} \geq c\} \\ F_2 &:= \{D_{2t+2} \geq c\} \end{aligned}$$

Note that we have a successful cycle in iteration $2t$ if and only if $E_1 \cap E_2 \cap F_1^c \cap F_2^c$ holds.

$$\begin{aligned} &\Pr(\text{A cycle fails to be successful at iteration } 2t) \\ =&\Pr(E_1^c \cup E_2^c \cup F_1 \cup F_2 \mid E_0) \end{aligned}$$

Using the union bound gives

$$\leq \Pr(E_1^c \cup E_2^c \mid E_0) + \Pr(F_1 \cup F_2 \mid E_0)$$

Using Eq (A.1) gives

$$\begin{aligned} \leq &\Pr(E_1^c \mid E_0) + \Pr(E_2^c \mid E_0, E_1) \\ &+ \Pr(F_1 \mid E_0) + \Pr(F_2 \mid E_0, F_1^c) \end{aligned}$$

Using the conditions $(1) - (3)$ gives

$$\leq 2p_c + 2p_e \tag{A.2}$$

Next, we consider τ consecutive cycles. We define the failure event F_t as the algorithm has an unsuccessful cycle in iteration t. By using the union bound and Eq (A.2), we have

$$\begin{aligned} &\Pr(\exists t \in [\tau] \text{ s.t. } F_t \text{ occurs during } \tau \text{ consecutive cycles}) \\ \leq &2\tau(p_c + p_e) \end{aligned}$$

So, we can derive that

$$\begin{aligned}&\Pr\left(\text{The algorithm has } \tau \text{ consecutive successful cycles}\right)\\ \geq& 1-2\tau(p_c+p_e).\end{aligned}$$

A similar analysis holds for the case that (X_{2t+1}, Y_{2t+1}) in iteration $2t+1$.

□

A.6.4 Proofs of Technical Lemmas for Section 5.4.3

Proof of Lemma 5.4.4. If $\chi = O(1)$, then

$$\Pr\left(\text{No offspring of } x_t \text{ is identical to } x_t\right) = \left(1-(1-\frac{\chi}{n})^n\right)^\lambda$$

Here, we use $(1-\frac{\chi}{n})^n \geq e^{-\chi}(1-\frac{\chi^2}{n})$ and taking n large enough such that $(1-\frac{\chi^2}{n}) \geq \frac{1}{2}$ gives

$$\leq (1-\frac{e^{-\chi}}{2})^\lambda$$

Note that $1-\frac{e^{-\chi}}{2}$ is a constant in $(0,1)$ so that we obtain the exponentially small tail in λ.

$$\leq e^{-\Omega(\lambda)}.$$

Then there exists $k, \ell > 0$ s.t. $x_t^k = x_t$ and $y_t^\ell = y_t$ w.h.p. respectively.

Moreover, note that $\Pr(X_t^{(1)} > X_t^{(\lambda)}) = 1 - \Pr(X_t^{(1)} \leq X_t^{(\lambda)}) = 1 - \Pr(X_t^{(1)} = X_t^{(\lambda)})$. Then we can bound the following first (denote the number of 1-bits of original x_t by X_t):

$$\begin{aligned}\Pr\left(X_t^{(i)} = X_t\right) = &\sum_{k=0}^{\min\{n-X_t, X_t\}} \binom{n-X_t}{k}(\frac{\chi}{n})^k\\ &\times \binom{X_t}{k}(\frac{\chi}{n})^k(1-\frac{\chi}{n})^{n-2k}\end{aligned}$$

Using $\binom{n_1}{k_1}\binom{n_2}{k_2} \leq \binom{n_1+n_2}{k_1+k_2}$ gives

$$\leq \sum_{k=0}^{\min\{n-X_t, X_t\}} \binom{n}{2k}(\frac{\chi}{n})^{2k} \cdot (1-\frac{\chi}{n})^{n-2k}$$

Note that $\min\{n - X_t, X_t\} \leq \frac{n}{2}$

$$\leq \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2k} (\frac{\chi}{n})^{2k} \cdot (1 - \frac{\chi}{n})^{n-2k}$$

Consider a binomial random variable $Z \sim Bin(n, \frac{\chi}{n})$ and we consider the probability that Z is even. Then, we have

$$= \Pr(Z \text{ is even})$$

Using Lemma 5.4.3 with $p = \frac{\chi}{n}$ gives

$$\begin{aligned} &= \frac{1}{2} + \frac{1}{2} \cdot (1 - \frac{2\chi}{n})^n \\ &\leq \frac{1}{2} + \frac{1}{2e^{2\chi}} \end{aligned}$$

Since $\chi > 0$ is some constant, so $\frac{1}{2} + \frac{1}{2e^{2\chi}} = c < 1$ is also constant. Given that there exists $k > 0$ s.t. $x_t^{(k)} = x_t$ w.h.p., we have

$$\begin{aligned} \Pr(X_t^{(1)} = X_t^{(\lambda)}) &\leq \Pr(\forall j \in [\lambda], X_t^{(j)} = X_t) \\ &\quad \times (1 - e^{-\Omega(\lambda)}) + e^{-\Omega(\lambda)}) \cdot 1 \\ &\leq c^{\lambda}(1 - e^{-\Omega(\lambda)}) + e^{-\Omega(\lambda)} \\ &= e^{-\Omega(\lambda)}. \end{aligned}$$

Then, $X_t^{(1)} > X_t^{(\lambda)}$ w.h.p and so is $Y_t^{(1)} > Y_t^{(\lambda)}$. Then we have λ offspring cross the diagonal w.h.p if $X_t^{(\lambda)} \leq Y_t^{(1)}$.

□

Proof of Lemma 5.4.6. We apply Lemma 5.4.5 and Markov's inequality for the concentration. For any $s \geq 0$ and $\lambda \geq 1$,

$$\Pr(U \geq s) = \Pr\left(e^{\eta U} \geq e^{\eta s}\right) \leq E[e^{\eta U}]e^{-\eta s}$$

Taking $E[e^{\eta U}] \leq \exp\left(\chi(e^{\eta}-1)\right)$ and $\eta := \ln\ln\lambda$ gives

$$\begin{aligned} &\leq e^{\chi(\ln\lambda - 1)} e^{-s\ln\ln\lambda} \\ &\leq e^{-\chi}\lambda^{\chi} e^{-s\ln\ln\lambda} \end{aligned}$$

For the second part, we notice that

$$\begin{aligned} \Pr\left(U \geq s\right) = \Pr\left(e^{\eta U} \geq e^{\eta s}\right) &\leq E[e^{\eta U}]e^{-\eta s} \\ &\leq e^{-\chi} e^{\chi e^{\eta} - s\eta} \end{aligned}$$

Note that $h(\eta) := \chi e^{\eta} - s\eta$ attains its minimum at $\eta = \ln(\frac{s}{\chi})$ by differentiation. So setting $\eta := \ln(\frac{s}{\chi})$ gives

$$\leq e^{-\chi} e^{-s\left(\ln(\frac{s}{\chi})-1\right)}$$

$s \geq e^2\chi$ gives $\ln(\frac{s}{\chi}) - 1 \geq 1$. Then, we have

$$\leq e^{-\chi} e^{-s}.$$

□

Proof of Lemma 5.4.7. Let us divide the analysis into two cases. Assume t is even and $Y_t > X_t$ first. The only way the search point can cross the diagonal is to cross horizontally. Assume now we are in iteration t such that we mutate and select the new X_{t+1}. By using Lemma 5.4.4, we conclude that if t is even, then $\Pr\left(\exists k \in [\lambda], x_t^k = x_t\right) \geq 1 - e^{-\Omega(\lambda)}$ and $\Pr\left(\max_{i\in[\lambda]} X_t^i > \min_{i\in[\lambda]} X_t^i\right) \geq 1 - e^{-\Omega(\lambda)}$. Then, we first compute the probability that E does not occur under the case that we have both $F_1 := \{\exists k \in [\lambda], x_t^k = x_t\}$ and $F_2 := \{\max_{i\in[\lambda]} X_t^i > \min_{i\in[\lambda]} X_t^i\}$ hold.

$$\begin{aligned} \Pr\left(E_t^c \mid F_1 \cap F_2\right) &= \Pr\left(X_t^{(1)} < Y_t\right) \\ &= \Pr\left(\max_{i\in[\lambda]} X_t^i < Y_t\right). \end{aligned}$$

We define for $i \in [\lambda]$ for x_t-bitstring, a random variable $W_{i,-}$ to be the number of 1-bits that the mutation operator flips into 0-bits and another the random variable $W_{i,+}$ to be the number of 0-bits that the mutation operator flips into 1-bits. Then, we define the change in 1-bits $W_i = W_{i,+} - W_{i,-}$ is subject to the distribution $\mathrm{Bin}(n - X_t, \frac{\chi}{n}) - \mathrm{Bin}(X_t, \frac{\chi}{n})$. After the mutation in Algorithm 13, we have λ offspring at iteration $t+1$: $X_t^i = X_t + W_i$. So we have

$$\begin{aligned}
\Pr\left(E_t^c \mid F_1 \cap F_2\right) &= \Pr\left(\max_{i\in[\lambda]} W_i < Y_t - X_t\right) \\
&= \prod_{i\in[\lambda]} \Pr\left(W_i < D_t\right) \\
&= \prod_{i\in[\lambda]} \left(1 - \Pr\left(W_i \geq D_t\right)\right).
\end{aligned}$$

We consider the event that we only flip 0-bit in x_t. We modify the probability pessimistically by assuming we flip exactly D_t bits in 0-bit of x_t. In particular, $\Pr\left(W_i \geq D_t\right) \geq \Pr\left(W_{i,+} = D_t\right)$. Let $D_t = Y_t - X_t$. Then, we have

$$\leq \left(1 - \binom{n - X_t}{D_t} \left(\frac{\chi}{n}\right)^{D_t} \times \left(1 - \frac{\chi}{n}\right)^{n - D_t}\right)^{\lambda}.$$

Using $\binom{n-X_t}{D_t} \geq \frac{(n-X_t)^{D_t}}{D_t^{D_t}}$ and $n - X_t \geq \varepsilon n$ gives

$$\begin{aligned}
&\leq \left(1 - \frac{(n - X_t)^{D_t}}{D_t^{D_t}} \left(\frac{\chi}{n}\right)^{D_t} \times \left(1 - \frac{\chi}{n}\right)^{n - D_t}\right)^{\lambda} \\
&\leq \left(1 - \left(\frac{\chi\varepsilon}{D_t}\right)^{D_t} \left(1 - \frac{\chi}{n}\right)^{n - D_t}\right)^{\lambda}.
\end{aligned}$$

Using $(1 - \frac{x}{n})^n \geq e^{-x}(1 - \frac{x^2}{n})$ and $1 - \frac{x^2}{n} \geq \frac{1}{2}$ for sufficiently large n gives

$$\leq \left(1 - \left(\frac{\chi\varepsilon}{D_t}\right)^{D_t} e^{-\chi\frac{n - D_t}{n}} \frac{1}{2}\right)^{\lambda}.$$

Recall that $D_t \leq c \leq n$. Thus, $\frac{n - D_t}{n} \leq 1$ for sufficiently large n.

$$\leq \left(1 - \frac{\frac{1}{2}e^{-\chi}}{\left(\frac{D_t}{\varepsilon\chi}\right)^{D_t}}\right)^{\lambda}.$$

Using $(1-\frac{x}{n})^n \leq e^{-x}$ gives

$$\leq \exp\left(-\frac{1}{2e^{\chi}}\frac{\lambda}{\left(\frac{D_t}{\varepsilon\chi}\right)^{D_t}}\right).$$

If $\left(\frac{D_t}{\varepsilon\chi}\right)^{D_t} \leq \frac{\lambda}{\ln\lambda}$, this gives

$$\leq \exp\left(-\frac{1}{2e^{\chi}}\frac{\lambda}{\lambda/\ln\lambda}\right) = \left(\frac{1}{\lambda}\right)^{\frac{1}{2e^{\chi}}}.$$

So we would like to check the range of D_t which satisfies $\left(\frac{D_t}{\varepsilon\chi}\right)^{D_t} \leq \frac{\lambda}{\ln\lambda}$. It is known that the Lambert function $W(x) = \ln x - \ln\ln x + o(1)$ (Corless et al., 1996; Lehtonen, 2016). We can derive $W(x) \leq \ln(x)$, and thus we have

$$\begin{aligned}\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)} \geq \frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\ln\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)} &= \frac{\ln\lambda - \ln\ln\lambda}{\ln\ln\left(\frac{\lambda}{\ln\lambda}\right) + \ln(\frac{1}{\varepsilon\chi})} \\ &= \frac{\ln\lambda - \ln\ln\lambda}{\ln\ln\lambda - \ln\ln\ln\lambda + \ln(\frac{1}{\varepsilon\chi})} \\ &\geq c := \frac{\kappa\ln\lambda}{\ln\ln\lambda}\end{aligned}$$

for any constant $\kappa \in (0,1)$.

So $D_t \leq c$ implies that $D_t \leq \frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)}$. Next, if $D_t \leq \frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)}$, then by definition of W function ($e^{W(x)} = \frac{x}{W(x)}$), we have

$$\frac{D_t}{\varepsilon\chi} \leq \frac{\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}}{W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)} = \exp\left(W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right)\right).$$

Taking ln gives

$$\ln\left(\frac{D_t}{\varepsilon\chi}\right) \leq W\left(\frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}\right).$$

Considering $y \leq W(x)$ is the solution to the inequality $ye^y \leq x$ and using $x = \frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}, y = \ln\left(\frac{D_t}{\varepsilon\chi}\right)$ give

$$\ln\left(\frac{D_t}{\varepsilon\chi}\right) e^{\ln\left(\frac{D_t}{\varepsilon\chi}\right)} \leq \frac{\ln\left(\frac{\lambda}{\ln\lambda}\right)}{\varepsilon\chi}.$$

Taking exponent on both sides and rearranging the expression give

$$\left(\frac{D_t}{\varepsilon\chi}\right)^{D_t} \leq \frac{\lambda}{\ln\lambda}.$$

Now, we combine the probability of E_t under the case that we have both $F_1 = \{\exists k \in [\lambda], x_t^k = x_t\}$ and $F_2 = \{\max_{i\in[\lambda]} X_t^i > \min_{i\in[\lambda]} X_t^i\}$ hold with Lemma 5.4.4 to give the overall probability of E_t. Using the law of total probability gives:

$$\begin{aligned}\Pr\left(E_t^c\right) &= \Pr\left(E_t^c \mid F_1 \cap F_2\right)\Pr\left(F_1 \cap F_2\right) + \Pr\left(E_t^c \mid F_1^c \cup F_2^c\right)\Pr\left(F_1^c \cup F_2^c\right)\\ &\leq \left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}} + \Pr\left(F_1^c \cup F_2^c\right).\end{aligned}$$

Using Union bound and Lemma 5.4.4 gives

$$\leq \left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}} + 2e^{-\Omega(\lambda)}.$$

For sufficiently large λ, we have $2e^{-\Omega(\lambda)} \leq \left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}}$. So, this leads to

$$\leq 2\left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}}.$$

□

Proof of Lemma 5.4.8. A similar statement works for ΔY_t^i. In this proof, we deal with the case ΔX_t^i. We assume t is even and $X_t < Y_t$. Statement (A) follows immediately from Lemma 5.4.7 because any constant $\kappa \in (\chi, (1+\chi)/2) \subset (0,1)$ for $\chi \in (0,1)$.

Next, we deal with the second inequality. To estimate K, we use Lemma 5.4.6. We denote the number of 1-bits that i-th offspring increases by U_i for $i \in [\lambda]$. Lemma 5.4.6

implies that for all $i \in [\lambda]$,

$$\begin{aligned}\Pr\left(U_i \geq D_t + c\right) &\leq e^{-\chi}\lambda^{\chi}e^{-(c+D_t)\ln\ln\lambda} \\ &= e^{-\chi}\lambda^{\chi-\kappa}e^{-D_t\ln\ln\lambda} := q_k.\end{aligned}$$

We can consider such sampling with at least (D_t+c)-jump as λ independent trials, and thus K is stochastic dominated by a binomial distributed random variable $Bin(\lambda, q_K) \sim K^*$. So

$$E[K^*] = \lambda q_K = e^{-\chi}\lambda^{1+\chi-\kappa}e^{-D_t\ln\ln\lambda}.$$

Using $D_t \leq \frac{\kappa\ln\lambda}{\ln\ln\lambda}$ gives

$$\geq e^{-\chi}\lambda^{1+\chi-2\kappa}$$

We can apply Chernoff's bound to get for any $\delta_1 \in (0,1)$, and using stochastic dominance ($K^* \succcurlyeq K$) gives

$$\Pr\left(K > (1+\delta_1)E[K^*]\right) \leq \Pr\left(K^* > (1+\delta_1)E[K^*]\right).$$

Using Chernoff's bound gives

$$\leq e^{-E[K^*]\delta_1^2/3}.$$

Using the lower bound for $\mathrm{E}(K^*)$ gives

$$\leq e^{\frac{e^{-\chi}\lambda^{1+\chi-2\kappa}\delta_1^2}{3}} = \exp(-\lambda\Omega(1)),$$

where $1+\chi-2\kappa > 0$ directly follows from $\kappa \in (\chi, \frac{1+\chi}{2})$.

To estimate M, we note that

$$\Pr\left(\Delta X_t^i \geq D_t\right) \geq \binom{n-X_t}{D_t}(\frac{\chi}{n})^{D_t}(1-\frac{\chi}{n})^{n-D_t}.$$

Using $\binom{n-X_t}{D_t} \geq \frac{(n-X_t)^{D_t}}{D_t^{D_t}}$ and $n - X_t \geq \varepsilon n$ gives

$$\geq \frac{(n-X_t)^{D_t}}{D_t^{D_t}}\left(\frac{\chi}{n}\right)^{D_t}(1-\frac{\chi}{n})^{n-D_t}.$$

Using $n - X_t \geq \varepsilon n$ gives

$$\geq \frac{(\varepsilon n)^{D_t}}{D_t^{D_t}} \left(\frac{\chi}{n}\right)^{D_t} (1 - \frac{\chi}{n})^{n-D_t}.$$

Using $(1 - \frac{x}{n})^n \geq e^{-x}(1 - \frac{x^2}{n})$ and $1 - \frac{x^2}{n} \geq \frac{1}{2}$ for sufficiently large n gives

$$\geq \frac{1}{2e^{\chi}} \left(\frac{\varepsilon\chi}{D_t}\right)^{D_t} := q_M.$$

M is stochastic dominated the random variable $M^* = Bin(\lambda, q_M)$ So we have

$$E[M^*] = \lambda q_M.$$

$D_t \leq c$ implies that

$$\geq \frac{1}{2e^{\chi}} \lambda \left(\frac{\varepsilon\chi}{c}\right)^{c}.$$

Note that $c = \frac{\kappa \ln \lambda}{\ln \ln \lambda}$ and thus $\left(\frac{\varepsilon\chi}{c}\right)^c = \frac{1}{e^{\frac{\kappa \ln \lambda}{\ln \ln \lambda} \ln\left(\frac{\kappa \ln \lambda}{\varepsilon\chi \ln \ln \lambda}\right)}} \geq \lambda^{-\kappa}$. Then, we have

$$\geq \frac{\lambda^{1-\kappa}}{2e^{\chi}}.$$

where $\kappa < 1$ follows from $\kappa \in (\chi, \frac{1+\chi}{2})$. By using Chernoff's bound on M^* and $M^* \preccurlyeq M$, we have for any $\delta_2 \in (0, 1)$,

$$\begin{aligned}\Pr\left(M \leq (1 - \delta_2)\lambda q_M\right) &\leq \Pr\left(M^* \leq (1 - \delta_2)\lambda q_M\right) \\ &\leq e^{-E[M^*]\delta_2^2/2}.\end{aligned}$$

Using the lower bound for $\mathrm{E}(M^*)$ gives

$$\leq e^{-\lambda^{1-\kappa}\delta_2^2/2e^{\chi}}.$$

Now, we have for any $\delta_1, \delta_2 \in (0, 1)$,

$$\Pr\left(K > (1 + \delta_1)\lambda q_K\right) \leq e^{\frac{e^{-\chi}\lambda^{1+\chi-2\kappa}\delta_1^2}{3}}, \quad \Pr\left(M \leq (1 - \delta_2)\lambda q_M\right) \leq e^{-\lambda^{1-\kappa}\delta_2^2/2e^{\chi}}.$$

We then use the union bound to prove that either of $F_1 = \{K > (1 + \delta_1)\lambda q_K\}$ and $F_2 = \{M \leq (1 - \delta_2)\lambda q_M\}$ occurs with exponentially small probability in λ:

$$\Pr\left(F_1 \cup F_2\right) \leq e^{\frac{e^{-\chi}\lambda^{1+\chi-2\kappa}\delta_1^2}{3}} + e^{-\lambda^{1-\kappa}\delta_2^2/2e^{\chi}}.$$

So we can prove that

$$\Pr\left(\frac{K}{M} \leq \frac{1+\delta_1}{1-\delta_2}\frac{q_K\lambda}{q_M\lambda}\right) \geq \Pr\left(F_1^c \cap F_2^c\right) = 1 - \Pr\left(F_1 \cup F_2\right)$$
$$\geq 1 - e^{\frac{e^{-\chi}\lambda^{1+\chi-2\kappa}\delta_1^2}{3}} - e^{-\lambda^{1-\kappa}\delta_2^2/2e^{\chi}}$$
$$= 1 - e^{-\Omega(\lambda)}.$$

Finally, we want to bound the ratio $\frac{q_K}{q_M}$ in a small quantity. Note that

$$\frac{q_K}{q_M} \leq \frac{e^{-\chi}\lambda^{\chi-\kappa}e^{-D_t\ln\ln\lambda}}{\frac{1}{2e^{\chi}}\left(\frac{\varepsilon\chi}{D_t}\right)^{D_t}} = 2\lambda^{\chi-\kappa}\frac{1}{e^{D_t\ln\ln\lambda}\left(\frac{\varepsilon\chi}{D_t}\right)^{D_t}}.$$

Note that for $y \in [1,c]$, $\phi(y) := e^{y\ln\ln\lambda}\left(\frac{\varepsilon\chi}{y}\right)^y$ attains its minimum at 1 or c from extreme value theorem and the fact that $\phi'(y) < 0$ for $y \in [1,c]$. So $\phi(D_t) \geq \min\{\phi(1),\phi(c)\}$. We can see $\phi(1) = e^{\ln\ln\lambda}$ and $\phi(c) = \lambda^{\kappa}\cdot(\frac{\varepsilon\chi}{c})^c \geq \lambda^{\kappa}\cdot\lambda^{-\kappa} = 1$. Then, $\phi(D_t) \geq 1$.

$$\leq 2\lambda^{\chi-\kappa}.$$

In overall, for any $\chi \in (0,1)$, $\kappa \in (\chi, \frac{\chi+1}{2})$, $\delta_1, \delta_2 \in (0,1)$, we derive

$$\Pr\left(\frac{K}{M} \leq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) \geq 1 - e^{-\Omega(\lambda)}.$$

Note that $\frac{1+\delta_1}{1-\delta_2} = 1 + \frac{\delta_1+\delta_2}{1-\delta_2} = 1+\delta$ where $\delta := \frac{\delta_1+\delta_2}{1-\delta_2} > 0$. The second case follows from a similar analysis and we complete the proof.

□

Proof of Lemma 5.4.9. In this proof, we assume that $Y_t > X_t$ and a similar analysis is applied to the case $Y_t \leq X_t$. Let us choose c in Lemma 5.4.8. For any $\chi \in (0,1)$ and c ($c = \frac{\kappa\ln\lambda}{\ln\ln\lambda}$ where any constant $\kappa \in (\chi, \frac{1+\chi}{2})$), we have

(A): $\Pr\left(\max_{i\in[\lambda]}\Delta X_t^i \geq D_t \mid D_t < c\right) \geq 1 - 2\left(\frac{1}{\lambda}\right)^{\frac{1}{2e^{\chi}}}$

(B): $\Pr\left(\frac{K}{M} \leq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) \geq 1 - e^{-\Omega(\lambda)}$ for any constant $\delta_1, \delta_2 \in (0,1)$

where ΔX_t^i denotes the change of number of 1-bit in i-th offspring of x_t, and

$$K := \{\#\text{ of samples/offspring s.t. } \Delta X_t^i \geq D_t + c\}$$
$$M := \{\#\text{ of samples/offspring s.t. } \Delta X_t^i \geq D_t\}.$$

Given that $D_t \in [1, c)$, some of the offspring may escape from the lower boundary ($Y_t = X_t - c$) of the tube, while there is only a small probability that the algorithm selects such escaping offspring. By using Lemma 5.4.8, we have

$$\Pr\left(D_{t+1} > c \mid D_t < c\right) = \Pr\left(|X_{t+1} - Y_{t+1}| > c \mid D_t < c\right)$$

Let $E = \{\lambda$ offspring cross the diagonal$\}$ and we use the law of total probability

$$\begin{aligned} &= \Pr\left(\Delta X_t - D_t > c \mid D_t < c, E\right) \Pr(E \mid D_t < c) \\ &+ \Pr\left(D_t - \Delta X_t > c \mid D_t < c, E^c\right) \Pr(E^c \mid D_t < c). \end{aligned}$$

Using definition of conditional probability and the condition of c from Lemma 5.4.8 gives

$$\begin{aligned} &\leq \Pr\left(\Delta X_t - D_t > c \mid D_t < c\right) \\ &\qquad + 2\Pr\left(D_t - \Delta X_t > c \mid D_t < c, E^c\right) \left(\frac{1}{\lambda}\right)^{\frac{1}{2e^X}} \\ &\leq \Pr\left(\Delta X_t > D_t + c \mid D_t < c\right) + 2 \cdot \left(\frac{1}{\lambda}\right)^{\frac{1}{2e^X}}. \end{aligned}$$

We define $K := \#\{\text{offspring s.t. } \Delta X_t^i \geq D_t + c\}$ and $M := \#\{\text{offspring s.t. } \Delta X_t^i \geq D_t\}$. The algorithm assigns fitness 1 to M offspring in this case, including those crossing outside the tube (denote their number by K). Since their fitness is the same, the next search point will be selected uniformly at random from M. Using the second condition in Lemma 5.4.8 and the law of total probability again, we have for any constant $\delta_1, \delta_2 \in (0,1)$,

$$\begin{aligned}
&\leq \Pr\left(\Delta X_t > D_t + c \mid D_t < c, \frac{K}{M} < 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) \\
&\quad \times \Pr\left(\frac{K}{M} < 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) \\
&\quad + \Pr\left(\Delta X_t > D_t + c \mid D_t < c, \frac{K}{M} \geq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) \\
&\quad \times \Pr\left(\frac{K}{M} \geq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) + 2 \cdot \left(\frac{1}{\lambda}\right)^{\frac{1}{2e^\chi}}.
\end{aligned}$$

Using (B) in Lemma 5.4.8 gives

$$\begin{aligned}
&\leq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi} \cdot 1 + \Pr\left(\frac{K}{M} \geq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi}\right) + 2\left(\frac{1}{\lambda}\right)^{\frac{1}{2e^\chi}} \\
&\leq 2\frac{1+\delta_1}{1-\delta_2}\left(\frac{1}{\lambda}\right)^{\kappa-\chi} + e^{-\Omega(\lambda)} + 2\left(\frac{1}{\lambda}\right)^{\frac{1}{2e^\chi}}.
\end{aligned}$$

Taking $\delta_1 = \delta_2 = \frac{1}{2}$ gives

$$\leq 6\left(\frac{1}{\lambda}\right)^{\min\{\kappa-\chi, \frac{1}{2e^\chi}\}} + \left(\frac{1}{\lambda}\right)^{\min\{\kappa-\chi, \frac{1}{2e^\chi}\}} + 2\left(\frac{1}{\lambda}\right)^{\min\{\kappa-\chi, \frac{1}{2e^\chi}\}}.$$

Notice that $\kappa \in (\chi, \frac{\chi+1}{2})$ and thus $\kappa - \chi \in (0, \frac{1-\chi}{2})$. It is known that $e^x \geq 1 + x$ for all $x \in \mathbb{R}$. In our case, we derive $\frac{1-\chi}{2} \leq \frac{1}{2e^\chi}$ for any $\chi \in (0,1)$. Then, $\min\{\kappa - \chi, \frac{1}{2e^\chi}\} = \kappa - \chi$. Then, we derive

$$\leq 9\left(\frac{1}{\lambda}\right)^{\kappa-\chi}.$$

We introduce another variable $\gamma := \kappa - \chi$ and we complete the proof.

$$= 9\left(\frac{1}{\lambda}\right)^{\gamma}.$$

□

Proof of Lemma 5.4.10. We use Lemma 5.4.9 can conclude that with probability at most $9\left(\frac{1}{\lambda}\right)^{\gamma}$ where constant $\gamma \in (0, \frac{1-\chi}{2})$, the search point deviates from c-tube where c in Lemma 5.4.8. By Lemma 5.4.7, the offspring crosses the diagonal with probability at least $1-\left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}}$. So we have shown the first part and second parts directly from the fact that at each iteration, when the search point stays within the tube, either $X_{t+1} - X_t \geq 1$ or $Y_{t+1} - Y_t \geq 1$ with probability at least $1 - 9\left(\frac{1}{\lambda}\right)^{\gamma} - \left(\frac{1}{\lambda}\right)^{\frac{1}{2e\chi}} = 1 - O(\frac{1}{\lambda^{\gamma}})$. By taking $\gamma := \frac{1-\chi}{4}$ (and thus in Lemma 5.4.9, $\kappa = \frac{1+3\chi}{4}$), We conclude that either $X_{t+1} - X_t \geq 1$ or $Y_{t+1} - Y_t \geq 1$ with high probability i.e. with probability $1 - O(\frac{1}{\lambda^{(1-\chi)/4}})$. Finally, by taking sufficiently large λ s.t. $1 - O(\frac{1}{\lambda^{(1-\chi)/4}}) \geq \frac{1}{2}$, it yields the positive constant drift $\frac{1}{2}$ for H_t.

□

A.7 Supplementary Materials of Chapter 6

A.7.1 Roadmap of Appendix A.7

The appendix provides additional technical details and support for the main content of our work. It includes a summary of our contributions, technical lemmas for a new level-based theorem, and an extended analytic toolbox with the Negative Drift Theorem and its population-based variant. Additionally, omitted proofs from the main text are presented in detail, along with extra empirical results that present further validation of our findings.

A.7.2 Summary of Our Contributions

We provide a table to summarise our contributions in this paper (see Table A.1). As shown in the table, we present the runtime bounds for evolutionary algorithms on pseudo-Boolean maximin benchmarks. In particular, the runtime bound of PDCoEA on DIAGONAL is

$O(\lambda n(n^2+\lambda^2))$ for $\lambda = \Omega(\log n)$ and this is a much better bound than previous work by Lehre and Lin (2024c) which has $O(\lambda n)$ for $\lambda \geq n^{4/(1-\chi)}$ when analysing $(1,\lambda)$-CoEA.

Algorithm/Benchmark	**Diagonal**	**Discrete-Bilinear**
(1,λ)-EA	Upper: - Lower:$e^{\Omega(n)}$ (Lehre and Lin, 2024c)	Upper: - Lower: -
RLS-PD	Upper: N.A. Lower: $e^{\Omega(n)}$ **(Theorem 6.4.1)**	Upper: $O(n^{1.5})$ $(\alpha,\beta \approx 1/2)$ (Fajardo et al., 2023) Lower: -
(1+1)-CoEA	Upper: N.A. Lower: $e^{\Omega(n)}$ **(Theorem 6.4.2)**	Upper: - Lower: -
(1,λ)-CoEA	Upper: $O(\lambda n)$ for $\lambda \geq n^{4/(1-\chi)}$ (Lehre and Lin, 2024c) Lower: -	Upper: $M_1^2/r - r$ for $\lambda = 2r+1$ with radius $r \in \mathbb{N}$ (Hevia Fajardo and Lehre, 2023) Lower: -
PD-CoEA	Upper: $O(\lambda n(n^2+\lambda^2))$ **for** $\lambda = \Omega(\log n)$ **(Theorem 6.5.1)** Lower: -	Upper: $O(\lambda^3 n)$ (Lehre, 2022) Lower: -

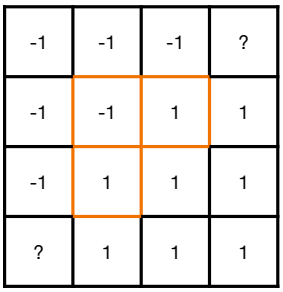
Table A.1: Runtime bounds for evolutionary algorithms on pseudo-Boolean maximin benchmarks. Bounds in bold are the main contributions of this paper. N.A. denotes that the bounds are not available. "-" denotes that the bounds are still missing in current literature. For DIAGONAL game, we provide the tail bounds for the runtime of algorithms finding the optimum. In particular, for $(1,\lambda)$-CoEA on DIAGONAL, the mutation rate $\chi \in (0,1)$ is constant and note that Lehre and Lin (2024c) only shows the runtime of $(1,\lambda)$-CoEA for finding the approximation of the NE of DIAGONAL. For DISCRETE-BILINEAR, we present the upper bound for the expectation of runtime for RLS-PD finding the optimum under the configuration $\alpha,\beta = \frac{1}{2} \pm O(\frac{1}{\sqrt{n}}) \approx \frac{1}{2}$ and tail bound for the runtime of PDCoEA as shown in previous work. For $(1,\lambda)$-CoEA, Hevia Fajardo and Lehre (2023) considered the search domain in $\mathbb{Z}$ and $M_t := |x_t - \beta| + |y_t - \alpha|$ for $t \in \mathbb{N}$. Note that two $(1,\lambda)$-CoEAs presented here are different in both the search domains and mechanisms.

A.7.3 Partitions in Level-Based Analysis

Before showing our main result (Theorem 6.5.1), we define the following partitions that we will use in Theorem 6.3.1. Given $(U_0 \times V_0), \ldots, (U_n \times V_n)$, we define $U_0 = \mathcal{X}$ and $V_0 = \mathcal{Y}$ and for $j \in \{1, \ldots, n\}$, where we suppose $n = 4k$ (and thus $n/2 = 2k$ to simplify the notation during our calculation. One can use a different alternating partition for odd n)

$$\begin{aligned} U_{2j} &:= A_{2j} \text{ and} \\ U_{2j+1} &:= A_{2j+1} \cup A_{2j+2} \text{ and} \\ V_{2j} &:= \{y \in \{0,1\}^n \mid |y|_1 = 2j, 2j+1\} \text{ and} \\ V_{2j+1} &:= \{y \in \{0,1\}^n \mid |y|_1 = 2j+1\}. \end{aligned} \tag{A.3}$$

Given two populations of solutions $P_t \in \mathcal{X}^\lambda, Q_t \in \mathcal{Y}^\lambda$, we define the current level by $j := \max\{i \in [n] \cup \{0\} \mid |(P_t \times Q_t) \cap (A_i \times B_i)| \geq \gamma_0 \lambda^2\}$ where $\gamma_0 \in (0,1)$ is constant and $\lambda \in \mathbb{N}$ is the population size.



$$\begin{aligned} u_0 &= \Pr_{x \sim \text{Unif}(P)}(x \in A_{<j}) & v_0 &= \Pr_{y \sim \text{Unif}(Q)}(y \in B_{<j}) \\ u_1 &= \Pr_{x \sim \text{Unif}(P)}(x \in A_j) & v_1 &= \Pr_{y \sim \text{Unif}(Q)}(y \in B_j) \\ u_2 &= \Pr_{x \sim \text{Unif}(P)}(x \in A_{j+1}) & v_2 &= \Pr_{y \sim \text{Unif}(Q)}(y \in B_{j+1}) \\ u_3 &= 1 - u_0 - u_1 - u_2 & v_3 &= 1 - v_0 - v_1 - v_2 \end{aligned}$$

$$r_1 = \Pr_{x \sim \text{Unif}(P), y \sim \text{Unif}(Q)}(\textsc{Diagonal}(x,y) = 1 \mid x \in A_{<j}, y \in B_{<j})$$
$$r_2 = \Pr_{x \sim \text{Unif}(P), y \sim \text{Unif}(Q)}(\textsc{Diagonal}(x,y) = 1 \mid x \in A_{>j+1}, y \in B_{>j+1})$$

Figure A.4: 2-pixel-partition and orange pixels denote the current and the next levels.

A.7.4 Technical Lemmas for a New Level-Based Theorem and Analysis for Condition (G2b)

Lemma A.7.1 (Lemma 1 in (Lehre, 2022))**.** *Given subsets $A \subseteq X$, $B \subseteq Y$, assume that for any $\delta > 0$ and $\gamma \in (0,1)$, the sample $(x,y) \sim \mathcal{D}(P,Q)$ satisfies*

$$\Pr(x \in A)\Pr(y \in B) \geq (1+\delta)\gamma.$$

Then the random variable $Z_{t+1} := |(P_{t+1} \times Q_{t+1}) \cap A \times B|$ satisfies

(1) $\mathrm{E}_t(Z_{t+1}) \geq \lambda(1-1)(1+\delta)\gamma$.

(2) $\mathrm{E}_t\big(e^{-\eta Z_{t+1}}\big) \leq e^{-\eta\lambda(\gamma\lambda-1)}$ *for* $0 \leq \eta \leq (1-(1+\delta)^{-1/2})/\lambda$.

(3) $\Pr_t\left(Z_{t+1} < \lambda(\gamma\lambda-1)\right) \leq \exp\left(-\delta_1\left(1-\sqrt{\frac{1+\delta_1}{1+\delta}}\gamma\right)\right)$ *for* $\delta_1 \in (0,\delta)$.

Lemma A.7.2. *Given subsets $A_i \subseteq \mathcal{X}, B_i \subseteq \mathcal{Y}$ for $i \in [m]$ where $A_1 = \mathcal{X}$ and $B_1 = \mathcal{Y}$, $m_2 \in \mathbb{N}$ where $m_2 < m$, and any population $P_t \times Q_t \subseteq \mathcal{X}^\lambda \times \mathcal{Y}^\lambda$ at iteration $t \in \mathbb{N}$ where $\lambda \in \mathbb{N}$, we define the number of individuals in level i at iteration t by $X_t^{(i)} := |(P_t \times Q_t) \cap (A_i \times B_i)|$ and current level at iteration t by $Y_t := \max\{i \in [m] \mid X_t^{(i)} \geq \gamma_0\lambda\}$ where $\gamma_0 \in (0,1)$.*

If there exist $z_{m_2}, \ldots, z_{m-1}, \delta \in (0,1)$ such that for any populations $P_t \in \mathcal{X}^\lambda$ and $Q_t \in \mathcal{Y}^\lambda$ with current level $Y_t \geq m_2$ and for any $t \geq \mathbb{N}$,

(G2b) for $(x,y) \sim \mathcal{D}(P_t,Q_t)$, if $\Pr(x \in A_{Y_t})\Pr(y \in B_{Y_t}) \geq (1+\delta)\gamma_0$, then

$$\Pr\left(Y_{t+1} < Y_t \mid Y_t \geq m_2\right) \leq e^{-\Omega(\lambda)}.$$

Moreover, if $Y_0 = m_2$, then $\Pr\left(Y_t < m_2\right) \leq e^{-\Omega(\lambda)}$ for any $t \geq 0$.

Proof. From the proof of Theorem 3 in (Lehre, 2022), $Y_{t+1} < Y_t$ holds if and only if $X_{t+1}^{(Y_t)} < \gamma_0\lambda^2$. Note that the $(G2b)$ condition satisfies the condition in Lemma A.7.1 and

thus using (3) in Lemma A.7.1 gives

$$\Pr\left(Y_{t+1} < Y_t \mid Y_t \geq m_2\right) = \Pr\left(X_{t+1}^{(Y_t)} < \gamma_0\lambda^2 \mid Y_t \geq m_2\right) \leq e^{-c\lambda} \tag{A.4}$$

where $c > 0$ is some constant. For the second claim, we use proof by induction. If $t = 0$, then the claim holds trivially. Now, suppose the claim holds when $t = N$, i.e.

$$\Pr(Y_N < m_2) \leq e^{-\Omega(\lambda)}.$$

For $t = N + 1$, by using the law of total probability, we have

$$\begin{aligned}\Pr(Y_{N+1} < m_2) &= \Pr\left(Y_{N+1} < m_2 \mid Y_N < m_2\right) \cdot \Pr\left(Y_N < m_2\right) \\ &\quad + \Pr\left(Y_{N+1} < m_2 \mid Y_N \geq m_2\right) \cdot \Pr\left(Y_N \geq m_2\right) \\ &\leq 1 \cdot \Pr\left(Y_N < m_2\right) + \Pr\left(Y_{N+1} < m_2 \mid Y_N \geq m_2\right) \cdot 1.\end{aligned}$$

Using the induction hypothesis gives

$$\leq e^{-\Omega(\lambda)} + \Pr\left(Y_{N+1} < m_2 \leq Y_N \mid Y_N \geq m_2\right).$$

Using Eq. (A.4) gives

$$\leq 2 \cdot e^{-\Omega(\lambda)} = e^{-\Omega(\lambda)}.$$

Thus, the proof is completed. □

We need three additional lemmas to lower the probability that the individuals in the previous level are selected. We define the set of pixels by using for $j \in [n]$,

$$A_j := \{x \in \{0,1\}^n \mid |x|_1 = j\}; B_j := \{y \in \{0,1\}^n \mid |y|_1 = j\}.$$

Lemma A.7.3. *Given two populations P and Q in PDCoEA from Algorithm 15 with population size $c \log n \leq \lambda \in$ poly(n) where $c > 0$ is some constant and constant mutation rate $\chi \in (0, \ln 2/2)$ on* DIAGONAL *with problem size n, assume constant $\gamma_0 > (e^{2\chi}/2)^{1/3}$*

and sufficiently large n. Suppose Algorithm 15 initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$ with the runtime of Algorithm 15 finding the optimum $T \in \mathbb{N}$. Let $R_t(i) := \sum_{j=1}^{\lambda} \mathbf{1}_{\{I_t(j)=i\}}$ where $I_t(i)$ represents the parent of the i-th individual at iteration $t \leq T$. Then, given a constant $\beta \in (1/7, 1/6)$, the reproductive rate of each P-individual and Q-individual between $A'_{\beta n}$ and $A'_{n/6}$ (or $B'_{\beta n}$ and $B'_{n/6}$) is bounded by $1 + o(1)$, i.e. for all $i \in [\lambda]$ and $z \in \{0,1\}^n$,

$$\mathrm{E}\Big(R_t(i) \mid \beta n < H(P_t(i), z) < \frac{n}{6}\Big) \leq 1 + o(1);$$
$$\mathrm{E}\Big(R_t(i) \mid \beta n < H(Q_t(i), z) < \frac{n}{6}\Big) \leq 1 + o(1).$$

where H is the Manhattan distance and $|z|_1 = \beta n$.

Proof of Lemma A.7.3. Without loss of generality, we consider the reproductive rate for P-individual between level βn and $n/6$. Before proving the lemma, we need to introduce the following notations:

$$R_0 := \{x \in \{0,1\}^n \mid |x|_1 \leq \beta n\};$$
$$R_1 := \{x \in \{0,1\}^n \mid |x|_1 > \beta n\}.$$
$$S_0 := \{y \in \{0,1\}^n \mid |y|_1 \leq \beta n\};$$
$$S_1 := \{y \in \{0,1\}^n \mid |y|_1 > \beta n\}.$$

The fraction probabilities among the populations P_t, Q_t are denoted by

$$\rho_0 := \Pr(x \in R_0); \rho_1 := \Pr(x \in R_1).$$
$$\sigma_0 := \Pr(y \in S_0); \sigma_1 := \Pr(y \in S_1).$$

Now, we consider $\textsc{Select}(P, Q)$ only. Given two uniformly sampled pairs $(x_1, y_1), (x_2, y_2) \sim \mathrm{Unif}(\{0,1\}^n \times \{0,1\}^n)$, there are three main cases in which we select either x (or y) in R_0 (or S_0):

(1) Sample both (x_1, y_1) and (x_2, y_2) in R_0, then no matter $(x_1, y_1) \succeq (x_2, y_2)$ or $(x_1, y_1) \not\succeq$

(x_2, y_2), we both have x_1 or $x_2 \in R_0$; (Or sample both (x_1, y_1) and (x_2, y_2) in S_0, then no matter $(x_1, y_1) \succeq (x_2, y_2)$ or $(x_1, y_1) \not\succeq (x_2, y_2)$, we both have y_1 or $y_2 \in S_0$) ;

(2) $(x_1, y_1) \succeq (x_2, y_2)$ and $x_1 \in R_0$ while x_2 does not need to be in R_0 (Or $(x_1, y_1) \succeq (x_2, y_2)$ and $y_1 \in S_0$ while y_2 does not need to be in S_0);

(3) $(x_1, y_1) \not\succeq (x_2, y_2)$ and $x_2 \in R_0$ while x_1 does not need to be in R_0 (Or $(x_1, y_1) \not\succeq (x_2, y_2)$ and $y_2 \in S_0$ while y_1 does not need to be in S_0).

Note that if $(x, y) \in R_0 \times S_1$ or $R_1 \times S_0$, then $g(x, y) = -1$ or $g(x, y) = 1$ respectively. Since $R_0 \times S_0, R_1 \times S_1$ contains more possible scenarios, (i.e. for $(x, y) \in R_0 \times S_0, R_1 \times S_1$, $g(x, y)$ can be either -1 or 1. we assume that if $(x, y) \in R_0 \times S_0$, then $g(x, y) = -1$ with probability $r_1 \in [0, 1]$ and $g(x, y) = 1$ with probability $1 - r_1 \in [0, 1]$. Furthermore, if $(x, y) \in R_1 \times S_1$, then $g(x, y) = -1$ with probability $r_2 \in [0, 1]$ and $g(x, y) = 1$ with probability $1 - r_1 \in [0, 1]$.

For $x \in R_0$, there are possible pairs probabilities as follows.

$$
\begin{aligned}
&\lambda \cdot \Pr_{(x,y)\sim \textsc{Select}(P_t,Q_t)}(\text{events } \{(2),(3)\}) \\
=&(1-r_1)\rho_0\sigma_0\rho_0\sigma_1 + r_1\rho_0\sigma_0\rho_1\sigma_0 + r_1\rho_0\sigma_1\rho_0\sigma_0 \\
&+ r_1\rho_0\sigma_0\rho_0\sigma_1 + r_1\rho_0\sigma_0\rho_0\sigma_0 \\
&+ \rho_0\sigma_1\rho_0\sigma_1 + (1-r_1)\rho_0\sigma_0\rho_0\sigma_0 + (1-r_1)\rho_0\sigma_1\rho_0\sigma_0 \\
=&\rho_0\left(\rho_0(\sigma_0+\sigma_1)^2 + \rho_1\sigma_0^2 r_1\right)
\end{aligned}
$$

Using $\rho_0 + \rho_1 = \sigma_0 + \sigma_1 = 1$ and $r_1 \leq 1$ gives

$$
\leq \rho_0\left(\rho + \rho_1\sigma_0^2\right) \leq \rho_0 \leq 1.
$$

For $y \in S_0$, there are possible pairs probabilities as follows.

$$\begin{aligned}
&\lambda \cdot \Pr_{(x,y)\sim\textsc{Select}(P_t,Q_t)}(\text{events } \{(2),(3)\}) \\
=&\rho_1\sigma_0\rho_1\sigma_0 + (1-r_1)\rho_0\sigma_0\rho_0\sigma_0 \\
&+ (1-r_1)\rho_1\sigma_0\rho_0\sigma_0 + (1-r_1)\rho_0\sigma_0\rho_0\sigma_1 \\
&+ (1-r_1)\rho_0\sigma_0\rho_1\sigma_0 + r_2\rho_1\sigma_0\rho_1\sigma_1 \\
&+ \rho_1\sigma_0\rho_0\sigma_1 + r_1\rho_0\sigma_0\rho_0\sigma_0 \\
&+ \rho_0\sigma_1\rho_1\sigma_0 + r_1\rho_1\sigma_0\rho_0\sigma_0 + r_1\rho_0\sigma_0\rho_1\sigma_0 \\
=&\sigma_0\left(2\rho_0\rho_1(\sigma_0+\sigma_1) + \rho_0^2(\sigma_0+\sigma_1-\sigma_1 r_1) + \rho_1^2(\sigma_0+\sigma_1 r_2)\right)
\end{aligned}$$

Using $\rho_0+\rho_1 = \sigma_0+\sigma_1 = 1$ and $r_1, r_2 \in [0,1]$ gives

$$\leq \sigma_0\left(2\rho_0\rho_1 + \rho_0^2 + \rho_1^2\right) = \sigma_0 \leq 1.$$

Above all, we can compute the probability that (recall $\lambda \in \mathrm{poly}(n)$ and assume sufficiently large n), for each $i \in [\lambda]$,

$$\Pr_{(x,y)\sim\textsc{Select}(P_t,Q_t)}(x \in R_0) \leq \frac{1}{\lambda^2} + \frac{1}{\lambda}\cdot 1 = \frac{1}{\lambda}(\frac{1}{\lambda}+1).$$

Thus for each $i \in [\lambda]$, the reproductive rate for either P-individual or Q-individual is bounded by $\frac{1}{\lambda}(\frac{1}{\lambda}+1)\cdot\lambda = 1+\frac{1}{\lambda} = 1+o(1)$ for sufficiently large n. □

Lemma A.7.4. *Assume that for a sufficiently large constant c, it holds $c\log n \leq \lambda \in \mathrm{poly}(n)$ and Algorithm 15 initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$. Given some constant $\beta \in (1/7, 1/6)$, we define*

$$T_{\beta n} := \inf\{t > 0 \mid (P_t \times Q_t) \cap (A'_{\lfloor \beta n\rfloor} \times B_{\lfloor(\beta n\rfloor})' \neq \emptyset\}.$$

where P_t and Q_t are the population of search points from Algorithm 15 on Diagonal. *Then, $\Pr\left(T_{\beta n} \leq e^{\Omega(n)}\right) \leq e^{-\Omega(n)}$. Furthermore, for $\tau \in \mathbb{N}$,*

$$\Pr\left(\cup_{t=0}^{\tau}(P_t \times Q_t) \cap (A'_{<\beta n} \times B'_{<\beta n}) \neq \emptyset\right) \leq \tau e^{-\Omega(n)}$$

where $A_{<\beta n} := \cup_{k=0}^{\lfloor\beta n\rfloor} A'_k$ and $B_{<\beta n} := \cup_{k=0}^{\lfloor\beta n\rfloor} B'_k$.

Proof of Lemma A.7.4. From Lemma A.7.3, for all $i \in [\lambda]$, (assuming sufficiently large n),

$$\mathrm{E}\Big(R_t(i) \mid \beta n < H(P_t(i), z) < \frac{n}{6}\Big) \leq 1 + o(1);$$
$$\mathrm{E}\Big(R_t(i) \mid \beta n < H(Q_t(i), z) < \frac{n}{6}\Big) \leq 1 + o(1),$$

where H is the Manhattan distance and $z \in \{0,1\}^n$ with $|z|_1 = \beta n$. We satisfy condition (1) in Theorem 2.8.5 with $a(n) := \beta n$ (a constant $\beta \in (1/7, 1/6)$), $b(n) := n/6$ and $\alpha_0 = 1$. By setting $\psi := \ln(1)/\chi + \frac{1}{400}$ and noting that $b(n)/n = 1/6 < \min\{1/5, 1/2 - \sqrt{\psi(2-\psi)}/2\}$, we satisfy condition (2)-(3) in Theorem 2.8.5. Thus, Theorem 2.8.5 yields that $\Pr\left(T_{\beta n} \leq e^{\Omega(n)}\right) \leq e^{-\Omega(n)}$. Moreover, by using a union bound for each iteration, we can easily derive the second statement. □

Lemma A.7.5. *With the same setting as Lemma A.7.4, assume P_t and Q_t are the populations of search points from Algorithm 15 on* DIAGONAL *at iteration $t \in \mathbb{N}$ and the current level of the populations at level $j \in (\beta n, n)$. For any $t \in$ poly(n),*

$$\Pr\left(|P_t \cap A'_{<j}| \leq \chi e^{-\chi}\gamma_0\lambda/7\right) \leq e^{-\Omega(n)};$$
$$\Pr\left(|Q_t \cap B'_{<j}| \leq \chi e^{-\chi}\gamma_0\lambda/7\right) \leq e^{-\Omega(n)},$$

where $A_{<j} := \cup_{k=0}^{j-1} A'_k$ and $B_{<j} := \cup_{k=0}^{j-1} B'_k$ (A'_k, B'_k are defined in Section 4.1).

Proof of Lemma A.7.5. Suppose P_t and Q_t are the populations of search points from Algorithm 15 on DIAGONAL at iteration t. Without loss of generality, we consider P-individual. We divide the analysis into two cases.

(1) The current level of $P_t \times Q_t$ remains at the same level j as previous iteration $t-1$. In this case, after mutation in iteration $t-1$, for each $i \in [\lambda]$,

$$\begin{aligned}\Pr_{(x,y)\sim\textsc{Mutate}(P_{t-1},Q_{t-1})}(x \in A_{<j} \mid j \geq \beta n) &\geq \binom{j}{1}\frac{\chi}{n}(1-\frac{\chi}{n})^{n-1} \\ &\geq \frac{j\chi}{n}e^{-\chi}(1-o(1))\end{aligned}$$

Using the condition $j \geq \beta n$ gives

$$\geq \beta\chi e^{-\chi}\left(1 - o(1)\right).$$

By the law of total probability, we have

$$\begin{aligned}\Pr_{(x,y)\sim\textsc{Mutate}(P_{t-1},Q_{t-1})}\left(x \in A_{<j}\right) = &\Pr_{(x,y)\sim\textsc{Mutate}(P_{t-1},Q_{t-1})}\left(x \in A_{<j} \mid j \geq \beta n\right) \cdot \Pr\left(j \geq \beta n\right) \\ &+ \Pr_{(x,y)\sim\textsc{Mutate}(P_{t-1},Q_{t-1})}\left(x \in A_{<j} \mid j < \beta n\right) \cdot \Pr\left(j < \beta n\right).\end{aligned}$$

Using Lemma A.7.4 with $\tau \in \text{poly}(n)$ gives that $\Pr\left(\cup_{t=0}^{\tau}(P_t \times Q_t) \cap (A_{<\beta n} \times B_{<\beta n}) \neq \emptyset\right) \leq \tau e^{-\Omega(n)} = e^{-\Omega(n)}$. This means that the probability that the current level is below βn is $e^{-\Omega(n)}$.

$$\begin{aligned}&\geq \beta\chi e^{-\chi}\left(1 - o(1)\right)\left(1 - e^{-\Omega(n)}\right) \\ &= \beta\chi e^{-\chi}\left(1 - o(1)\right) := \theta_1.\end{aligned}$$

So $|P_t \cap A_{<j}| \succeq \text{Bin}(\gamma_0\lambda, \theta_1) := M_1$. Then, using stochastic dominance gives for any $\delta \in (0, 1)$

$$\Pr\left(|P_t \cap A_{<j}| \leq (1-\delta)\beta\chi e^{-\chi}\left(1 - o(1)\right)\gamma_0\lambda\right) \leq \Pr\left(M_1 \leq (1-\delta)\gamma_0\lambda\theta_1\right)$$

Using Chernoff's bound and $\lambda \in \text{poly}(n)$ give

$$\leq e^{-\Omega(\lambda)} = e^{-\Omega(n)}.$$

By taking suitable δ such that $(1-\delta)\beta\left(1 - o(1)\right) \leq 1/7$, we obtain the desired result.

(2) The current level of $P_t \times Q_t$ moves to the level $j' > j$ where j is the "current" level of $P_{t-1} \times Q_{t-1}$. In this case, after mutation in iteration $t-1$, for each $i \in [\lambda]$,

$$\begin{aligned}\Pr_{(x,y)\sim\textsc{Mutate}(P_{t-1},Q_{t-1})}\left(x \in A_{<j'}\right) &\geq \left(1 - \frac{\chi}{n}\right)^n \\ &\geq e^{-\chi}(1 - o(1)) := \theta_2.\end{aligned}$$

So $|P_t \cap A_{<j'}| \succeq \mathrm{Bin}(\gamma_0\lambda, \theta_2) := M_2$. Then, using stochastic dominance gives for any $\delta \in (0,1)$

$$\Pr\left(|P_t \cap A_{<j'}| \leq (1-\delta)e^{-\chi}\left(1-o(1)\right)\gamma_0\lambda\right) \leq \Pr\left(M_2 \leq (1-\delta)\gamma_0\lambda\theta_2\right)$$

Using Chernoff's bound and $\lambda \in \mathrm{poly}(n)$ give

$$\leq e^{-\Omega(\lambda)} = e^{-\Omega(n)}.$$

By taking suitable δ such that $(1-\delta)\left(1-o(1)\right) \leq \chi/7$, we obtain the desired result.

□

A.7.5 Omitted Proofs

Theorem 6.3.1. *Given subsets $A_j \subseteq \mathcal{X}, B_j \subseteq \mathcal{Y}$ for $j \in [m]$ where $A_1 = \mathcal{X}$ and $B_1 = \mathcal{Y}$, define $T := \min\{t\lambda \mid (P_t \times Q_t) \cap (A_m \times B_m) \neq \emptyset\}$, where for all $t \in \mathbb{N}, P_t \in \mathcal{X}^\lambda$ and $Q_t \in \mathcal{Y}^\lambda$ are the populations of Algorithm 5 in generation t. Denote the current level of the population by $j := \max\{\ i \in [m-1] \mid |(P \times Q) \cap (A_i \times B_i)| \geq \gamma_0\lambda^2\}$ where $\gamma_0 \in (0,1)$.*

Given $m_2 \in \mathbb{N}$ where $m_2 < m$, suppose $P_0 \times Q_0$ is initialised with the current level $j \geq m_2$. If there exist $z_{m_2}, \ldots, z_{m-1}, \delta \in (0,1)$ such that for any populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ with the current level $j \geq m_2$,

(G1) for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_{j+1})\Pr(y \in B_{j+1}) \geq z_j$;

(G2a) for all $\gamma \in (0, \gamma_0)$, if $|(P \times Q) \cap (A_{j+1} \times B_{j+1})| \geq \gamma\lambda^2$, then for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_{j+1})\Pr(y \in B_{j+1}) \geq (1+\delta)\gamma$;

(G2b) for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in A_j)\Pr(y \in B_j) \geq (1+\delta)\gamma_0$;

(G3) and the population size $\lambda \in \mathbb{N}$ satisfies $\lambda \geq c' \log(m/z_*)$ for a sufficiently large constant c', where $z_: = \min_{m_2 \leq i \leq m-1} z_i$;

then there exists a constant $c'' > 0$ such that any $r > 0$,

$$\Pr\left(T \geq r\frac{c''\lambda}{\delta}\left(\lambda^2(m-m_2) + \sum_{i=m_2}^{m-1}\frac{1}{z_i}\right)\right) \leq \frac{1+o(1)}{r}.$$

Proof of Theorem 6.3.1. We follow a similar routine used in (Lehre, 2022). We define the same level function (see Definition 3 in (Lehre, 2022)) and stochastic processes. For any $j \in [m]$ and time $t \geq 0$, $X_t^{(j)} := |(P_t \times Q_t) \cap (A_j \times B_j)|$ denote the number of pairs in level j at time t and the current Y_t of the population is defined as $Y_t := \max\{j \in [m] \mid X_t^{(j)} \geq \gamma_0\lambda^2\}$. Finally, $Z_t := 0$ if $Y_t = m$ and otherwise if $Y_t < m$, we define

$$Z_t := g(X_t^{(Y_t+1)}, Y_t),$$

where for all $k \in [\lambda^2]$ and for all $j \in [m-1]$, $g(k,j) := g_1(k,j) + g_2(k,j)$ and

$$\begin{aligned} g_1(k,j) &:= \frac{\eta}{1+\eta} \cdot \big((m-j)\lambda^2 - k\big) \\ g_2(k,j) &:= \varphi \cdot \left(\frac{e^{-\eta k}}{q_j} + \sum_{i=j+1}^{m-1}\frac{1}{q_i}\right), \end{aligned}$$

where the parameters $\eta, \varphi \in (0,1)$ are constant and for $j \in [m-1]$, $q_j := \lambda z_j/(4+\lambda z_j)$. As shown in the proof of Theorem 3 in (Lehre, 2022) and property (1) & (2) of level functions, by setting $z^* := \min_{m_2 \leq i \leq m-1} z_i$, for $k \in [\lambda^2], j \geq m_2$,

$$\begin{aligned} 0 \leq g(k,j) &\leq g(0,m_2) \\ &= \frac{\eta(m-m_2)\lambda^2}{1+\eta} + \sum_{i=m_2}^{m-1}\frac{\varphi}{q_i} \\ &< (m-m_2)\left(\eta\lambda^2 + \frac{4\varphi}{\lambda z_*} + \varphi\right) \end{aligned}$$

Using $\lambda \in \text{poly}(n)$ and $\varphi, z_* \in (0,1)$ gives

$$< \frac{3\eta\lambda^2(m-m_2)}{z_*}.$$

Also, as shown in (Lehre, 2022) that $g(k,j) \leq g(0,1) < \infty$ for all $k \in [\lambda^2] \cup \{0\}$ and $j \geq m_2$, we obtain: if $Y_t \geq m_2$, then

$$0 \leq Z_t \leq g(0,m_2) < \infty. \tag{A.5}$$

Eq. (A.5) means that we satisfy the finite state space condition in the additive drift theorem (He and Yao, 2001). For now, suppose we always have $Y_t \geq m_2$ for all $t \in \mathbb{N}$ and denote this by event E. Following the proof of Theorem 3 in (Lehre, 2022), we consider the drift:

$$\begin{aligned}\mathrm{E}_t(\Delta_{t+1} \mid E) :&= \mathrm{E}_t(Z_t - Z_{t+1} \mid E) \\ &= \mathrm{E}_t\Big(g(X_t^{(Y_t+1)}, Y_t) - g(X_t^{(Y_{t+1}+1)}, Y_{t+1}) \mid E\Big).\end{aligned}$$

Using (G1), (G2a), (G2b) with $j \geq m_2$ in the proof of Theorem 3 in (Lehre, 2022) gives the following drift: there exists $\varphi = \delta(1-\delta') > 0$ for a constant $\delta' \in (0,1)$ such that for any constant $\rho \in (0,1)$ and sufficiently large λ

$$\geq \frac{\eta\varphi(1-\rho)}{1+\eta}.$$

Also, it is verified in the proof of Theorem 3 (Lehre, 2022) that $\mathrm{E}(T \mid E) < \infty$ and thus fulfil the conditions of additive drift theorem. By the additive drift theorem, we obtain: there exists $c'' > 1$ such that

$$\mathrm{E}(T \mid E) \leq \frac{c''\lambda}{\delta}\left((m-m_2)\lambda^2 + 16\sum_{i=m_2}^{m-1}\frac{1}{z_i}\right) := \tau.$$

Using conditional Markov inequality gives that for any $r > 0$,

$$\Pr\left(T \geq r\tau \mid E\right) \leq \frac{\mathrm{E}(T \mid E)}{r\tau} \leq \frac{1}{r}. \tag{A.6}$$

Now, we take into account the event E_c and by using the law of total probability, we obtain that for any $r > 0$,

$$\begin{aligned}\Pr\left(T \geq r\tau\right) = &\Pr\left(T \geq r\tau \mid E\right) \cdot \Pr\left(E\right) \\ &+ \Pr\left(T \geq r\tau \mid E^c\right) \cdot \Pr\left(E^c\right).\end{aligned}$$

Using Eq. (A.6) and Lemme A.7.2 under sufficiently large population size λ gives

$$\leq \frac{1}{r} \cdot 1 + 1 \cdot e^{-\Omega(\lambda)} = \frac{1}{r}(1 + o(1)).$$

□

Lemma 6.2.1. *Given a* DIAGONAL *game denoted by* g, $(x, y) \succeq_g (u, v)$ *if and only if* $(x, y), (u, v)$ *satisfy one of the following:*

(1) $|x|_1 < |v|_1, |x|_1 < |y|_1$ *and* $|u|_1 < |y|_1$;

(2) $|x|_1 \geq |v|_1, |x|_1 < |y|_1$ *and* $|u|_1 < |y|_1$;

(3) $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ *and* $|u|_1 < |y|_1$;

(4) $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ *and* $|u|_1 \geq |y|_1$.

Proof of Lemma 6.2.1. To see the "iff" equivalence relation, we list all possible configurations in the table.

Suppose we have $(x, y) \succeq_g (u, v)$. Note that g only takes two values. Thus, in the dominance relation,

$$g(x, v) \geq g(x, y) \geq g(u, y),$$

there are only 8 possible cases in total, which shows in the following tables.

Configuration	$g(x, v)$	$g(x, y)$	$g(u, y)$
Config. pairs 1	-1	-1	-1
Config. pairs 2	-1	-1	1
Config. pairs 3	-1	1	-1
Config. pairs 4	-1	1	1
Config. pairs 5	1	-1	-1
Config. pairs 6	1	-1	1
Config. pairs 7	1	1	-1
Config. pairs 8	1	1	1

From the table, we see that only configuration pairs $1, 5, 7, 8$ can have dominance relation $(x, y) \succeq_g (u, v)$. We can easily derive the equivalence relation as follows.

(a) Configuration pair 1 means that $|x|_1 < |v|_1, |x|_1 < |y|_1$ and $|u|_1 < |y|_1$ and this is equivalent to case (1) in Lemma 6.2.1.

(b) Configuration pair 8 means that $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ and $|u|_1 \geq |y|_1$ and this is equivalent to case (4) in Lemma 6.2.1.

(c) Configuration pair 5 means that $|x|_1 \geq |v|_1, |x|_1 < |y|_1$ and $|u|_1 < |y|_1$ and this is equivalent to case (2) in Lemma 6.2.1.

(d) Configuration pair 7 means that $|x|_1 \geq |v|_1, |x|_1 \geq |y|_1$ and $|u|_1 < |y|_1$ and this is equivalent to case (3) in 6.2.1.

Thus, the proof is completed. □

Theorem 6.4.1. *Consider RLS-PD on* DIAGONAL *and problem size $n \in \mathbb{N}$. The runtime of RLS-PD for finding the maximin optimum of* DIAGONAL *is $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.*

Proof of Theorem 6.4.1. Suppose we have search point $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ in iteration $t \in \mathbb{N}$ and define $M_t := 2n - X_t - Y_t$ where $X_t = |x_t|_1$ and $Y_t = |y_t|_1$. Define $T := \inf\{t \geq 0 \mid M_t = n\}$ and $T_\varepsilon := \inf\{t \geq 0 \mid M_t < \varepsilon n\}$ for any $\varepsilon > 0$. We can see from the definition that $T \geq T_\varepsilon$. Next, we want to show there exists some constant $\varepsilon > 0$ such that $\Pr(T_\varepsilon \leq e^{cn}) \leq e^{-cn}$ for some constant $c > 0$.

Instead of considering the algorithm exactly reaches the optimum in this game, we first consider the algorithm reaches a εn-approximation to the optimum where $\varepsilon \in (0, 1/4)$. Assume that $\varepsilon n < M_t = s \leq n/4$ at iteration $t \in \mathbb{N}$, then recall that

$$T_\varepsilon := \inf\{t \geq 0 \mid M_t \leq \varepsilon n\}.$$

Now, let us check the conditions of Theorem 2.8.2. Firstly, we show that the initial search point (X_0, Y_0) satisfies $M_0 > n/4$ with high probability. Note that RLS-PD initialises the search point uniformly at random. So X_0, Y_0 are subject to Binomial distribution $\mathrm{Bin}(n, 1/2)$. We compute the probability of $M_0 \leq n/4$.

$$\Pr\left(M_0 \leq \frac{n}{4}\right) = \Pr\left(X_0 + Y_0 \geq \frac{7n}{4}\right)$$

By using Chernoff's bound on $X_0 + Y_0 \sim \mathrm{Bin}(2n, 1/2)$, we have

$$\leq \exp\left(-\frac{n(\frac{3}{4})^2}{2 + \frac{3}{4}}\right) = \exp\left(-\frac{9n}{44}\right) = e^{-\Omega(n)}.$$

This means that the initial search point (X_0, Y_0) satisfies $M_0 > n/4$ with probability $1 - e^{-\Omega(n)}$ or with high probability for sufficiently large n.

For condition (2), it is easy to show for RLS-type algorithms. Since RLS only moves 1 step at each iteration, then it is certainly true for $j \geq 2$. For $j = 0, 1$, we set $\eta > 0$ and $r = \eta + 1$, then condition (2) is satisfied.

For condition (1), we divide into several cases:

(a) Consider (x_t, y_t) with $X_t < Y_t$,

(a1) Consider $X_t < Y_t - 1 \Leftrightarrow X_t + 1 < Y_t$, we compute the drift:

$$\mathrm{E}_t(M_t - M_{t+1}) = \mathrm{E}_t(X_{t+1} - X_t + Y_{t+1} - Y_t)$$

Since $X_{t+1} - X_t, Y_{t+1} - Y_t \in \{-1, 0, 1\}$,

$$\begin{aligned} &= 1 \cdot \frac{n - X_t}{2n} + (-1) \cdot \frac{X_t}{2n} \\ &\quad + 1 \cdot \frac{n - Y_t}{2n} + (-1) \cdot \frac{Y_t}{2n} \\ &= \frac{4n - 2X_t - 2Y_t - 2n}{2n} \end{aligned}$$

Recall that $M_t = 2n - X_t - Y_t$ and assume $M_t \leq n/4$.

$$= \frac{M_t}{n} - 1 \leq \frac{1}{4} - 1 = -\frac{3}{4}.$$

(a2) Consider $X_t = Y_t - 1 \Leftrightarrow X_t + 1 = Y_t$, we compute the drift:

$$\mathrm{E}_t(M_t - M_{t+1}) = \mathrm{E}_t(X_{t+1} - X_t + Y_{t+1} - Y_t)$$

Since in this case, from Lemma 6.2.1, the search point can only move upwards, left-hand sided and right-hand sided. So

$$\begin{aligned} &= 1 \cdot \frac{n - X_t}{2n} + (-1) \cdot \frac{X_t}{2n} + \frac{n - Y_t}{2n} \\ &= (\frac{1}{2n} + \frac{1}{4n})M_t + \frac{1}{4n} - \frac{1}{2} \\ &= \frac{3}{4n}M_t + \frac{1}{4n} - \frac{1}{2} \end{aligned}$$

Using $M_t \leq n/4$ gives

$$\leq \frac{3}{16} - \frac{1}{2} + \frac{1}{4n} = \frac{-5}{16} + \frac{1}{4n} \leq -\frac{1}{16}.$$

(b) Consider (x_t, y_t) with $X_t = Y_t$, we compute the drift:

$$\mathrm{E}_t(M_t - M_{t+1}) = \mathrm{E}_t(X_{t+1} - X_t + Y_{t+1} - Y_t)$$

Since in this case, the search point can only move upwards, downwards and right-hand sided. So

$$= 1 \cdot \frac{n - X_t}{2n} + 1 \cdot \frac{n - Y_t}{2n} + (-1) \cdot \frac{Y_t}{2n}$$

Using $M_t = 2b - X_t - Y_t = 2n - 2Y_t \Leftrightarrow Y_t = \frac{2n - M_t}{2} \Leftrightarrow X_t = \frac{2n - M_t}{2}$,

$$= \frac{3}{4n}M_t - 1 \leq \frac{3}{4n} \cdot \frac{n}{4} - \frac{1}{2} = -\frac{5}{16}.$$

(c) Consider (x_t, y_t) with $X_t > Y_t$, by symmetry of DIAGONAL Game (see the detail analysis in case (a)), we can easily derive that

$$\begin{aligned} \mathrm{E}_t(M_t - M_{t+1}) &= \mathrm{E}_t(X_{t+1} - X_t + Y_{t+1} - Y_t) \\ &\leq \max\{\frac{-3}{4}, \frac{-1}{16}\} = \frac{-1}{16}. \end{aligned}$$

Together, we have the negative drift with:

$$\mathrm{E}(M_t - M_{t+1}|\mathcal{F}_t) \leq -\delta$$

where $-\delta = \max\{-3/4, -1/16, -5/16\} = -1/16$.

Then, by applying Theorem 2.8.2, there exists some constant $c > 0$ such that

$$\Pr(T \leq e^{cn}) \leq \Pr(T_\varepsilon \leq e^{cn}) \leq e^{-cn} = e^{-\Omega(n)}.$$

□

Theorem 6.4.2. *Consider (1+1)-CoEA with pairwise dominance relation on* DIAGONAL*, constant mutation parameter $\chi \in (0, 1]$, and problem size $n \in \mathbb{N}$. The runtime of (1+1)-CoEA for finding the maximin optimum of* DIAGONAL *is $e^{\Omega(n)}$ with probability $1 - e^{-\Omega(n)}$.*

Proof of Theorem 6.4.2. Suppose we have search point $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ in iteration $t \in \mathbb{N}$ and define $M_t := 2n - X_t - Y_t$ where $X_t = |x_t|_1$ and $Y_t = |y_t|_1$. Define $T := \inf\{t \geq 0 \mid M_t = 0\}$ and $T_\varepsilon := \inf\{t \geq 0 \mid M_t \leq \varepsilon n\}$ for any $\varepsilon \in (0, 1/20]$. Note that $T \geq T_\varepsilon$. Next, we want to show $\Pr(T_\varepsilon \leq e^{cn}) \leq e^{-cn}$ for some constant $c > 0$.

We follow the setting and notations of Lemma 6.4.1. Instead of considering the algorithm exactly reaches the optimum in this game, we first consider the algorithm reaches a ε-approximation to the optimum where $\varepsilon > 0$ satisfies the condition in proof of Lemma 6.4.2. Assume that $\varepsilon n < M_t \leq 2\varepsilon n \leq n/4$ at iteration $t \in \mathbb{N}$, then we define $T_\varepsilon := \inf\{t \geq 0 \mid M_t \leq \varepsilon n\}$. Recall from the definitions that $T_0 \geq T_\varepsilon$. Then, to derive an exponential lower bound for T, it is sufficient to derive the counterpart for T_ε. Then, by applying Theorem 2.8.2,

$$\Pr(T \leq e^{cn}) \leq \Pr(T_\varepsilon \leq e^{cn}) \leq e^{-cn} = e^{-\Omega(n)}.$$

□

Lemma 6.4.1. *Consider (1+1)-CoEA with pairwise dominance relation on* DIAGONAL *and a constant mutation rate $\chi = O(1)$. Suppose we have a search point $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ in iteration $t \in \mathbb{N}$ and define $M_t := 2n - (X_t + Y_t)$. For any t, if $\varepsilon n < M_t \leq n/4$ for some constant $\varepsilon > 0$, then there exist some $r, \eta > 0$ s.t. for any $j \geq 0$, $\Pr(|M_t - M_{t+1}| \geq j \mid \mathcal{F}_t) \leq r/(1+\eta)^j$.*

Proof of Lemma 6.4.1. By the triangle inequality, if we have $j \leq |M_t - M_{t+1}|$, then

$$\begin{aligned} j \leq |M_t - M_{t+1}| &= |X_{t+1} - X_t + Y_{t+1} - Y_t| \\ &\leq |X_{t+1} - X_t| + |Y_t - Y_{t+1}|. \end{aligned}$$

We proceed by using contradiction. Assume that both $|X_{t+1} - X_t| < \frac{j}{2}$ and $|Y_t - Y_{t+1}| < \frac{j}{2}$, then we have $|M_t - M_{t+1}| < j$ which leads to a contradiction. So by contradiction, we have at least either $|X_{t+1} - X_t|$ or $|Y_t - Y_{t+1}|$ greater or equal to $j/2$. To obtain the upper bound of change in the number of 1 bits, we consider the algorithm selects $j/2$ bit positions among n bits. Without loss of generality, for any $j \geq 0$,

$$\Pr_t\left(|X_{t+1} - X_t| \geq \frac{j}{2}\right) \leq \binom{n}{j/2}(\frac{\chi}{n})^{j/2} \leq \frac{n^{j/2}}{(j/2)!} \cdot \frac{\chi^{j/2}}{n^{j/2}}$$

Here, we use the fact that $\lim_{n\to\infty} \frac{x^n}{n!} = 0$ for any constant x, $\chi = O(1)$ and thus $k! \geq (2\chi)^k$ for sufficiently large k (e.g $k \geq 100$).

$$\leq (\frac{1}{2})^{j/2}.$$

Next, we use the union bound.

$$\begin{aligned} &\Pr_t(|M_t - M_{t+1}| \geq j) \\ \leq &\Pr_t\left(\left\{|X_{t+1} - X_t| \geq \frac{j}{2}\right\} \cup \left\{|Y_{t+1} - Y_t| \geq \frac{j}{2}\right\}\right) \\ \leq &\Pr_t\left(|X_{t+1} - X_t| \geq \frac{j}{2}\right) + \Pr_t\left(|Y_{t+1} - Y_t| \geq \frac{j}{2}\right) \\ \leq &2 \cdot (\frac{1}{2})^{j/2} = \frac{2}{(\sqrt{2} - 1 + 1)^j} \end{aligned}$$

In this case, we take $r = 2$ and $\eta = \sqrt{2} - 1 > 0$, and we get the exponential tail bound for the jump at each iteration. □

Lemma 6.4.2. *With the same setting as Theorem 6.4.2 and sufficiently large $n \geq 1$, for any t and $M_t := 2n - X_t - Y_t$, if $\varepsilon n < M_t \leq 2\varepsilon n$ for any constant $\varepsilon \in (0, 1/20]$, then there exists $\delta > 0$ such that $\mathrm{E}_t(M_t - M_{t+1}) \leq -\delta$.*

Proof of Lemma 6.4.2. As is shown in Fig 6.1, we can divide the analysis into three cases. Given that $\varepsilon n < M_t \leq 2\varepsilon n$ for any constant $\varepsilon \in (0, 1/20]$ we mainly focus on computing the drift of the ε-approximation towards the optimum. There are four regions that current search points will move based on Lemma 6.2.1. We denote the event that search points will move to each region by E_i for $i \in \{0\} \cup [3]$. In particular, we denote the event that current search points move to the upper right corner by E_0. In such an event, it contributes to positive drift into the whole drift.

(a) Consider (x_t, y_t) with $X_t = Y_t$, we compute the drift by using the law of total probability:

$$\mathrm{E}_t(M_t - M_{t+1}) = \sum_{i=0}^{3} \mathrm{E}_t(M_t - M_{t+1}; E_i)$$

Here, we underestimate the negative drift by modifying the expectation in a pessimistic way (using $\{X_{t+1} = X_t\} \cup \{Y_{t+1} = Y_t + 1\} \subseteq \cup_{i=1}^{3} E_i$). This leads to the upper bound for the overall drift by underestimating the negative drift and overestimating the positive drift.

$$
\begin{aligned}
&\leq \mathrm{E}_t(M_t - M_{t+1}; X_{t+1} = X_t) \\
&\quad + \mathrm{E}_t(M_t - M_{t+1}; Y_{t+1} = Y_t + 1) \\
&\quad + \mathrm{E}_t(M_t - M_{t+1}; E_0) \\
&= \mathrm{E}_t(Y_{t+1} - Y_t; X_{t+1} = X_t) \\
&\quad + \mathrm{E}_t(X_{t+1} - X_t + 1; Y_{t+1} = Y_t + 1) \\
&\quad + \mathrm{E}_t(X_{t+1} - X_t + Y_{t+1} - Y_t; E_0)
\end{aligned}
$$

Note that $Y_{t+1} - Y_t \sim \mathrm{Bin}(n - Y_t, \frac{\chi}{n}) - \mathrm{Bin}(Y_t, \frac{\chi}{n})$ and thus $\mathrm{E}_t(Y_{t+1} - Y_t) = (n - 2Y_t)\frac{\chi}{n}$. Since $Y_t = X_t$ and $M_t = 2n - X_t - Y_t \leq 2\varepsilon n$, $\mathrm{E}_t(Y_{t+1} - Y_t) = (n - 2Y_t)\frac{\chi}{n} = \chi\frac{M_t - n}{n} \leq (2\varepsilon - 1)\chi$ for the case that $X_{t+1} = X_t$. Next, we consider the case that $Y_{t+1} = Y_t + 1$. Note that $X_{t+1} - X_t \sim \mathrm{Bin}(n - X_t, \frac{\chi}{n}) - \mathrm{Bin}(X_t, \frac{\chi}{n})$ and thus $\mathrm{E}_t(M_t - M_{t+1}) = 1 + \mathrm{E}_t(X_{t+1} - X_t) = 1 + (2\varepsilon - 1)\chi$. Finally, we consider the event E_0 (i.e. the next search point moves to the orange and green region in Fig. 6.1). This case contributes to the positive drift. So we overestimate it by only considering both 0-bit will be flipped in x_t and y_t. Then, $\mathrm{E}_t(M_t - M_{t+1}) \leq \mathrm{E}(Z) + \mathrm{E}(W) \leq (2n - X_t - Y_t)\frac{\chi}{n} = M_t\frac{\chi}{n} \leq 2\varepsilon\chi$ where $\mathrm{Bin}(n - X_t, \frac{\chi}{n})$ stochastic dominates Z and $\mathrm{Bin}(n - Y_t, \frac{\chi}{n})$ stochastic dominates W. Overall, we obtain the following upper bound.

$$
\begin{aligned}
&\leq (2\varepsilon - 1)\chi \mathrm{E}\big(\mathbb{1}_{\{X_{t+1} = X_t\}}\big) \\
&\quad + (1 + (2\varepsilon - 1)\chi)\mathrm{E}\big(\mathbb{1}_{\{Y_{t+1} = Y_t + 1\}}\big) + 2\varepsilon\chi \mathrm{E}(\mathbb{1}_{E_0})
\end{aligned}
$$

Next, we need to derive lower bounds for the probability of each event contributing to negative drift and upper bounds for the probability of each event contributing to

positive drift (which can be bounded by using union bounds).

$$\begin{aligned}
\Pr(X_{t+1} = X_t) \geq & \Pr(\{\text{All bits remain unflipped in } x_t\}) \\
= & (1 - \frac{\chi}{n})^n \\
\geq & e^{-\chi}(1 - \frac{\chi^2}{n}) \\
\Pr(Y_{t+1} = Y_t + 1) \leq & \Pr\left(\{\text{At least one 0-bit flipped in } y_t\}\right) \\
\leq & \binom{n - Y_t}{1} \frac{\chi}{n} \\
\leq & \varepsilon\chi \\
\Pr(E_0) \leq & \Pr\left(\{\text{At least one 0-bit flipped in } x_t, y_t\}\right) \\
\leq & \binom{M_t}{1} \frac{\chi}{n} \leq 2\varepsilon\chi.
\end{aligned}$$

Together, we can provide an upper bound of the drift for constant mutation rate χ:

$$\begin{aligned}
& \mathrm{E}_t(M_t - M_{t+1}) \\
\leq & (2\varepsilon - 1)\chi e^{-\chi}(1 - \frac{\chi^2}{n}) + (1 + (2\varepsilon - 1)\chi)\varepsilon\chi + 2\varepsilon^2\chi^2 \\
= & -\chi e^{-\chi}(1 - \frac{\chi^2}{n}) \\
& + 2\varepsilon\chi \left(e^{-\chi}(1 - \frac{\chi^2}{n}) + \frac{1}{2} + \frac{2\varepsilon - 1}{2}\chi + \varepsilon\chi \right)
\end{aligned}$$

Note that $\varepsilon \in (0, \frac{1}{20}]$ implies that $\frac{2\varepsilon - 1}{2}\chi < 0$

$$\leq -\chi e^{-\chi}(1 - \frac{\chi^2}{n}) + 2\varepsilon\chi \left(e^{-\chi} + \frac{1}{2} + \frac{\chi}{20} \right)$$

Taking sufficiently large n such that $(1 - \frac{\chi^2}{n}) \geq 1/2$ gives

$$\leq -\chi e^{-\chi}\frac{1}{2} + 2\varepsilon\chi \left(e^{-\chi} + 1 \right)$$

Using $\varepsilon \in (0, \frac{1}{20}]$ gives

$$\leq \chi \left(\frac{-e^{-\chi}}{2} + \frac{e^{-\chi}}{10} + \frac{1}{10} \right)$$

Note that $\frac{-e^{-\chi}}{2} + \frac{e^{-\chi}}{10} + \frac{1}{10} = \frac{-2e^{-\chi}}{5} + \frac{1}{10} < 0$ for all $\chi \in (0, 1]$. Then we end up with a negative constant drift $-\delta_1 := \chi \left(\frac{-2e^{-\chi}}{5} + \frac{1}{10} \right) < 0$

$$\leq -\delta_1.$$

(b) Consider (x_t, y_t) with $X_t > Y_t$, we use the similar analysis. We denote the event that search points will move to each region by E_i for $i \in \{0\} \cup [3]$. In particular, we denote the event that current search points move to the upper right corner by E_0 (enclosed by orange and green area). In such an event, it contributes to positive drift into the whole drift. We assume that $\varepsilon n < M_t \leq 2\varepsilon n$. We can derive the bound for X_t, Y_t respectively by combining $M_t = 2n - X_t - Y_t$.

$$X_t < 2n - \varepsilon n - Y_t \textbf{ and } n - 2\varepsilon n \leq 2n - 2\varepsilon n - X_t \leq Y_t$$

Next, using the upper bound for X_t gives

$$X_t - Y_t < (2n - \varepsilon n - Y_t) - Y_t$$

Using the lower bound for $Y_t \geq n - 2\varepsilon n$ gives

$$\begin{aligned} &\leq 2n - \varepsilon n - 2n + 4\varepsilon n \\ &\leq 3\varepsilon n. \end{aligned}$$

Hence, we derive the difference of $X_t - Y_t$ is

$$X_t - Y_t < 3\varepsilon n. \tag{A.7}$$

Next, we compute the drift

$$\begin{aligned} &\mathrm{E}_t(M_t - M_{t+1}) \\ =&\sum_{i=0}^{3} \mathrm{E}_t(M_t - M_{t+1}; E_i) \end{aligned}$$

We neglect the negative drift induced by moving to the blue region in Fig 6.1 and only consider the negative drift in the red region. In particular, we underestimate the negative drift by modifying the expectation in a pessimistic way (using $\{X_{t+1} = X_t\} \subseteq \cup_{i=1}^{3} E_i$) and considering the negative drift only when $X_{t+1} = X_t$.

$$\leq \mathrm{E}_t(M_t - M_{t+1}; X_{t+1} = X_t) + \mathrm{E}_t(M_t - M_{t+1}; E_0)$$

Note that $\#\{\text{1-bit flipped in } y_t\} \sim \text{Bin}(Y_t, \frac{\chi}{n})$ and $\#\{\text{0-bit flipped in } y_t\} \sim \text{Bin}(n - Y_t, \frac{\chi}{n})$. We also underestimate the prob by considering only one bit flipped to simplify the computation.

$$\begin{aligned}
&\leq (-Y_t \cdot \frac{\chi}{n} + (n - Y_t) \cdot \frac{\chi}{n}) \Pr(X_{t+1} = X_t) \\
&\quad + \left((n - Y_t) \cdot \frac{\chi}{n} + (n - X_t) \cdot \frac{\chi}{n}\right) \Pr(E_0) \\
&\leq (n - 2Y_t)\frac{\chi}{n} \Pr(X_{t+1} = X_t) + (2n - X_t - Y_t)\frac{\chi}{n} \Pr(E_0)
\end{aligned}$$

Recall that $M_t = 2n - X_t - Y_t \leq 2\varepsilon n$ and thus $n - 2Y_t = M_t - n + (X_t - Y_t) \leq 2\varepsilon n - n + 3\varepsilon n = 5\varepsilon n - n$ since the second inequality follows from Eq. (A.7).

$$\leq (5\varepsilon - 1)\chi \Pr(X_{t+1} = X_t) + 2\varepsilon\chi \Pr(E_0)$$

Note that $\Pr(X_{t+1} = X_t) \geq (1 - \frac{\chi}{n})^n \geq e^{-\chi}(1 - \frac{\chi^2}{n})$ by considering all bits remain unflipped in x_t and $\Pr(E_0) \leq 1 - (1 - \frac{\chi}{n})^{2n} \leq 1 - e^{-2\chi}(1 - \frac{\chi^2}{n})^2$ since we upper bound the probability of E_0 by considering at least one bit out of the two bitstrings (i.e. x_t, y_t) flipped.

$$\begin{aligned}
&\leq (n - 2Y_t)\frac{\chi}{n} e^{-\chi}(1 - \frac{\chi^2}{n}) \\
&\quad + (2n - X_t - Y_t)\frac{\chi}{n}\left(1 - e^{-2\chi}(1 - \frac{\chi^2}{n})^2\right) \\
&\leq (5\varepsilon - 1)\chi e^{-\chi}(1 - \frac{\chi^2}{n}) + 2\varepsilon\chi\left(1 - e^{-2\chi}(1 - \frac{\chi^2}{n})^2\right)
\end{aligned}$$

For sufficiently large n, we have $1/2 \leq (1 - \frac{\chi^2}{n}) \leq 1$. Note that $5\varepsilon - 1 < 0$ since $\varepsilon \in (0, \frac{1}{20}]$.

$$\leq (5\varepsilon - 1)\chi e^{-\chi}\frac{1}{2} + 2\varepsilon\chi$$

Using $\varepsilon \in (0, \frac{1}{20}]$ gives

$$\leq \chi\left(\frac{-15}{40}e^{-\chi} + \frac{1}{10}\right)$$

It is easy to check for any $\chi \in (0, 1]$, $\left(\frac{-15}{40}e^{-\chi} + \frac{1}{10}\right) < 0$. Hence, we end up with a negative constant drift $-\delta_2 < 0$

$$= -\delta_2.$$

(c) Consider (x_t, y_t) with $X_t < Y_t$, we use the similar analysis. We denote the event that search points will move to each region by E_i for $i \in \{0\} \cup [3]$. In particular, we denote the event that current search points move to the upper right corner by E_0. Such an event contributes to positive drift into the whole drift. By symmetry of DIAGONAL Game, we omit the repetitive steps as case (b) already calculated and conclude that there exists $\delta_3 > 0$ such that

$$\mathrm{E}(M_t - M_{t+1} \mid \mathcal{F}_t) \leq -\delta_3.$$

Together, if we take $\delta = \min\{-\delta_1, -\delta_2, -\delta_3\}$, then we obtain the negative drift of $(1+1)$-CoEAs: there exists $\delta > 0$ such that,

$$\mathrm{E}(M_t - M_{t+1} \mid \mathcal{F}_t) \leq -\delta.$$

□

Lemma 6.5.1. *Given two populations P and Q in PDCoEA with population size $\lambda \in$ poly(n) and constant mutation rate $\chi \in (0, \ln 2/2)$ on* DIAGONAL *with problem size $n \in \mathbb{N}$, assume PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$. For the current level $j \geq n/2$, there exists $z_{n/2}, \ldots, z_{n-1} \in (0,1)$ such that for $(x, y) \sim \mathcal{D}(P, Q)$,*

$$\Pr(x \in U_{j+1}) \cdot \Pr(y \in V_{j+1}) \geq z_j.$$

Proof of Lemma 6.5.1. To obtain the upgrade probability, we estimate it pessimistically. Suppose the population stays at some current level $j \geq n/2$, recall the definition of partitions A_j, B_j: given $(A_0 \times B_0), \ldots, (A_n \times B_n)$, we define $A_0 = \mathcal{X}$ and $B_0 = \mathcal{Y}$ and for $j \in \{n/2, \ldots, n\}$,

$$\begin{aligned} A_j &:= \{x \in \{0,1\}^n \mid |x|_1 = j\} \\ B_j &:= \{y \in \{0,1\}^n \mid |y|_1 = j\}. \end{aligned} \tag{A.8}$$

So, to upgrade to either A_{j+1} or B_{j+1}, the algorithm only needs to flip one single zero bit in each binary bitstring before reaching the final level. We lower bound the probability pessimistically by assuming only 1 0-bit flipped to 1-bit and other bits remain the same. Thus,

$$\Pr(x \in A_{j+1}) \cdot \Pr(y \in B_{j+1}) \geq \left(\frac{\chi}{n}(1-\frac{\chi}{n})^{n-1}\right)^2$$

Using $(1-\frac{\chi}{n})^{n-1} \geq e^{-\chi}(1-o(1))$ gives

$$\geq \left(\frac{\chi}{n}e^{-\chi}(1-o(1))\right)^2 := z_j.$$

□

Lemma 6.5.2. *Given two populations P and Q in PDCoEA with population size $\lambda \in$ poly(n) and a sufficiently small constant mutation rate $\chi \in (0,1)$ on* DIAGONAL *with problem size n, assume (A | G2a) holds for a constant $\gamma_0 \geq \left(-1+\sqrt{4\alpha(1+\delta)+1}\right)/2\alpha$ where $\delta \in (0,\alpha)$. For all $\gamma \in (0,\gamma_0)$, if $|(P\times Q)\cap(U_{j+1}\times V_{j+1})| \geq \gamma\lambda^2$, then for a sample $(x,y) \sim \mathcal{D}(P,Q)$, there exists a constant $\delta_0 > 0$ such that for all current levels $j \in [n/2, n]$, $\Pr(x \in U_{j+1})\Pr(y \in V_{j+1}) \geq (1+\delta_0)\gamma$.*

Proof of Lemma 6.5.2. Let us denote the probability of selecting the pixel $j \in [n]$ by u_1 for prey x and v_1 for predator y and denote the probability of selecting the pixel $j+1 \in [n]$ by u_2 for prey x and v_2 for predator y. If $|(P\times Q)\cap(U_{j+1}\times V_{j+1})| \geq \gamma\lambda^2$, then we have

$$(u_1+u_2)\cdot v_2 = \frac{|\,(P\times Q)\cap(U_{j+1}\times V_{j+1})\,|}{\lambda^2} \geq \gamma.$$

Recall that "current level" in Theorem 6.3.1 means that $j := \max\{i \in [n] \mid |(P\times Q)\cap(U_i \times V_i)| \geq \gamma_0\lambda^2\}$. Then, we have

$$u_1\cdot(v_1+v_2) = \frac{|(P\times Q)\cap(U_j\times V_j)|}{\lambda^2} \geq \gamma_0.$$

Now, we consider SELECT(P,Q) only. Given two uniformly sampled pairs $(x_1,y_1),(x_2,y_2) \sim \text{Unif}(\{0,1\}^n \times \{0,1\}^n)$, there are two main cases in which we select either x or y in the next level $j+1$:

(1) $(x_1, y_1) \succeq (x_2, y_2)$ and $x_1 \in U_{j+1}$ (or $(x_1, y_1) \succeq (x_2, y_2)$ and $y_1 \in V_{j+1}$);

(2) $(x_1, y_1) \not\succeq (x_2, y_2)$ and $x_2 \in U_{j+1}$ (or $(x_1, y_1) \not\succeq (x_2, y_2)$ and $y_2 \in V_{j+1}$).

We check all the possible combinations for $X_1, Y_1, X_2, Y_2 \in \{j, j+1\}$ (we use capital letters to denote the number of 1 bits in each bit-string) which satisfy (1) and (2) above. Now, we can compute the following selected probability (without mutation for now). We trivially have

$$\begin{aligned}
\Pr_{(x,y)\sim\textsc{Select}(P,Q)}(x \in U_{j+1}) &\geq \Pr_{x_1,x_2\sim\mathrm{Unif}(P)}(x_1, x_2 \in A_{j+1}) \\
&= (u_1 + u_2)^2 \\
&\geq (u_1 + u_2)u_1 \\
&\geq (u_1 + u_2)\gamma_0,
\end{aligned}$$

where the last inequality follows because $u_1(v_1 + v_2) \geq \gamma_0$.

$$\begin{aligned}
\Pr_{(x,y)\sim\textsc{Select}(P,Q)}(y \in V_{j+1}) = v_2\Big[&(u_2 + u_3)^2(v_0 + v_1 + v_2) + u_3(2u_2 + r_2u_3)v_3 \\
&+ u_0^2(v_0 + r_1v_0 + v_1 + v_2 + v_3) + u_1^2(2v_0 + 2v_1 + v_2 + v_3) \\
&+ 2u_1\big((u_2 + u_3)(v_0 + v_1 + v_2) + u_3v_3\big) \\
&+ 2u_0\big((u_2 + u_3)(v_0 + v_1 + v_2) + u_3v_3 + u_1(2v_0 + 2v_1 + v_2 + v_3)\big)\Big]. \\
\geq v_2\Big(&v_0\left(1 + 2u_0u_1 + u_1^2\right) + v_1\left(1 + 2u_0u_1 + u_1^2\right) + v_2 \\
&+v_3\left((u_0 + u_1)^2 + 2\left(u_0 + u_1 + u_2\right)u_3\right)\Big).
\end{aligned}$$

Using $(u_0 + u_1 + u_2 + u_3)^2 = (u_0 + u_1)^2 + 2\left(u_0 + u_1 + u_2\right)u_3 + 2u_0u_2 + 2u_1u_2 + u2^2 + u3^2$ and $v_0 + v_1 + v_2 + v_3 = 1$ gives

$$= v_2\left(1 + (v_0 + v_1)\left(2u_0u_1 + u_1^2\right) - v_3\left(2u_0u_2 + 2u_1u_2 + u_2^2 + u_3^2\right)\right).$$

Using the assumption (A | G2a) gives $v_0 + v_1 \geq \alpha$, and also note that $u_1 \geq \gamma_0/(v_1 + v_2) \geq \gamma_0$ and $v_3 \leq 1 - (v_1 + v_2) \leq 1 - \gamma_0$.

$$\begin{aligned}
&\geq v_2\left(1 + \alpha\gamma_0^2 - (1 - \gamma_0)\right) \\
&= v_2\left(\alpha\gamma_0^2 + \gamma_0\right).
\end{aligned}$$

By the inequality $\alpha\gamma_0^2+\gamma_0 \geq 1+\delta$ for γ_0 gives $0 < \delta < \alpha$ and $\left(-1+\sqrt{4\alpha(1+\delta)+1}\right)/2\alpha \leq \gamma_0 < 1$. By choosing $\gamma_0 \geq \left(-1+\sqrt{4\alpha(1+\delta)+1}\right)/2\alpha$, we obtain

$$\geq v_2\,(1+\delta)$$

We underestimate the mutation probabilities, assuming $\chi \in (0,1)$

$$\Pr_{(x,y)\sim\textsc{Mutate}(P,Q)}(x \in U_{j+1}) \geq \left(1-\frac{\chi}{n}\right)^n > \left(1-\frac{\chi}{n}\right)^{(n/\chi-1)\chi}\left(1-\frac{\chi}{n}\right) > e^{-\chi}\left(1-\frac{1}{n}\right);$$
$$\Pr_{(x,y)\sim\textsc{Mutate}(P,Q)}(y \in V_{j+1}) > e^{-\chi}\left(1-\frac{1}{n}\right)$$

Together, we obtain

$$\Pr_{(x,y)\sim\mathcal{D}(P,Q)}(x \in U_{j+1}) \geq e^{-\chi}\left(1-\frac{1}{n}\right)(u_1+u_2)\,\gamma_0;$$
$$\Pr_{(x,y)\sim\mathcal{D}(P,Q)}(y \in V_{j+1}) \geq e^{-\chi}\left(1-\frac{1}{n}\right)v_2\,(1+\delta)\,.$$

Taking the product of two probabilities gives

$$\Pr(x \in U_{j+1})\Pr(y \in V_{j+1}) \geq e^{-2\chi}\left(1-\frac{1}{n}\right)(u_1+u_2)\,v_2\;\gamma_0\,(1+\delta)$$

Recall that $(u_1+u_2)\cdot v_2 \geq \gamma$. Then, we have

$$\geq \gamma e^{-2\chi}\left(1-\frac{1}{n}\right)(1+\delta)\,\gamma_0$$

choosing a sufficiently small mutation rate χ such that $e^{2\chi} \leq 1+c/2$ and thus there exists $\delta_0 > 0$ such that

$$\geq \gamma\,(1+\delta_0)\,.$$

□

Lemma 6.5.3. *Let $\mathcal{D}$ be the operator associated to Algorithms 15 (PDCoEA) on* Diagonal. *Let P and Q be populations of PDCoEA at current level $j \in [n/2, n]$ of* Diagonal *with respect to the partitions U, V and $\gamma_0 \in (0,1)$. There exists a constant $\delta_0 > 0$ and sufficiently small mutation rate χ, such that if for some constant $\delta > 0$, (A | G2b):*

$$\Pr_{x\sim\mathrm{Unif}(P)}(x \in A_{<j})\left(\Pr_{y\sim\mathrm{Unif}(Q)}(y \in B_{<j}\cup B_j)^2+1\right) \geq \frac{1-\gamma_0^2+\delta}{\gamma_0} \text{ holds,}$$

then for $(x,y) \sim \mathcal{D}(P,Q)$, $\Pr(x \in U_j)\Pr(y \in V_j) \geq (1+\delta_0)\gamma_0$.

Proof of Lemma 6.5.3. Let the probabilities $u_0, u_1, u_2, u_3, v_0, v_1, v_2, v_3$ be as given in Figure A.4. Recall that "current level" in Theorem 6.3.1 means that $j := \max\{i \in [n] \mid |(P \times Q) \cap (U_i \times V_i)| \geq \gamma_0 \lambda^2\}$. It then follows that

$$u_1 \cdot (v_1 + v_2) = \frac{|(P \times Q) \cap (U_j \times V_j)|}{\lambda^2} \geq \gamma_0. \tag{A.9}$$

Now, we consider $\textsc{Select}(P, Q)$ only. Given two uniformly sampled pairs $(x_1, y_1), (x_2, y_2) \sim \mathrm{Unif}(\{0,1\}^n \times \{0,1\}^n)$, there are two main cases in which we select either x or y in the same level j:

(1) $(x_1, y_1) \succeq (x_2, y_2)$ and $x_1 \in U_j$ (or $(x_1, y_1) \succeq (x_2, y_2)$ and $y_1 \in V_j$);

(2) $(x_1, y_1) \not\succeq (x_2, y_2)$ and $x_2 \in U_j$ (or $(x_1, y_1) \not\succeq (x_2, y_2)$ and $y_2 \in V_j$).

Now, we can compute the following selected probability (without mutation for now).

$$\begin{aligned}\Pr_{(x,y)\sim\textsc{Select}(P,Q)}(x \in U_j) &= u_1\left(u_0\left(1 + (v_0+v_1)^2\right) + u_1 + (u_2+u_3)\left(v_0+v_1+v_3\right)^2\right) \\ &\geq u_1\left(u_0\left(1 + (v_0+v_1)^2\right) + u_1\right)\end{aligned}$$

using $u_1 \geq \gamma_0/(v_1 + v2)$

$$\geq u_1\left(u_0\left(1 + (v_0+v_1)^2\right) + \frac{\gamma_0}{v_1+v_2}\right)$$

To compute the probability of selecting y in B_j, we use the trivial lower bound

$$\Pr_{(x,y)\sim\textsc{Select}(P,Q)}(y \in V_j) \geq (v_1+v_2)^2.$$

Together, we have

$$\Pr_{(x,y)\sim\textsc{Select}(P,Q)}(x \in U_j) \cdot \Pr_{(x,y)\sim\textsc{Select}(P,Q)}(y \in V_j) \geq u_1(v_1+v_2)^2\left(u_0\left(1 + (v_0+v_1)^2\right) + \frac{\gamma_0}{v_1+v_2}\right)$$

Applying assumption (A | G2b) gives

$$\geq u_1(v_1+v_2)^2\left(\frac{1-\gamma_0^2+\delta}{\gamma_0} + \frac{\gamma_0}{v_1+v_2}\right)$$

By assumption $u_1 (v_1 + v_2) \geq \gamma_0$ and therefore $v_1 + v_2 \geq \gamma_0$. Thus, we have

$$\begin{aligned} &\geq \gamma_0 \left(1 - \gamma_0^2 + \delta + \gamma_0\right) \\ &> \gamma_0 (1 + \delta) . \end{aligned}$$

We underestimate the mutation probability by using Bernoulli's inequality

$$\begin{aligned} \Pr\left(\textsc{Mutate}(x) \in U_j \mid x \in U_j\right) &\geq \left(1 - \frac{\chi}{n}\right)^n \geq 1 - \chi \\ \Pr\left(\textsc{Mutate}(y) \in V_j \mid y \in V_j\right) &\geq \left(1 - \frac{\chi}{n}\right)^n \geq 1 - \chi. \end{aligned}$$

Together, we obtain

$$\begin{aligned} \Pr(x \in U_j) \Pr(y \in V_j) \geq &\Pr_{(x,y) \sim \textsc{Select}(P,Q)} (x \in U_j) \Pr_{(x,y) \sim \textsc{Select}(P,Q)} (y \in V_j) \times \\ &\Pr\left(\textsc{Mutate}(x) \in U_j \mid x \in U_j\right) \Pr\left(\textsc{Mutate}(y) \in V_j \mid y \in V_j\right) \\ &\geq \gamma_0 (1 - \chi)^2 (1 + \delta) \geq \gamma_0 (1 + \delta_0) \end{aligned}$$

where the last inequality follows by choosing sufficiently small constants $\chi, \delta_0 > 0$. □

Lemma 6.5.4. *Suppose PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$ with the runtime of Algorithm 15 finding the optimum $T \in \mathbb{N}$ and $(x_t, y_t) \sim \mathcal{D}(P_t, Q_t)$ at iteration $t \leq T$. With the same setting of Lemma 6.5.3, there exists $1 \geq \gamma_0 \geq \sqrt{(1 + \delta) / (\chi e^{-\chi}/7 + 1)}$ such that (A | G2b) holds for some constant $\delta \in (0, \chi e^{-\chi}/7)$ with probability $1 - e^{-\Omega(n)}$.*

Proof of Lemma 6.5.4. Using Lemma A.7.5 with respect to $A_{<j}, B_{<j}$ and B_j defined in Lemma A.7.5 gives $u_0 \geq \chi e^{-\chi} \gamma_0 / 7$ with probability $1 - e^{-\Omega(n)}$. A rough lower bound can be derived for $u_0 \left((v_0 + v_1)^2 + 1\right) \geq u_0$. Note that

$$\gamma_0 \geq \sqrt{\frac{1 + \delta}{1 + \chi e^{-\chi}/7}}$$

Rearranging it gives

$$\left(1 + \frac{\chi e^{-\chi}}{7}\right)\gamma_0^2 \geq 1 + \delta$$

Thus, we have

$$u_0\left((v_0 + v_1)^2 + 1\right) \geq u_0 \geq \frac{\chi e^{-\chi}\gamma_0}{7} \geq \frac{1 - \gamma_0^2 + \delta}{\gamma_0}$$

Thus, we note (A) holds with probability $1 - e^{-\Omega(n)}$. □

Lemma 6.5.5. *Given two populations from PDCoEA, P and Q with population size $c \log n \leq \lambda \in poly(n)$ where $c > 0$ is some constant and a constant mutation rate χ on* Diagonal *with problem size n, assume constant $\gamma_0 \geq \sqrt{(1+\delta)/\left(\chi e^{-\chi}/7 + 1\right)}$ with constant $\delta \in (0, \chi e^{-\chi}/7)$ and sufficiently large n. Suppose PDCoEA initialises P_0, Q_0 at $|P_0(i)|_1 = |Q_0(i)|_1 = n/2$ for all $i \in [\lambda]$ with the runtime of Algorithm 15 finding the optimum $T \in \mathbb{N}$ and $(x_t, y_t) \sim \mathcal{D}(P_t, Q_t)$ at iteration $t \leq T$. There exists a constant $\delta > 0$ such that for all current levels $j \in [n/2, n]$, $\Pr(x_t \in U_j)\Pr(y_t \in V_j) \geq (1+\delta)\gamma_0$.*

Proof of Lemma 6.5.5. By combining Lemma 6.5.3 and Lemma 6.5.4, we obtain the lower bound $\Pr(x_t \in U_j \mid (A \mid G2b))\Pr(y_t \in V_j \mid (A \mid G2b)) \geq (1+\delta)\gamma_0$ with high probability. Using the law of total probability, we have

$$\begin{aligned}\Pr(x_t \in U_j)\Pr(y_t \in V_j) &\geq \Pr(x_t \in A_j \mid (A \mid G2b)) \cdot \Pr\left((A \mid G2b)\right) \\ &\quad \times \Pr(y_t \in B_j \mid (A \mid G2b)) \cdot \Pr\left((A \mid G2b)\right) \\ &\geq (1+\delta)\gamma_0 \cdot \left(1 - e^{-\Omega(n)}\right)^2\end{aligned}$$

For sufficiently large n, $\left(1 - e^{-\Omega(n)}\right)^2 = 1 - o(1)$.

$$= (1 - o(1))\,(1+\delta)\gamma_0$$

□

A.7.6 More Empirical Results

We defer more experiment data here. First, we present the heatmaps for the empirical mean of runtimes for all CoEAs on DIAGONAL with $n = 100$. Next, we present the detailed statistics corresponding to Figure 6.2 including the tables for the empirical mean of runtimes (Tables A.2,A.4,A.6), p-value from Wilcoxon rank-sum tests for runtimes (Tables A.3,A.5,A.7). Finally, we consider the fixed mutation rate $\chi = 0.3$ and $\lambda = c \log n$ where $c = 10$-60 under problem sizes $n = 100$-1000, providing the tables for the empirical mean of runtimes (Table A.8) and boxplots for runtime distribution (Figure A.8).

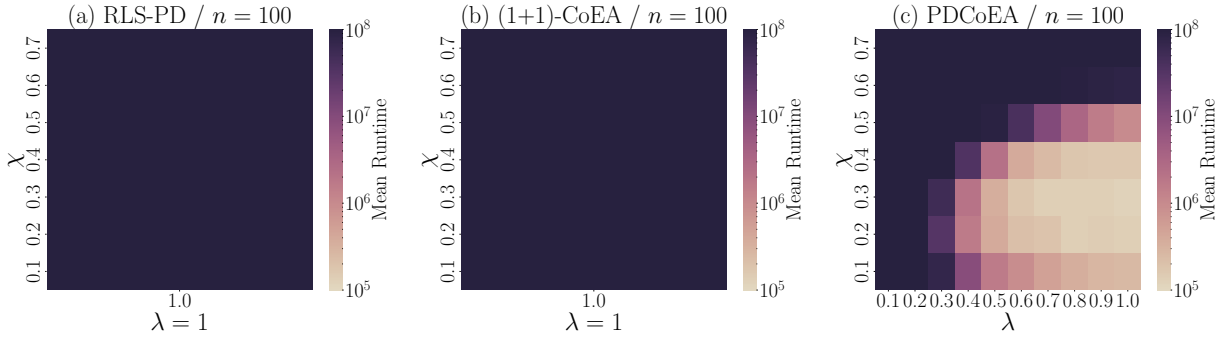


Figure A.5: Heatmaps for the runtime of all CoEAs on Diagonal with problem size $n = 100$ against different mutation rates. Each pixel in the plot represents the empirical mean of runtime among 100 independent runs with respect to different mutation rates and population sizes. In particular, since RLS-PD and (1+1)-CoEA only consist of 1 offspring, there is no extra parameter for population size. For all CoEAs, the mutation rate χ in the vertical axis ranges from 0.1 to 0.7, For PDCoEA, the population size λ in the horizontal axis ranges from $0.1n$ to n.

Figure A.5 represents the empirical mean of runtime for CoEAs with various possible configurations. It is worth noting that RLS-PD and (1+1)-CoEA reach the function evaluation budgets with respect to different mutation rates. However, under certain mutation rates and population sizes, PDCoEA can find the optimum efficiently. These empirical findings match our theoretical analysis. Hence, the rest of the experiments only focus on

the empirical results of PDCoEA.

In the second experiment, we explore the best possible configurations of χ and λ for PDCoEA with problem sizes $n = 100, 500, 1000$. For Wilcoxon rank-sum tests, we consider the null hypothesis that runtimes from two different configurations are drawn from the same distribution, with the alternative hypothesis being that runtimes in one configuration are more likely to be larger than in the other.

Table A.2: Empirical mean runtimes of PDCoEA for different values of χ and λ_c where $\lambda = \lambda_c \cdot n$ on DIAGONAL with problem size $n = 100$, expressed as multiples of 10^6. For each configuration, we conduct 100 independent runs. The function evaluation for each run is 10^8. Bold texts denote the best runtime with respect to the same mutation rate χ, and bold & underline texts denote the best runtime with respect to various combinations of χ and λ.

$\chi \backslash \lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	100.00	100.00	71.42	9.29	1.68	0.92	0.54	0.37	0.29	**0.27**
0.2	100.00	100.00	31.23	1.69	0.41	0.22	0.20	**0.14**	0.15	**0.14**
0.3	100.00	100.00	52.90	2.10	0.36	0.19	0.14	0.13	0.14	**<u>0.12</u>**
0.4	100.00	100.00	99.74	34.37	2.28	0.41	0.25	0.19	**0.16**	0.17
0.5	100.00	100.00	100.00	99.08	89.79	40.88	10.49	3.26	1.61	**1.03**
0.6	100.00	100.00	100.00	100.00	100.00	99.97	99.01	95.92	85.59	**81.96**
0.7	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table A.2 presents the empirical mean runtimes (scaled by 10^{-6}) of the PDCoEA algorithm for different combinations of mutation rates (χ) and population size constants (λ_c), with a problem size of $n = 100$. The best runtime, highlighted in bold and underlined, is $\chi = 0.3$ and $\lambda_c = 1.0$ with a value of 0.12×10^6. Other best runtimes for specific mutation rates are also bolded.

Table A.3: Wilcoxon rank-sum test: p-values for different values of χ and λ_c on Diagonal with problem size $n = 100$. Each p-value compares runtimes against the combination of $\chi = 0.3$ and $\lambda_c = 1.0$. Bold texts denote the combinations that have statistically significant differences against the runtime under the configuration of $\chi = 0.3$ and $\lambda_c = 1.0$. The p-value of less than 0.05 indicates that this test rejects the hypothesis at the 5% significance level.

$\chi \backslash \lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.60 \times 10^{-34}}$	$\mathbf{3.47 \times 10^{-33}}$	$\mathbf{8.11 \times 10^{-30}}$	$\mathbf{2.85 \times 10^{-21}}$	$\mathbf{3.87 \times 10^{-22}}$	$\mathbf{4.17 \times 10^{-18}}$	$\mathbf{7.89 \times 10^{-16}}$	$\mathbf{4.99 \times 10^{-21}}$
0.2	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.76 \times 10^{-34}}$	$\mathbf{1.46 \times 10^{-30}}$	$\mathbf{9.26 \times 10^{-16}}$	$\mathbf{2.97 \times 10^{-5}}$	$\mathbf{2.87 \times 10^{-2}}$	0.721	0.774	0.152
0.3	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.60 \times 10^{-34}}$	$\mathbf{3.84 \times 10^{-31}}$	$\mathbf{2.60 \times 10^{-12}}$	$\mathbf{8.13 \times 10^{-6}}$	0.596	0.207	0.076	-
0.4	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{8.39 \times 10^{-33}}$	$\mathbf{2.87 \times 10^{-20}}$	$\mathbf{1.91 \times 10^{-19}}$	$\mathbf{2.51 \times 10^{-11}}$	$\mathbf{6.10 \times 10^{-13}}$	$\mathbf{1.48 \times 10^{-15}}$
0.5	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.04 \times 10^{-33}}$	$\mathbf{1.82 \times 10^{-31}}$
0.6	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.7	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$

Table A.3 shows the Wilcoxon rank-sum test p-values comparing runtimes against the configuration of $\chi = 0.3$ and $\lambda_c = 1.0$. A p-value less than 0.05 indicates a statistically significant difference at the 5% significance level. The highlighted bold p-values indicate no significant difference, with $\chi = 0.2, \lambda_c = 0.8$ (0.721), $\chi = 0.2, \lambda_c = 0.9$ (0.774), $\chi = 0.3, \lambda_c = 0.7$ (0.596), and a few other values showing non-significance. These might suggest the best possible mutation rates range from $\chi \in [0.2, 0.3]$ and the best population size range from $\lambda \in [0.7n, n]$ under problem size $n = 100$.

Table A.4 presents the empirical mean runtimes of the PDCoEA algorithm for different combinations of mutation rates (χ) and population size (λ), with a problem size of $n = 500$. The best runtime, highlighted in bold and underlined, is $\chi = 0.3$ and $\lambda_c = 0.3$ with a value of 1.05×10^6. Other best runtimes for specific mutation rates are also bolded.

Table A.5 shows the Wilcoxon rank-sum test p-values comparing runtimes against the configuration of $\chi = 0.3$ and $\lambda_c = 0.3$. A p-value less than 0.05 indicates a statistically significant difference at the 5% significance level. All other configurations generally show

Table A.4: Empirical mean runtimes of PDCoEA for different values of χ and λ_c where $\lambda = \lambda_c \cdot n$ on DIAGONAL with problem size $n = 500$, expressed as multiples of 10^6. For each configuration, we conduct 100 independent runs. The function evaluation for each run is 10^9. Bold texts denote the best runtime with respect to the same mutation rate χ, and bold & underline texts denote the best runtime with respect to various combinations of χ and λ.

$\chi \backslash \lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1000.00	260.22	5.37	2.41	2.21	**2.17**	2.32	2.61	2.64	2.79
0.2	1000.00	2.68	**1.23**	1.30	1.33	1.60	1.69	1.88	1.98	2.14
0.3	1000.00	1.37	**<u>1.05</u>**	1.20	1.31	1.45	1.63	1.82	1.93	2.11
0.4	1000.00	68.72	1.55	**1.42**	1.54	1.70	1.87	2.03	2.18	2.36
0.5	1000.00	1000.00	1000.00	1000.00	1000.00	872.11	355.60	126.04	41.86	**19.78**
0.6	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
0.7	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00

highly significant differences, with p-values much lower than 0.05, indicating rejecting the null hypothesis at the 5% level. These might suggest the best possible mutation rates $\chi = 0.3$ and the best population size $\lambda = 0.3n$ under problem size $n = 500$.

Table A.6 presents the empirical mean runtimes of the PDCoEA algorithm for different combinations of mutation rates (χ) and population size (λ), with a problem size of $n = 1000$. The best runtime, highlighted in bold and underlined, is $\chi = 0.3$ and $\lambda_c = 0.2$ with a value of 2.71×10^6. Other best runtimes for specific mutation rates are also bolded.

Table A.7 shows the Wilcoxon rank-sum test p-values comparing runtimes against the configuration of $\chi = 0.3$ and $\lambda_c = 0.2$. A p-value less than 0.05 indicates a statistically significant difference at the 5% significance level. All other configurations generally show highly significant differences, with p-values much lower than 0.05, indicating rejecting the

Table A.5: Wilcoxon rank-sum test: p-values for different values of χ and λ_c on Diagonal with problem size $n = 500$. Each p-value compares runtimes against the combination of $\chi = 0.3$ and $\lambda_c = 0.3$. Bold texts denote the combinations that have statistically significant differences against the runtime under the configuration of $\chi = 0.3$ and $\lambda_c = 0.3$. The p-value of less than 0.05 indicates that this test rejects the hypothesis at the 5% significance level.

$\chi\backslash\lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.93 \times 10^{-34}}$	$\mathbf{2.15 \times 10^{-29}}$	$\mathbf{4.03 \times 10^{-28}}$	$\mathbf{2.99 \times 10^{-29}}$	$\mathbf{1.16 \times 10^{-29}}$	$\mathbf{3.13 \times 10^{-30}}$	$\mathbf{6.46 \times 10^{-32}}$	$\mathbf{2.78 \times 10^{-32}}$	$\mathbf{7.46 \times 10^{-33}}$
0.2	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.28 \times 10^{-17}}$	$\mathbf{2.55 \times 10^{-5}}$	$\mathbf{7.51 \times 10^{-16}}$	$\mathbf{1.27 \times 10^{-23}}$	$\mathbf{1.55 \times 10^{-27}}$	$\mathbf{2.10 \times 10^{-28}}$	$\mathbf{1.56 \times 10^{-28}}$	$\mathbf{7.46 \times 10^{-30}}$	$\mathbf{1.18 \times 10^{-31}}$
0.3	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{5.76 \times 10^{-4}}$	-	$\mathbf{1.93 \times 10^{-15}}$	$\mathbf{2.08 \times 10^{-23}}$	$\mathbf{6.03 \times 10^{-27}}$	$\mathbf{3.33 \times 10^{-28}}$	$\mathbf{2.28 \times 10^{-28}}$	$\mathbf{2.68 \times 10^{-29}}$	$\mathbf{1.25 \times 10^{-31}}$
0.4	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{3.41 \times 10^{-34}}$	$\mathbf{2.44 \times 10^{-25}}$	$\mathbf{7.85 \times 10^{-26}}$	$\mathbf{9.81 \times 10^{-28}}$	$\mathbf{2.34 \times 10^{-28}}$	$\mathbf{5.80 \times 10^{-29}}$	$\mathbf{1.16 \times 10^{-30}}$	$\mathbf{1.12 \times 10^{-31}}$	$\mathbf{4.30 \times 10^{-32}}$
0.5	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.6	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.7	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$

Table A.6: Empirical mean runtimes of PDCoEA for different values of χ and λ_c where $\lambda = \lambda_c \cdot n$ on DIAGONAL with problem size $n = 1000$, expressed as multiples of 10^6. For each configuration, we conduct 100 independent runs. The function evaluation for each run is 10^9. Bold texts denote the best runtime with respect to the same mutation rate χ, and bold & underline texts denote the best runtime with respect to various combinations of χ and λ.

$\chi\backslash\lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	1000.00	7.51	**5.40**	6.52	6.78	7.44	8.16	8.92	9.64	10.38
0.2	39.85	**2.86**	3.70	4.36	5.00	5.68	6.38	6.70	7.36	8.06
0.3	7.27	**2.71**	3.17	3.83	4.44	5.12	5.96	6.57	7.23	7.99
0.4	1000.00	3.38	**3.77**	4.43	5.12	5.89	6.55	7.27	8.03	8.76
0.5	1000.00	1000.00	1000.00	1000.00	1000.00	938.90	733.86	323.93	106.59	**61.90**
0.6	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
0.7	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00

Table A.7: Wilcoxon rank-sum test: p-values for different values of χ and λ_c on Diagonal with problem size $n = 1000$. Each p-value compares runtimes against the combination of $\chi = 0.3$ and $\lambda_c = 0.2$. Bold texts denote the combinations that have statistically significant differences against the runtime under the configuration of $\chi = 0.3$ and $\lambda_c = 0.2$. The p-value of less than 0.05 indicates that this test rejects the hypothesis at the 5% significance level.

$\chi\backslash\lambda_c$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{1.14 \times 10^{-26}}$	$\mathbf{5.71 \times 10^{-27}}$	$\mathbf{2.10 \times 10^{-28}}$	$\mathbf{2.50 \times 10^{-30}}$	$\mathbf{4.95 \times 10^{-33}}$	$\mathbf{3.08 \times 10^{-33}}$	$\mathbf{1.81 \times 10^{-33}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.2	$\mathbf{9.96 \times 10^{-32}}$	$\mathbf{5.03 \times 10^{-4}}$	$\mathbf{2.20 \times 10^{-21}}$	$\mathbf{5.05 \times 10^{-26}}$	$\mathbf{1.12 \times 10^{-27}}$	$\mathbf{6.47 \times 10^{-29}}$	$\mathbf{1.85 \times 10^{-32}}$	$\mathbf{4.66 \times 10^{-33}}$	$\mathbf{4.53 \times 10^{-33}}$	$\mathbf{1.35 \times 10^{-33}}$
0.3	$\mathbf{1.68 \times 10^{-15}}$	-	$\mathbf{2.18 \times 10^{-19}}$	$\mathbf{3.87 \times 10^{-25}}$	$\mathbf{5.42 \times 10^{-27}}$	$\mathbf{2.34 \times 10^{-28}}$	$\mathbf{6.46 \times 10^{-32}}$	$\mathbf{4.80 \times 10^{-33}}$	$\mathbf{4.66 \times 10^{-33}}$	$\mathbf{2.51 \times 10^{-33}}$
0.4	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.36 \times 10^{-21}}$	$\mathbf{1.18 \times 10^{-24}}$	$\mathbf{5.42 \times 10^{-27}}$	$\mathbf{2.61 \times 10^{-28}}$	$\mathbf{5.12 \times 10^{-32}}$	$\mathbf{4.95 \times 10^{-33}}$	$\mathbf{4.66 \times 10^{-33}}$	$\mathbf{9.70 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.5	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.6	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$
0.7	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$	$\mathbf{2.52 \times 10^{-34}}$

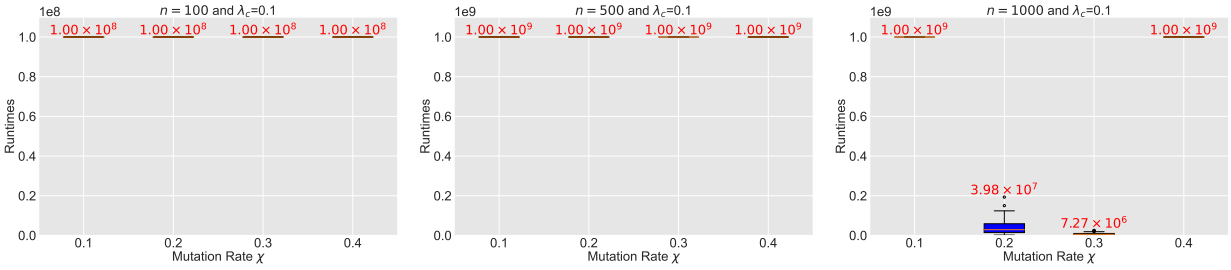
null hypothesis at the 5% level. These might suggest the best possible mutation rates $\chi = 0.3$ and the best population size $\lambda = 0.2n$ under problem size $n = 1000$.

Figure A.6 and Figure A.7 present the boxplots for runtime distribution of PDCoEA with different mutation rates and population size under problem size $n = 100, 500, 1000$. These boxplots suggest that low mutation rates and large population sizes can maintain the stable performance of PDCoEA. However, there is a trade-off between stable performance and better runtime. Thus, we need to be careful in choosing the proper population size for PDCoEA on DIAGONAL.

Finally, we conduct the experiments of PDCoEA on DIAGONAL by choosing $\chi = 0.3$ and $\lambda = c \log n$ where $c = 10, 20, 30, 40, 50, 60$. We can see that our current theoretical bound $O(\lambda n(\lambda^2 + n^2))$ (represented by the orange curve in Figure A.8) is validated in these cases. However, it is worth noticing that our bound is still rough compared to the bound $O(\lambda n \log n)$ (represented by the red curve in Figure A.8). More precise runtime bound can be derived for PDCoEA on DIAGONAL, which we conjecture might be $O(\lambda n \log n)$.

Table A.8: Empirical mean runtimes of PDCoEA for different values of n and c where $\lambda = c \cdot \log(n)$ on Diagonal with problem size $n = 100$ to $n = 1000$, expressed as multiples of 10^6. For each configuration, we conduct 100 independent runs. The function evaluation for each run is 10^8.

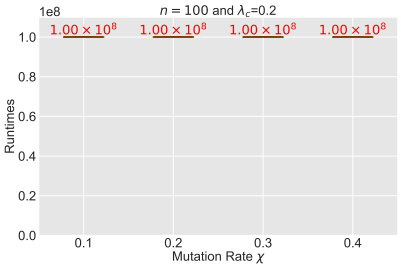
$c \backslash n$	100.0	200.0	300.0	400.0	500.0	600.0	700.0	800.0	900.0	1000.0
10.0	0.16	0.55	1.01	1.87	2.38	2.81	4.22	5.42	6.14	10.36
20.0	0.14	0.34	0.59	0.87	1.15	1.37	1.75	2.00	2.40	2.77
30.0	0.17	0.43	0.71	1.02	1.35	1.74	2.05	2.48	2.83	3.28
40.0	0.21	0.52	0.88	1.26	1.68	2.11	2.45	2.88	3.38	4.00
50.0	0.25	0.62	1.04	1.49	2.01	2.42	3.03	3.40	3.99	4.56
60.0	0.29	0.72	1.19	1.71	2.25	2.85	3.37	3.95	4.51	5.12



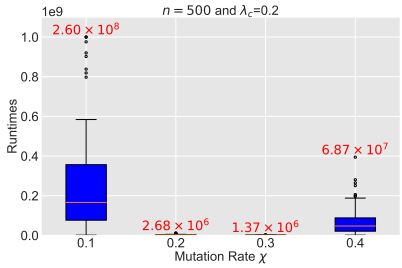
(a) $n = 100, \lambda = 0.1n$

(b) $n = 500, \lambda = 0.1n$

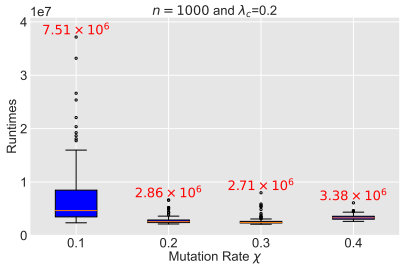
(c) $n = 1000, \lambda = 0.1n$



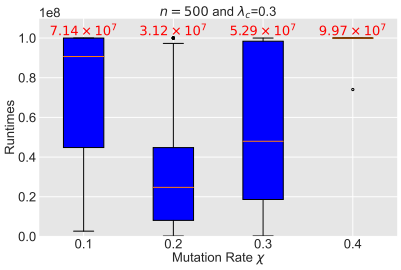
(d) $n = 100, \lambda = 0.2n$



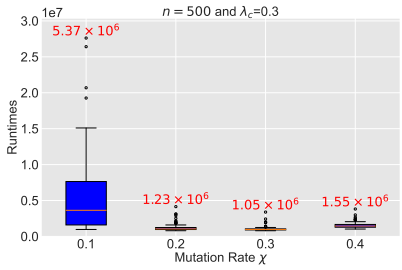
(e) $n = 500, \lambda = 0.2n$



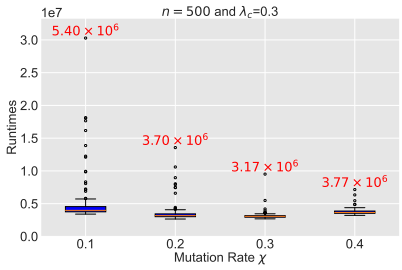
(f) $n = 1000, \lambda = 0.2n$



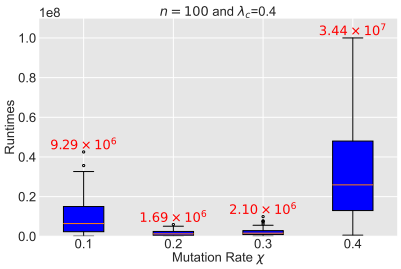
(g) $n = 100, \lambda = 0.3n$



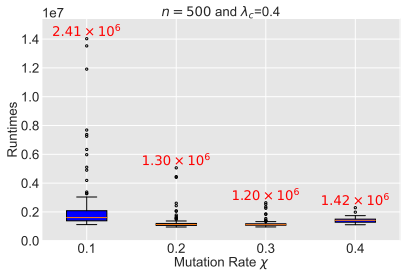
(h) $n = 500, \lambda = 0.3n$



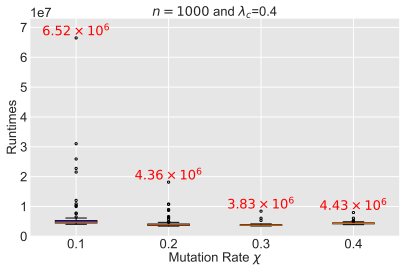
(i) $n = 1000, \lambda = 0.3n$



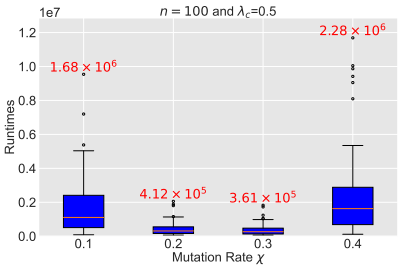
(j) $n = 100, \lambda = 0.4n$



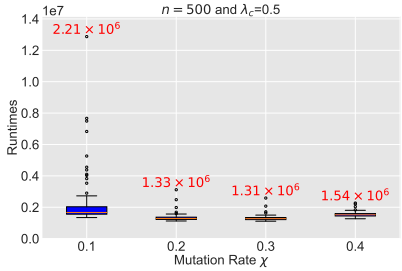
(k) $n = 500, \lambda = 0.4n$



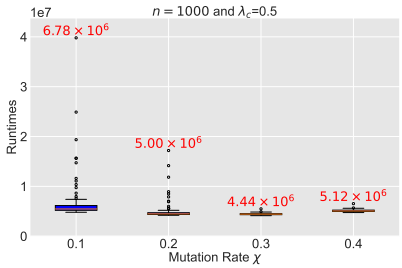
(l) $n = 1000, \lambda = 0.4n$



(m) $n = 100, \lambda = 0.5n$

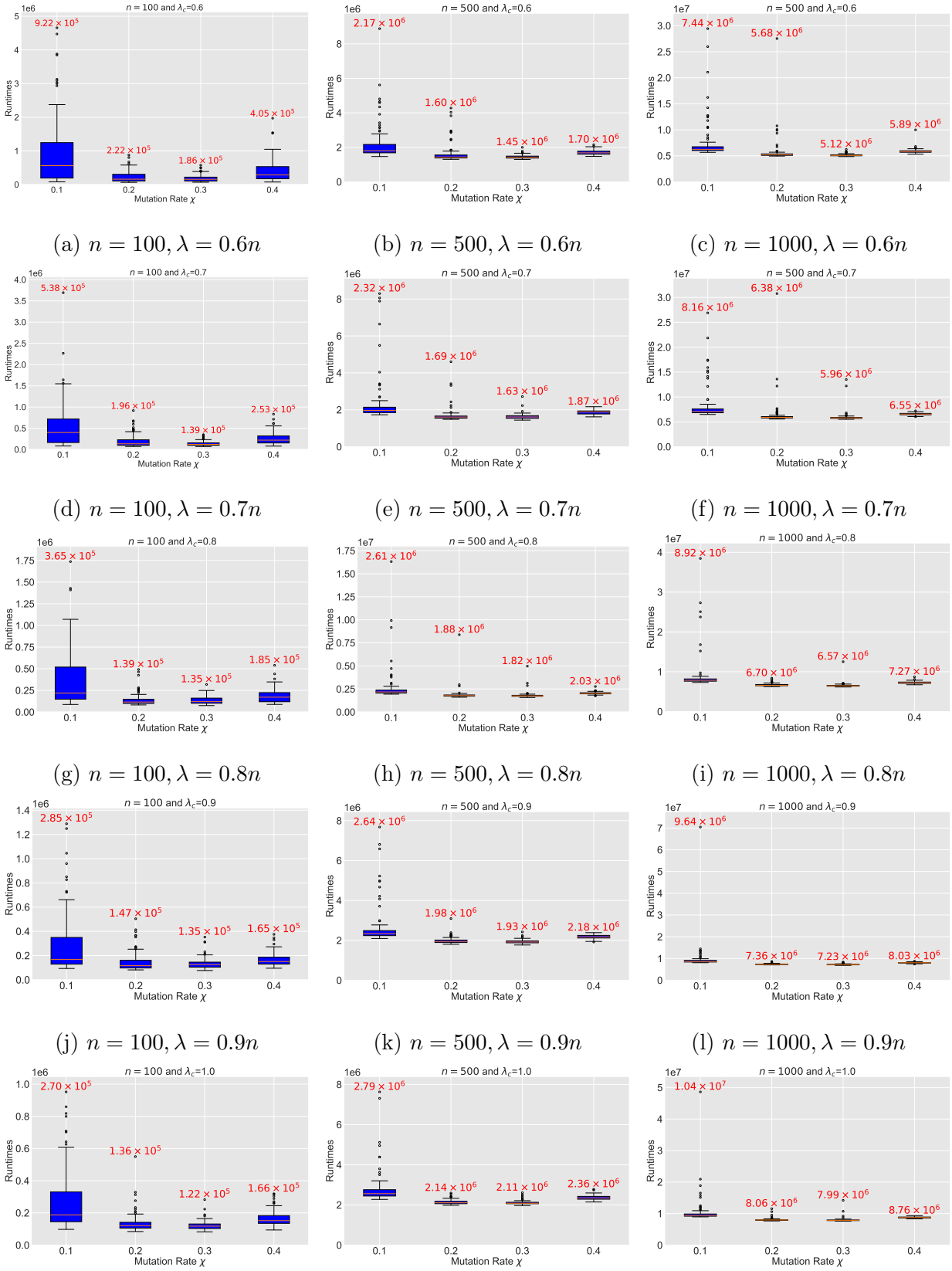


(n) $n = 500, \lambda = 0.5n$



(o) $n = 1000, \lambda = 0.5n$

Figure A.6: Comparison of boxplots for $\lambda = 0.1n$ to $\lambda = 0.5n$ across different configurations of n.



(a) $n = 100, \lambda = 0.6n$ (b) $n = 500, \lambda = 0.6n$ (c) $n = 1000, \lambda = 0.6n$

(d) $n = 100, \lambda = 0.7n$ (e) $n = 500, \lambda = 0.7n$ (f) $n = 1000, \lambda = 0.7n$

(g) $n = 100, \lambda = 0.8n$ (h) $n = 500, \lambda = 0.8n$ (i) $n = 1000, \lambda = 0.8n$

(j) $n = 100, \lambda = 0.9n$ (k) $n = 500, \lambda = 0.9n$ (l) $n = 1000, \lambda = 0.9n$

(m) $n = 100, \lambda = 1.0n$ (n) $n = 500, \lambda = 1.0n$ (o) $n = 1000, \lambda = 1.0n$

Figure A.7: Comparison of boxplots for $\lambda = 0.6n$ to $\lambda = n$ across different configurations

of n.

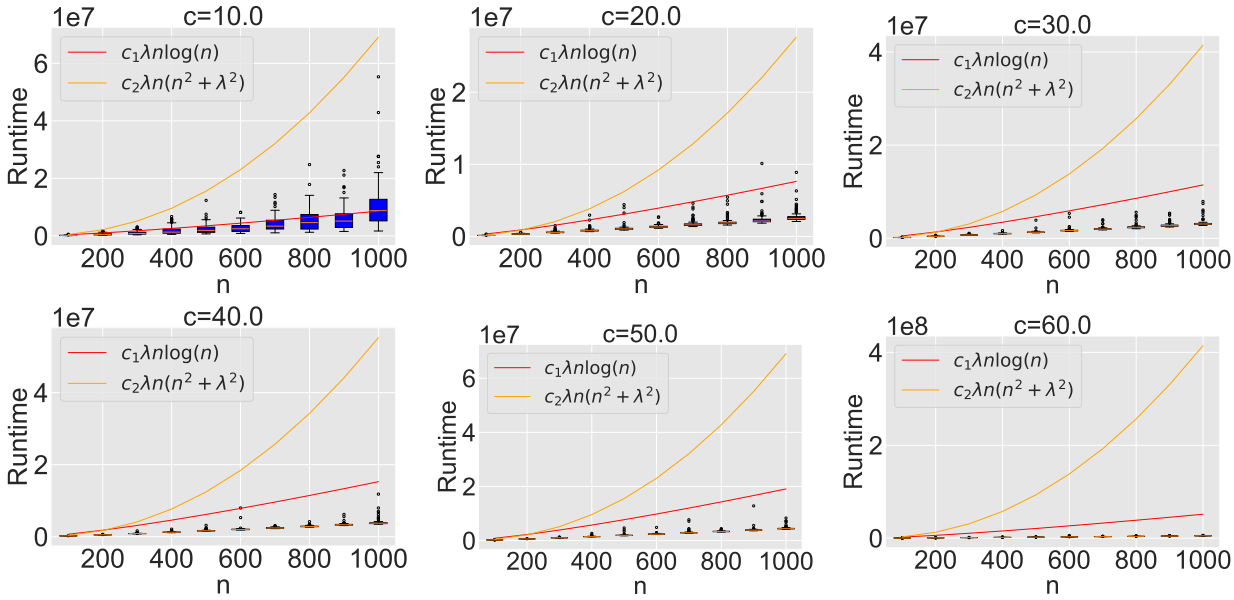


Figure A.8: Boxplots for runtime under $\chi = 0.3$ with respect to different c and problem size n.

References

Adams, Stefan (2021). "MA3K0 - High-Dimensional Probability Lecture Notes". en. In: p. 77.

Anderson, Roy M and Robert M May (1982). "Coevolution of hosts and parasites". In: *Parasitology* 85.2, pp. 411–426.

Auer, Peter, Nicolo Cesa-Bianchi and Paul Fischer (2002). "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47, pp. 235–256.

Auger, David, Jianlin Liu, Sylkvie Ruette, David Saint-Pierre and Oliver Teytaud (2015). "Sparse Binary Zero-Sum Games". In: *Asian Conference on Machine Learning*. PMLR, pp. 173–188.

Axelrod, Robert et al. (1987). "The evolution of strategies in the iterated prisoner's dilemma". In: *The dynamics of norms* 1, pp. 1–16.

Aziz, Haris, Felix Brandt, Edith Elkind and Piotr Skowron (2019). "Computational social choice: The first ten years and beyond". In: *Computing and Software Science: State of the Art and Perspectives*, pp. 48–65.

Babichenko, Yakov (2016). "Query complexity of approximate Nash equilibria". In: *Journal of the ACM (JACM)* 63.4, pp. 1–24.

– (2020). "Informational bounds on equilibria (a survey)". In: *ACM SIGecom Exchanges* 17.2, pp. 25–45.

Back, Thomas (1994). "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms". In: *Proceedings of the first IEEE conference on evolutionary computation. IEEE World Congress on Computational Intelligence*. IEEE, pp. 57–62.

Bäck, Thomas, David B Fogel and Zbigniew Michalewicz (1997). "Handbook of evolutionary computation". In: *Release* 97.1, B1.

Banzhaf, Wolfgang, Peter Nordin, Robert E Keller and Frank D Francone (1998). *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc.

Batcher, Kenneth E (1968). "Sorting networks and their applications". In: *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pp. 307–314.

Benford, Alistair and Per Kristian Lehre (2024a). "A General Upper Bound for the Runtime of a Coevolutionary Algorithm on Impartial Combinatorial Games ". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '25. Málaga, Spain: Association for Computing Machinery.

– (2024b). "Runtime Analysis of Coevolutionary Algorithms on a Class of Symmetric Zero-Sum Games". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. Melbourne, VIC, Australia: Association for Computing Machinery.

Benford, Alistair, Markus Olhofer, Tobias Rodemann and Per Kristian Lehre (2024). "Bi-criteria optimisation of average and worst-case performance using coevolutionary algorithms". In: *IEEE Congress on Evolutionary Computation (IEEE CEC) 2024*. IEEE.

Beyer, Hans-Georg and Hans-Paul Schwefel (2002). "Evolution strategies–a comprehensive introduction". In: *Natural computing* 1, pp. 3–52.

Beznosikov, Aleksandr, Abdurakhmon Sadiev and Alexander Gasnikov (2020). "Gradient-free methods with inexact oracle for convex-concave stochastic saddle-point problem". In: *International Conference on Mathematical Optimization Theory and Operations Research*. Springer, pp. 105–119.

Bhattacharya, Rabindra Nath and Edward C Waymire (2007). *A Basic Course in Probability Theory*. Vol. 69. Springer.

Bian, Chao, Chao Feng, Chao Qian and Yang Yu (2020). "An efficient evolutionary algorithm for subset selection with general cost constraints". In: *AAAI*.

Boros, Endre and Peter L Hammer (2002). "Pseudo-boolean optimization". In: *Discrete applied mathematics* 123.1-3, pp. 155–225.

Brandt, Felix, Vincent Conitzer and Ulle Endriss (2012). "Computational social choice". In: *Multiagent systems* 2, pp. 213–284.

Brandt, Felix, Vincent Conitzer, Ulle Endriss, Jérôme Lang and Ariel D Procaccia (2016). *Handbook of computational social choice*. Cambridge University Press.

Bubeck, Sébastien, Nicolo Cesa-Bianchi et al. (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *Foundations and Trends® in Machine Learning* 5.1, pp. 1–122.

Bucci, Anthony (2007). *Emergent geometric organization and informative dimensions in coevolutionary algorithms*. Brandeis University.

Bucci, Anthony, Jordan B Pollack and Edwin De Jong (2004). "Automated extraction of problem structure". In: *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*. Springer, pp. 501–512.

Cameron, Peter J. (1994). *Combinatorics: Topics, Techniques, Algorithms*. Cambridge, UK: Cambridge University Press.

Case, Brendan and Per Kristian Lehre (2020a). "Self-adaptation in non-Elitist Evolutionary Algorithms on Discrete Problems with Unknown Structure". In: *IEEE Transactions on Evolutionary Computation* 24.4, pp. 650–663.

– (2020b). "Self-Adaptation in Nonelitist Evolutionary Algorithms on Discrete Problems With Unknown Structure". In: *IEEE Transactions on Evolutionary Computation* 24.4, pp. 650–663.

– (2020c). "Self-adaptation in nonelitist evolutionary algorithms on discrete problems with unknown structure". In: *IEEE Transactions on Evolutionary Computation* 24.4, pp. 650–663.

Chleboun, Paul (2021). "ST318 - Probability Theory Lecturer Notes". In.

Cliff, Dave and Geoffrey F. Miller (1995). "Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations". In: *Advances in Artificial Life*. Ed. by G. Goos, J. Hartmanis, J. Leeuwen, Jaime G. Carbonell, Jörg Siekmann, Federico Morán, Alvaro Moreno, Juan Julián Merelo and Pablo Chacón. Vol. 929. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 200–218.

Cohn, Donald L. (2013). *Measure Theory*. Birkhäuser Advanced Texts Basler Lehrbücher. New York, NY: Springer New York.

Colson, Benoit, Patrice Marcotte and Gilles Savard (2007). "An overview of bilevel optimization". In: *Annals of Operations Research* 153, pp. 235–256.

Corless, Robert M, Gaston H Gonnet, David EG Hare, David J Jeffrey and Donald E Knuth (1996). "On the Lambert W function". In: *Advances in Computational mathematics* 5, pp. 329–359.

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.

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. 14, 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, Anton Eremeev and Per Kristian Lehre (2021c). "Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions and dense valleys". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1133–1141.

Dang, Duc-Cuong and Per Kristian Lehre (2024). "The SLO Hierarchy of pseudo-Boolean Functions and Runtime of Evolutionary Algorithms". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1551–1559.

Dang, Duc-Cuong, Per Kristian Lehre and Phan Trung Hai Nguyen (2019a). "Level-Based Analysis of the Univariate Marginal Distribution Algorithm". In: *Algorithmica* 81.2, pp. 668–702.

– (2019b). "Level-based analysis of the univariate marginal distribution algorithm". In: *Algorithmica* 81.2, pp. 668–702.

Dantzig, George B (1951). "A Proof of the Equivalence of the Programming Problem and the Game Problem". In: *Activity analysis of production and allocation* 13.

Darwin, Charles (1859). *On the Origin of Species*.

Daskalakis, Constantinos, Paul W Goldberg and Christos H Papadimitriou (2009). "The Complexity of Computing a Nash Equilibrium". In: *Communications of the ACM* 52.2, pp. 89–97.

Daskalakis, Constantinos and Ioannis Panageas (2018). "The limit points of (optimistic) gradient descent in min-max optimization". In: *Advances in Neural Information Processing Systems* 31.

Dawkins, Richard and John Richard Krebs (1979). "Arms Races between and within Species". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 205.1161, pp. 489–511.

De Jong, Edwin (2005). "The maxsolve algorithm for coevolution". In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 483–489.

De Jong, Edwin D (2004a). "The incremental pareto-coevolution archive". In: *Genetic and Evolutionary Computation Conference*. Springer, pp. 525–536.

– (2004b). "The Incremental Pareto-Coevolution Archive". en. In: *Genetic and Evolutionary Computation – GECCO 2004*. Ed. by Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum and Kalyanmoy Deb. Vol. 3102. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 525–536.

De Jong, Edwin D. and Jordan B. Pollack (June 2004). "Ideal Evaluation from Coevolution". en. In: *Evolutionary Computation* 12.2, pp. 159–192.

De Jong, Edwin D., Kenneth O. Stanley and R. Paul Wiegand (2007). "Introductory tutorial on coevolution". en. In: *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation - GECCO '07*. London, United Kingdom: ACM Press, p. 3133. ISBN: 978-1-59593-698-1. DOI: 10.1145/1274000.1274108.

De Jong, Kenneth (2017). "Evolutionary computation: a unified approach". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 373–388.

De Jong, Kenneth Alan (1975). *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan.

Dempe, Stephan (2002). *Foundations of bilevel programming*. Springer Science & Business Media.

Do, Anh Viet, Aneta Neumann, Frank Neumann and Andrew M. Sutton (2023). "Rigorous Runtime Analysis of MOEA/D for Solving Multi-Objective Minimum Weight Base Problems". In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Doerr, Benjamin (2020). "Does comma selection help to cope with local optima?" In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 1304–1313.

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 and Leslie Ann Goldberg (2013). "Adaptive Drift Analysis". In: *Algorithmica* 65.1, pp. 224–250. ISSN: 1432-0541.

Doerr, Benjamin, Daniel Johannsen and Carola Winzen (2010). "Multiplicative Drift Analysis". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. GECCO '10. Portland, Oregon, USA: Association for Computing Machinery, pp. 1449–1456.

Doerr, Benjamin and Timo Kötzing (2021). "Multiplicative Up-Drift". In: *Algorithmica* 83.10, pp. 3017–3058.

Doerr, Benjamin, Timo Kötzing, Johannes Lengler and Carola Winzen (2013). "Black-box complexities of combinatorial problems". In: *Theoretical Computer Science* 471, pp. 84–106.

Doerr, Benjamin and Frank Neumann (2019). *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. Natural Computing Series. Springer Nature.

– eds. (2020). *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*. en. Natural Computing Series. Cham: Springer International Publishing.

Doerr, Benjamin and Zhongdi Qu (2023). "Runtime analysis for the NSGA-II: Provable speed-ups from crossover". In: *AAAI*.

Doerr, Benjamin, Dirk Sudholt and Carsten Witt (2013). "When Do Evolutionary Algorithms Optimize Separable Functions in Parallel?" In: *Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII*. FOGA XII '13. Adelaide, Australia: Association for Computing Machinery, pp. 51–64.

Doerr, Benjamin and Weijie Zheng (2020). "Sharp Bounds for Genetic Drift in Estimation of Distribution Algorithms". In: *IEEE Transactions on Evolutionary Computation* 24.6, pp. 1140–1149.

Doob, J. L. (1971). "What is a Martingale?" In: *The American Mathematical Monthly* 78.5. Publisher: Mathematical Association of America, pp. 451–463.

Droste, Stefan, Thomas Jansen and Ingo Wegener (2002a). "On the analysis of the (1+1) evolutionary algorithm". In: *Theoretical Computer Science* 276.1-2, pp. 51–81.

– (2002b). "Optimization with randomized search heuristics—the (A) NFL theorem, realistic scenarios, and difficult functions". In: *Theoretical Computer Science* 287.1, pp. 131–144.

– (2006). "Upper and lower bounds for randomized search heuristics in black-box optimization". In: *Theory of computing systems* 39.4, pp. 525–544.

Al-Dujaili, Abdullah, Tom Schmiedlechner, and Erik Hemberg and Una-May O'Reilly (Aug. 2018). *Towards Distributed Coevolutionary GANs*. en. arXiv:1807.08194 [cs].

Dvinskikh, Darina, Vladislav Tominin, Yaroslav Tominin and Alexander Gasnikov (2022). "Gradient-Free Optimization for Non-Smooth Saddle Point Problems under Adversarial Noise". In: *arXiv preprint arXiv:2202.06114*.

Eiben, A.E. and J.E. Smith (2015). *Introduction to Evolutionary Computing*. en. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg.

Fajardo, Mario Alejandro Hevia and Per Kristian Lehre (2024). "Ranking Diversity Benefits Coevolutionary Algorithms on an Intransitive Game". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 213–229.

Fajardo, Mario Alejandro Hevia, Per Kristian Lehre and Shishen Lin (2023). "Runtime Analysis of a Co-Evolutionary Algorithm: Overcoming Negative Drift in Maximin-Optimisation". In: *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. FOGA '23. Potsdam, Germany: Association for Computing Machinery, pp. 73–83.

Ficici, Sevan G (2004). "Solution Concepts in Coevolutionary Algorithms". en. In: p. 299.

Ficici, Sevan G and Jordan B Pollack (1998a). "Challenges in Coevolutionary Learning, Arms-Race Dynamics: Open-Endedness, and Mediocre Stable States". en. In: p. 10.

– (1998b). "Challenges in Coevolutionary Learning, Arms-Race Dynamics: Open-Endedness, and Mediocre Stable States". In: *Proceedings of the sixth international conference on Artificial life*. MIT Press Cambridge, MA, pp. 238–247.

Ficici, Sevan G, Jordan B Pollack et al. (1998). "Coevolving communicative behavior in a linear pursuer-evader game". In: *From Animals to Animats. Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior, SAB-98*, pp. 263–269.

Fogel, David B (1998). *Evolutionary computation: the fossil record*. John Wiley & Sons.

Forsyth, Richard (1981). "BEAGLE—A Darwinian approach to pattern recognition". In: *Kybernetes* 10.3, pp. 159–166.

Friedrich, Tobias, Timo Kötzing and Martin S. Krejca (2016). "EDAs Cannot Be Balanced and Stable". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: Association for Computing Machinery, pp. 1139–1146.

Friedrich, Tobias, Timo Kötzing, Francesco Quinzan and Andrew M Sutton (2018). "Improving the run time of the (1+ 1) evolutionary algorithm with Luby sequences". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, NY, USA: Association for Computing Machinery, pp. 301–308.

Fudenberg, Drew and David K Levine (1998). *The Theory of Learning in Games*. MIT press.

Gasnikov, Alexander, Anton Novitskii, Vasilii Novitskii, Farshed Abdukhakimov, Dmitry Kamzolov, Aleksandr Beznosikov, Martin Takáč, Pavel Dvurechensky and Bin Gu (2022). "The power of first-order smooth optimization for black-box non-smooth problems". In: *arXiv preprint arXiv:2201.12289*.

Gidel, Gauthier, Tony Jebara and Simon Lacoste-Julien (2017). "Frank-wolfe algorithms for saddle point problems". In: *Artificial Intelligence and Statistics*. PMLR, pp. 362–371.

Göbel, Andreas, Timo Kötzing and Martin S Krejca (2022). *Intuitive Analyses via Drift Theory*. arXiv: 1806.01919 [math.PR].

Golovin, Daniel, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro and David Sculley (2017). "Google vizier: A service for black-box optimization". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495.

Gomes, Jorge, Pedro Mariano and Anders Lyhne Christensen (2014a). "Novelty Search in Competitive Coevolution". In: *Parallel Problem Solving from Nature–PPSN XIII: 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings 13*. Springer, pp. 233–242.

– (2014b). "Novelty Search in Competitive Coevolution". en. In: vol. 8672. arXiv:1407.0576 [cs], pp. 233–242. DOI: 10.1007/978-3-319-10762-2_23.

Goodfellow, Ian (Apr. 2017). *NIPS 2016 Tutorial: Generative Adversarial Networks*. en. Number: arXiv:1701.00160 arXiv:1701.00160 [cs].

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing systems* 27.

– (2020). "Generative Adversarial Nets". In: *Commun. ACM* 63.11, pp. 139–144.

Grand-Clément, Julien and Christian Kroer (2021). "Conic blackwell algorithm: Parameter-free convex-concave saddle-point solving". In: *Advances in Neural Information Processing Systems* 34, pp. 9587–9599.

Grandi, Umberto, Davide Grossi and Paolo Turrini (2015). "Equilibrium Refinement through Negotiation in Binary Voting". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, pp. 540–546.

Grimmett, Geoffrey and David Stirzaker (2001). *Probability and Random Processes*. en. 3rd ed. Oxford ; New York: Oxford University Press.

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.

Hamming, Richard W (1986). *Coding and information theory*. Prentice-Hall, Inc.

Hansen, Nikolaus and Andreas Ostermeier (2001). "Completely derandomized self-adaptation in evolution strategies". In: *Evolutionary Computation* 9.2, pp. 159–195.

Harrenstein, Paul, Wiebe van der Hoek, John-Jules Meyer and Cees Witteveen (2001). "Boolean Games". In: *Proceedings of the 8th Conference on Theoretical Aspects of Ra-*

tionality and Knowledge. TARK '01. Siena, Italy: Morgan Kaufmann Publishers Inc., pp. 287–298.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman and Jerome H Friedman (2009). *The elements of statistical learning: Data mining, inference, and prediction.* Vol. 2. Springer.

He, Jun and Xin Yao (Mar. 2001). "Drift analysis and average time complexity of evolutionary algorithms". In: *Artificial Intelligence* 127.1, pp. 57–85.

Hemberg, Erik, Jamal Toutouh, Abdullah Al-Dujaili, Tom Schmiedlechner and Una-May O'Reilly (2021). "Spatial coevolution for generative adversarial network training". In: *ACM Transactions on Evolutionary Learning and Optimization* 1.2, pp. 1–28.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler and Sepp Hochreiter (2017). "GANs Trained by a Two Time-scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems* 30.

Hevia Fajardo, Mario Alejandro, Erik Hemberg, Jamal Toutouh, Una-May O'Reilly and Per Kristian Lehre (July 2024). "A Self-adaptive Coevolutionary Algorithm". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '24. New York, NY, USA: Association for Computing Machinery, pp. 841–849.

Hevia Fajardo, Mario Alejandro and Per Kristian Lehre (2023). "How Fitness Aggregation Methods Affect the Performance of Competitive CoEAs on Bilinear Problems". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. Lisbon, Portugal: Association for Computing Machinery, pp. 1593–1601. ISBN: 9798400701191.

Hevia Fajardo, Mario Alejandro, Per Kristian Lehre and Shishen Lin (2023). "Runtime Analysis of a Co-Evolutionary Algorithm: Overcoming Negative Drift in Maximin-Optimisation". In: *Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA)*. Potsdam, Germany.

Hillis, W.Daniel (June 1990). "Co-evolving parasites improve simulated evolution as an optimization procedure". en. In: *Physica D: Nonlinear Phenomena* 42.1-3, pp. 228–234. ISSN: 01672789.

Ho, Y.C. and D.L. Pepyne (Dec. 2002). "Simple Explanation of the No-Free-Lunch Theorem and Its Implications". In: *Journal of Optimization Theory and Applications* 115.3, pp. 549–570.

Holland, John H (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hu, Junling, Michael P Wellman et al. (1998). "Multiagent reinforcement learning: theoretical framework and an algorithm." In: *ICML*. Vol. 98, pp. 242–250.

Hu, Junling and Michael P Wellman (2003). "Nash Q-Learning for General-Sum Stochastic Games". In: *Journal of machine learning research* 4.Nov, pp. 1039–1069.

Igel, Christian and Marc Toussaint (2005). "A no-free-lunch theorem for non-uniform distributions of target functions". In: *Journal of Mathematical Modelling and Algorithms* 3.4, pp. 313–322.

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 (2013a). *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Springer Publishing Company, Incorporated.

– (2013b). *Analyzing evolutionary algorithms: The computer science perspective*. Springer.

Jansen, Thomas and R. Paul Wiegand (2004). "The Cooperative Coevolutionary (1+1) EA". In: *Evol. Comput.* 12.4, pp. 405–434.

Jaśkowski, Wojciech (2011). "Algorithms for test-based problems". In: *Adviser: Krzysztof Krawiec. PhD thesis. Poznan, Poland: Institute of Computing Science, Poznan University of Technology*.

Jaśkowski, Wojciech and Krzysztof Krawiec (2010). "Coordinate system archive for coevolution". In: *IEEE Congress on Evolutionary Computation*. IEEE, pp. 1–10.

Jensen, Mikkel T (2001). "A New Look at Solving Minimax Problems with Coevolution". en. In: p. 6.

Johannsen, Daniel (2010). "Random combinatorial structures and randomized search heuristics". In.

Juillé, Hugues and Jordan B Pollack (1996). "Co-evolving Intertwined Spirals". In: *in Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Citeseer.

Kötzing, Timo (2016). "Concentration of First Hitting Times Under Additive Drift". In: *Algorithmica* 75.3, pp. 490–506.

Kötzing, Timo and Martin S. Krejca (2019). "First-hitting times under drift". In: *Theoretical Computer Science* 796, pp. 51–69.

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. Aberystwyth, United Kingdom: Association for Computing Machinery, pp. 40–51.

Koza, John R et al. (1989). "Hierarchical genetic algorithms operating on populations of computer programs." In: *IJCAI*. Vol. 89, pp. 768–774.

Krawiec, Krzysztof and Malcolm Heywood (2020). "Solving complex problems with coevolutionary algorithms". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. GECCO '20. Cancún, Mexico: Association for Computing Machinery, pp. 832–858. ISBN: 9781450371278. DOI: 10.1145/3377929.3389874.

Krejca, Martin S. (2019). "Theoretical Analyses of Univariate Estimation-of-Distribution Algorithms". PhD thesis. Hasso Plattner Institute, University of Potsdam.

Larcher, Maxime, Robert Meier and Angelika Steger (2023). "A Simple Optimal Algorithm for the 2-Arm Bandit Problem". In: *Symposium on Simplicity in Algorithms (SOSA)*. SIAM, pp. 365–372.

Lattimore, Tor and Csaba Szepesvári (2020). *Bandit Algorithms*. Cambridge University Press.

Lehre, P. K. and C. Witt (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 (2010). "Negative Drift in Populations". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 244–253.

– (2012). *Lecture Notes of Drift Analysis*. https://www.cs.bham.ac.uk/~lehrepk/drift/. Accessed: 2021-11-15.

– (2022). "Runtime Analysis of Competitive co-Evolutionary Algorithms for Maximin Optimisation of a Bilinear Function". en. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '22. Boston, Massachusetts, pp. 1408–1416.

– (Apr. 2024). "Runtime Analysis of Competitive Co-evolutionary Algorithms for Maximin Optimisation of a Bilinear Function". en. In: *Algorithmica*.

Lehre, Per Kristian, Mario Hevia Fajardo, Jamal Toutouh, Erik Hemberg and Una-May O'Reilly (2023). "Analysis of a Pairwise Dominance Coevolutionary Algorithm And DefendIt". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '23. Lisbon, Portugal: Association for Computing Machinery, pp. 1027–1035.

Lehre, Per Kristian and Shishen Lin (2023). "Is CC-(1+1) EA more efficient than (1+1) EA on either separable or inseparable problems?" In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. Chicago, USA.

– (2024a). "Concentration Tail-Bound Analysis of Coevolutionary and Bandit Learning Algorithms". In: *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*. Jeju, South Korea.

– (2024b). "No Free Lunch Theorem and Black-Box Complexity Analysis for Adversarial Optimisation". In: *Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS)*. Vancouver, Canada.

Lehre, Per Kristian and Shishen Lin (2024c). "Overcoming Binary Adversarial Optimisation with Competitive Coevolution". In: *Proceedings of the 18th International Conference on Parallel Problem Solving From Nature (PPSN)*. Hagenberg, Austria.

– (2025). "Towards Runtime Analysis of Population-Based Co-evolutionary Algorithms on Sparse Binary Zero-Sum Game". In: *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI)*. Philadelphia, Pennsylvania, USA.

Lehre, Per Kristian and Phan Trung Hai Nguyen (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 Pietro Simone Oliveto (2023). "Runtime Analysis of Population-based Evolutionary Algorithms - Part I: Steady State EAs". In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. GECCO '23 Companion. Lisbon, Portugal: Association for Computing Machinery, pp. 1271–1300.

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.

Lehtonen, Jussi (2016). "The Lambert W function in ecological and evolutionary models". In: *Methods in Ecology and Evolution* 7.9, pp. 1110–1118.

Lengler, Johannes (June 2018). *Drift Analysis*. en. arXiv:1712.00964 [cs, math].

Liang, Tengyuan and James Stokes (Feb. 2019). *Interaction Matters: A Note on Non-asymptotic Local Convergence of Generative Adversarial Networks*. en. arXiv:1802.06132 [cs, stat].

Lin, Tianyi, Chi Jin and Michael Jordan (2020). "On gradient descent ascent for nonconvex-concave minimax problems". In: *International Conference on Machine Learning*. PMLR, pp. 6083–6093.

Lindgren, Kristian (1992). "Evolutionary phenomena in simple dynamics". In: *Artificial life II*, pp. 295–312.

Maiti, Arnab, Ross Boczar, Kevin Jamieson and Lillian J Ratliff (2023). "Query-Efficient Algorithm to Find all Nash Equilibria in a Two-Player Zero-Sum Matrix Game". In: *arXiv preprint arXiv:2310.16236*.

McDiarmid, Colin (1989). "On the method of bounded differences". In: *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*. London Mathematical Society Lecture Note Series. Cambridge University Press, pp. 148–188.

– (1993). "A Random Recolouring Method for Graphs and Hypergraphs". In: *Combinatorics, Probability and Computing* 2.3, pp. 363–365.

Meir, Reshef, Maria Polukarov, Jeffrey S. Rosenschein and Nicholas R. Jennings (2010). "Convergence to equilibria in plurality voting". In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, Georgia: AAAI Press, pp. 823–828.

Menshikov, Mikhail, Serguei Popov and Andrew Wade (2016). *Non-homogeneous Random Walks: Lyapunov Function Methods for Near-Critical Stochastic Systems*. Cambridge Tracts in Mathematics. Cambridge University Press.

Mertikopoulos, Panayotis, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar and Georgios Piliouras (2018). "Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile". In: *arXiv preprint arXiv:1807.02629*.

Miconi, Thomas (2009). "Why Coevolution Doesn't "Work": Superiority and Progress in Coevolution". In: *Genetic Programming*. Ed. by Leonardo Vanneschi, Steven Gustafson, Alberto Moraglio, Ivanoe De Falco and Marc Ebner. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 49–60. ISBN: 978-3-642-01181-8.

Mitavskiy, Boris, Jonathan Rowe and Chris Cannings (2009). "Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links". In: *International Journal of Intelligent Computing and Cybernetics* 2.2, pp. 243–284.

Mitchell, Melanie (2006). "Coevolutionary learning with spatially distributed populations". In: *Computational intelligence: principles and practice* 400.

Mitzenmacher, Michael and Eli Upfal (2005). *Probability and computing: an introduction to randomized algorithms and probabilistic analysis*. en. New York: Cambridge University Press.

Miyagi, Atsuhiro, Kazuto Fukuchi, Jun Sakuma and Youhei Akimoto (2022). "Black-box min-max continuous optimization using cma-es with worst-case ranking approximation". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 823–831.

Mokhtari, Aryan, Asuman Ozdaglar and Sarath Pattathil (Sept. 2019). *A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach*. en. arXiv:1901.08511 [cs, math, stat].

Motwani, Rajeev and Prabhakar Raghavan (1995). *Randomized Algorithms*. Cambridge university press.

Nash, John (1951a). "Non-Cooperative Games". en. In: *Annals of Mathematics*, pp. 286–295.

– (1951b). "Non-Cooperative Games". In: *Annals of Mathematics* 54.2, pp. 286–295.

Neumann, Aneta, Denis Antipov and Frank Neumann (2022). *Coevolutionary Pareto Diversity Optimization*. en. arXiv:2204.05457 [cs].

Neumann, Frank and Ingo Wegener (June 2007). "Randomized local search, evolutionary algorithms, and the minimum spanning tree problem". en. In: *Theoretical Computer Science* 378.1, pp. 32–40. ISSN: 03043975. DOI: 10.1016/j.tcs.2006.11.002.

Neumann, Frank and Carsten Witt (2010a). "Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity". In.

– (2010b). *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg.

Neumann, John von (1928). "Zur Theorie der Gesellschaftsspiele". In: *Mathematische Annalen* 100.1, pp. 295–320.

Nisan, Noam, Tim Roughgarden, Eva Tardos and Vazirani Vijay V. (2007). *Algorithmic Game Theory*. Cambridge University Press.

Nisan, Noam, Tim Roughgarden, Eva Tardos and Vijay V. Vazirani, eds. (2007). *Algorithmic Game Theory*. en. Cambridge: Cambridge University Press. ISBN: 978-0-511-80048-1. DOI: 10.1017/CBO9780511800481.

Nolfi, Stefano and Dario Floreano (Oct. 1998). "Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution?" en. In: *Artificial Life* 4.4, pp. 311–335.

O'Donnell, Ryan (2008). "Some topics in analysis of Boolean functions". In: *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, pp. 569–578.

Oliveto, Pietro S. and Carsten Witt (Nov. 2012). *Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation*. arXiv:1211.7184 [cs].

Osborne, Martin J and Ariel Rubinstein (1994). *A course in game theory*. MIT press.

Pagie, L. and P. Hogeweg (2000). "Information integration and red queen dynamics in coevolutionary optimization". en. In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*. Vol. 2. La Jolla, CA, USA: IEEE, pp. 1260–1267.

Palaniappan, Balamurugan and Francis Bach (2016). "Stochastic variance reduction methods for saddle-point problems". In: *Advances in Neural Information Processing Systems* 29.

Panageas, Ioannis, Nikolas Patris, Stratis Skoulakis and Volkan Cevher (2023). "Exponential Lower Bounds for Fictitious Play in Potential Games". In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Papadimitriou, Christos H (1991). "On selecting a satisfying truth assignment". In: *Proceedings 32nd Annual Symposium of Foundations of Computer Science*. IEEE Computer Society, pp. 163–169.

Perolat, Julien, Bart de Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis and Karl Tuyls (Dec. 2022). "Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning". en. In: *Science* 378.6623, pp. 990–996.

Pollack, Jordan B (1998). "Co-Evolution in the Successful Learning of Backgammon Strategy". en. In: p. 16.

Pollack, Jordan B and Alan D Blair (1998). "Co-evolution in the Successful Learning of Backgammon Strategy". In: *Machine Learning* 32, pp. 225–240.

Popovici, Elena, Anthony Bucci, R. Paul Wiegand and Edwin D. De Jong (2012). "Coevolutionary Principles". en. In: *Handbook of Natural Computing*. Ed. by Grzegorz Rozenberg, Thomas Bäck and Joost N. Kok. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 987–1033.

Potter, Mitchell A (1997). *The design and analysis of a computational model of cooperative coevolution*. George Mason University.

Potter, Mitchell A. and Kenneth A. De Jong (Mar. 2000). "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents". en. In: *Evolutionary Computation* 8.1, pp. 1–29.

Qin, Xiaoyu and Per Kristian Lehre (2022). "Self-adaptation via multi-objectivisation: an empirical study". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 308–323.

Rechenberg, Ingo (1978). "Evolutionsstrategien". In: *Simulationsmethoden in der Medizin und Biologie: Workshop, Hannover, 29. Sept.–1. Okt. 1977*. Springer, pp. 83–114.

Rezek, Iead, David S Leslie, Steven Reece, Stephen J Roberts, Alex Rogers, Rajdeep K Dash and Nicholas R Jennings (2008). "On similarities between inference in game theory and machine learning". In: *Journal of Artificial Intelligence Research* 33, pp. 259–283.

Robey, Alexander, Fabian Latorre, George J. Pappas, Hamed Hassani and Volkan Cevher (2024). "Adversarial Training Should Be Cast as a Non-Zero-Sum Game". In: *The Twelfth International Conference on Learning Representations*.

Rosenbrock, HoHo (1960). "An automatic method for finding the greatest or least value of a function". In: *The computer journal* 3.3, pp. 175–184.

Rosin, Christopher D. and Richard K. Belew (1995a). "Methods for Competitive Co-Evolution: Finding Opponents Worth Beating". In: *Proceedings of the International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 373–381. ISBN: 978-1-55860-370-7.

– (1995b). "Methods for competitive co-evolution: finding opponents worth beating." In: *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA)*. Ed. by Larry J. Eshelman. San Mateo, CA, USA: Morgan Kaufmann, pp. 373–381.

– (1996). "A competitive approach to game learning". en. In: *Proceedings of the ninth annual conference on Computational learning theory - COLT '96*. Desenzano del Garda, Italy: ACM Press, pp. 292–302. ISBN: 978-0-89791-811-4. DOI: 10.1145/238061.238153.

– (Mar. 1997). "New Methods for Competitive Coevolution". en. In: *Evolutionary Computation* 5.1, pp. 1–29.

Rosin, Christopher Darrell (1997). *Coevolutionary search among adversaries*. University of California, San Diego.

Roughgarden, Tim (2010). "Algorithmic game theory". In: *Communications of the ACM* 53.7, pp. 78–86.

Rowe, Jonathan E. and Dirk Sudholt (2012). "The choice of the offspring population size in the (1,λ) EA". In: GECCO '12. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, pp. 1349–1356.

Ruder, Sebastian (June 2017). *An overview of gradient descent optimization algorithms*. arXiv:1609.04747 [cs].

Rudolph, G. (1997). *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač.

Sarafian, Elad, Mor Sinay, Yoram Louzoun, Noa Agmon and Sarit Kraus (2020). "Explicit Gradient Learning for Black-Box Optimization." In: *ICML*, pp. 8480–8490.

Sarker, Ruhul, Masoud Mohammadian and Xin Yao (2002). *Evolutionary optimization*. Vol. 48. Springer Science & Business Media.

Schaffer, Cullen (1994). "A Conservation Law for Generalization Performance". In: *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*. ICML'94. New Brunswick, NJ, USA, pp. 259–265.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel et al. (2020). "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model". In: *Nature* 588.7839, pp. 604–609.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford and Oleg Klimov (2017). "Proximal Policy Optimization Algorithms". In: *arXiv preprint arXiv:1707.06347*.

Schumacher, C., M. D. Vose and L. D. Whitley (2001). "The No Free Lunch and Problem Description Length". In: *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. GECCO'01. San Francisco, California: Morgan Kaufmann Publishers Inc., pp. 565–570.

Service, Travis C. and Daniel R. Tauritz (2008). "A No-Free-Lunch Framework for Coevolution". In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. GECCO '08. Atlanta, GA, USA: Association for Computing Machinery, pp. 371–378.

Shapley, Lloyd S, Samuel Karlin and HF Bohnenblust (1950). "Solutions of discrete, two-person games". In: *Contributions to the Theory of Games*.

Shirali, Satish and Harkrishan Lal Vasudeva (2005). *Metric spaces*. Springer Science & Business Media.

Shu, Han, Yunhe Wang, Xu Jia, Kai Han, Hanting Chen, Chunjing Xu, Qi Tian and Chang Xu (2019). "Co-evolutionary compression for unpaired image translation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3235–3244.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot et al. (2016). "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587, pp. 484–489.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis (Jan. 2016). "Mastering the game of Go with deep neural networks and tree search". en. In: *Nature* 529.7587, pp. 484–489.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel et al. (2018). "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-Play". In: *Science* 362.6419, pp. 1140–1144.

Sims, Karl (1994). "Evolving 3D Morphology and Behavior by Competition". en. In: p. 12.

Smith, John Maynard (1982). "Evolution and the Theory of Games". In: *Did Darwin get it right? Essays on games, sex and evolution*. Springer, pp. 202–215.

Solé, Ricard (2022). "Revisiting Leigh Van Valen's "A new evolutionary law"(1973)". In: *Biological Theory* 17.2, pp. 120–125.

Stanley, Kenneth O and Risto Miikkulainen (2002). "Continual Coevolution through Complexi cation". en. In: p. 8.

Sutton, Richard S, Andrew G Barto et al. (1998). *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Tino, Peter, Siang Yew Chong and Xin Yao (Apr. 2013). "Complex Coevolutionary Dynamics—Structural Stability and Finite Population Effects". en. In: *IEEE Transactions on Evolutionary Computation* 17.2, pp. 155–164. ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2013.2244897.

Toutouh, Jamal, Erik Hemberg and Una-May O'Reilly (2019). "Spatial Evolutionary Generative Adversarial Networks". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '19. Prague, Czech Republic: Association for Computing Machinery, pp. 472–480.

Van Valen, Leigh (1973). "A new evolutionary law". In: *Evolutionary Theory* 1, pp. 1–30.

Von Neumann, John, Oskar Morgenstern et al. (1953). "Theory of Games and Economic Behavior". In: *Princeton*.

Von Stengel, Bernhard (2008). "Game theory basics". In: *Lecture Notes, Department of Mathematics, London School of Economics, Houghton St, London WC2A 2AE, United Kingdom*.

Watson, Richard A and Jordan B Pollack (2001). "Coevolutionary Dynamics in a Minimal Substrate". en. In: p. 8.

Wegener, Ingo (2002). "Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions". In: *Evolutionary optimization*. Springer, pp. 349–369.

Wei, Chen-Yu, Chung-Wei Lee, Mengxiao Zhang and Haipeng Luo (2021). "Linear Last-iterate Convergence in Constrained Saddle-point Optimization". In: *International Conference on Learning Representations*.

Weibull, Jörgen W (1997). *Evolutionary Game Theory*. MIT press.

Wiegand, R. Paul and Mitchell A. Potter (2006). "Robustness in cooperative coevolution". en. In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06*. Seattle, Washington, USA: ACM Press, p. 369. ISBN: 978-1-59593-186-3. DOI: 10.1145/1143997.1144063.

Wiegand, Rudolf Paul (2004). *An analysis of cooperative coevolutionary algorithms*. George Mason University.

Williams, David (1991). *Probability with Martingales*. Cambridge University Press.

Winzen, Carola (2011). "Toward a complexity theory for randomized search heuristics : black-box models". In: *Adviser: Mehlhorn, Kurt. PhD thesis. Germany, Germany: Max-Planck-Institut für Informatik*.

Witt, Carsten (2006). "Runtime analysis of the (μ+ 1) EA on simple pseudo-Boolean functions". In: *Evolutionary Computation* 14.1, pp. 65–86.

– (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.

– (2019). "Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax". In: *Algorithmica* 81.2, pp. 632–667.

Wolpert, D.H. and W.G. Macready (2005). "Coevolutionary Free Lunches". In: *IEEE Transactions on Evolutionary Computation* 9.6, pp. 721–735.

Wolpert, David H (2002). "The Supervised Learning No-Free-Lunch Theorems". In: *Soft computing and industry: Recent applications*, pp. 25–42.

Wolpert, David H and William G Macready (1997). "No Free Lunch Theorems for Optimization". In: *IEEE Transactions on Evolutionary Computation* 1.1, pp. 67–82.

Xue, Ke, Ren-Jian Wang, Pengyi Li, Dong Li, HAO Jianye and Chao Qian (2023). "Sample-efficient quality-diversity by cooperative coevolution". In: *The Twelfth International Conference on Learning Representations*.

Yang, Zhenyu, Ke Tang and Xin Yao (2008). "Large scale evolutionary optimization using cooperative coevolution". en. In: *Information Sciences* 178.15, pp. 2985–2999.

Yao, Andrew Chi-Chin (Oct. 1977). "Probabilistic computations: Toward a unified measure of complexity". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 222–227.

Yo, Ting-Shuo and Edwin D De Jong (2007). "A comparison of evaluation methods in coevolution". In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pp. 479–487.

Zhang, Brian Hu and Tuomas Sandholm (2024). "Exponential Lower Bounds on the Double Oracle Algorithm in Zero-Sum Games". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. International Joint Conferences on Artificial Intelligence Organization.

Zhang, Guojun and Yaoliang Yu (Mar. 2020). "Convergence of Gradient Methods on Bilinear Zero-Sum Games". en. In: *arXiv:1908.05699 [cs, math, stat]*. arXiv: 1908.05699.

Zhang, Yihua, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang and Sijia Liu (2022). "Revisiting and advancing fast adversarial training through the lens of bi-level optimization". In: *International Conference on Machine Learning*. PMLR, pp. 26693–26712.

Zheng, Weijie, Yufei Liu and Benjamin Doerr (2022). "A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II)". In: *AAAI*.