Continuous Black-Box Optimization: Samplings and Dynamic Environments

AU, Chun Kit

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
May 2015

Thesis Assessment Committee

Professor WONG Man Hon (Chair)

Professor LEUNG Ho Fung (Thesis Supervisor)

Professor LEUNG Kwong Sak (Committee Member)

Professor Ting H.F. (External Examiner)

Abstract of thesis entitled:

Continuous Black-Box Optimization: Samplings and Dynamic Environments

Submitted by AU, Chun Kit

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in May 2015

Numerical optimization is one of the most active research areas and is widely used in science, engineering, economics and industry. In numerical black box optimization, the underlying objective functions, which can be non-differentiable, non-convex, multi-modal and noisy, are unknown to optimization algorithms. At any points in the continuous search space, the first and second order information is not available. Only the objective function values are available by means of function evaluations. The optimization algorithms, which consider these optimization problems as a black box, are designed to find the best solutions in the continuous search space.

This thesis focuses on continuous black box optimization and presents a collection of the novel sampling methods improving the state-of-the-art optimization algorithms. The algorithms are Covariance Matrix Adaptation Evolution Strategies (CMA-ES) and Cooperative Coevolutionary Algorithms (CCEAs). We will also study these algorithms in the dynamic environments where the objective functions change during the course of optimization. It is necessary to understand algorithms' performances because many real world problems are basically dynamic in nature. Examples of real world problems include but not limited

to random arrival of new tasks, machine faults and degradation, climate change, market fluctuation and economic factors.

In the first part of this thesis, two novel sampling methods that improves Evolution Strategies (ES) for continuous black-box optimization will be introduced: halfspace sampling and eigenspace sampling. In *Halfspace Sampling*, the hyperplane which goes through the current solution separates the search space into two halfspaces: a positive halfspace and a negative halfspace. When a candidate solution is sampled, the sample always lies in the positive halfspace that is estimated by successful steps in the recent iterations. We theoretically derive the log-linear convergence rates of a scale-invariant step size ES when ES are used to optimize spherical functions in finite and infinite dimensions. Halfspace sampling is implemented in a (1+1) CMA-ES, and the resulting algorithm is benchmarked on the Black-Box Optimization Benchmarking (BBOB) testbed. In *Eigenspace sampling*, the optimization algorithms consider the eigenspace of the underlying objective functions. A candidate solution is always sampled in an eigenspace spanned by eigenvectors with repeated or clustered eigenvalues. This demonstrates experimentally how eigenspace sampling can improve the CMA-ES for the current benchmark problems, In particular, the CMA-ES that uses eigenspace sampling often performs very well in ill-conditioned problems.

In the second part of this thesis, we will study the CMA-ES, ES and CCEA in *dynamic environments*. Two new types of individuals that address the dynamic environments will be introduced: 1) random immigrants (RIs) that increase the diversity for the changing environments, and 2) elitist individuals that improve the local convergence to the optima. The resulting algorithms are evaluated on a standard suite of benchmark problems. Superior results are observed when the two types of individuals are used. We also investigate the behavior of three CMA-ES

variants, which include an elitist (1+1)-CMA-ES, a non-elitist (μ,λ)-CMA-ES and a sep-CMA-ES. Our experimental results show the simple elitist strategies that include the (1+1)-ES and the (1+1)-CMA-ES generally outperform non-elitist CMA-ES variants. The elitist strategies are robust to dynamic changes with different severites, but performance is worsened when the problem dimensions are increased. In higher dimensions, the performance of elitist and non-elitist variants of CMA-ES are marginally identical.

論文摘要
香港中文大學工程學院
計算機科學及工程學系
哲學博士
區俊傑

「連續函數最優化」乃係研究領域中一項重要之議題。其廣泛被應用於科學、工程、經濟及工業等範籌。於黑箱最優化下，「優化算法」往往未能掌握目標函數之基本特性如可微分性，非凸性，多模態性及雜訊，甚至未能完全準確地使用連續函數之基本第一階及第二階導數。它們只能通過目標函數值來評估優化之進度。因此，設計最佳優化算法實為一個既困難且富挑戰性之研究議題。

本論文主要探討「連續函數黑箱最優化」，並提出新穎之抽樣方法，分別為「協方差矩陣適應進化策略」（CMA-ES）及「協同進化算法」（CCEAs），以改善現時研究領域中最佳之優化算法。本文亦深入了解此算法於動態環境中之優化表現。於應用科學中，如氣候變化，市場波動及經濟因素等等問題於本質上是動態，因此了解它們於動態環境中之優化表現是十分重要。

本論文第一部分提出兩種新抽樣方法，分別為「半空間抽樣法」及「特徵空間抽樣法」，作為提高「進化策略」（ES）之最優化表現。一、於「半空間抽樣法」下，連續函數之搜索空間會被一片超平面分割成兩個半空間，分別是「正半空間」及「負半空間」。當優化算法尋找一個元素時，它每一次會從正半空間裏面抽樣。本文由此推算出進化策略在球形函數收斂速度，並將之應用於最先進之(1+1) CMA-ES，從而測試它在最新之BBOB平台之表現。二、於「特徵空間抽樣法」下，優化算法首先在擁有重複或集群特徵值之特徵空間內抽出一個元素，然後將特徵空間抽樣法應用於CMA-ES。實驗結果發現使用特徵抽樣法能提升CMA-ES在俗稱「病態」函數中之優化表現。

本論文第二部分探討三個優化算法—「協方差矩陣適應進化策略」、「高進化策略」及「協同進化算法」於動態環境中之表現。本文提出以兩種新元素作處理動態環境—「隨機元素」(Random Immigrants)及「優生元素」(Elitist)。「隨機元素」用以增加優化過程中元素之多樣性；「優生元素」則作增强局部優化之收斂速度。另外，本文更使用最新之測試平台評估三個CMA-ES優化算法之表現，包括(1+1) CMA-ES、(μ,λ) CMA-ES及sep CMA-ES。實驗結果證明，簡單精英進化策略，如(1+1) ES 及(1+1) CMA-ES，普遍比非精英進化策略更能於動態環境作優化表現。精英進化策略較能應付不同程度轉變之動態環境。相反，當該連續函數之維度增加時，進化策略之優化表現則開始下降。於高維度下，精英CMA-ES及非精英CMA-ES之優化表現大致相同。

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Professor Ho-Fung Leung, for his great support and guidance throughout my study. I would like to thank him for accepting me as a PhD student in the Department of Computer Science and Engineering at the Chinese University of Hong Kong (CUHK). I am grateful for his advice back in October 2005 when I first expressed my wish to study a PhD degree. From then on, he has fully supported me in many aspects of my research. Working towards a part-time PhD degree has not been easy. It is particularly difficult for someone like me who has family and must work during the day as the time I could spend on research was very limited. I would also like to thank him for his patience during my study, as without the support of Professor Leung, I would have been unable to achieve what I have accomplished in this thesis.

I would like to thank my examiners, Professor Kwong-Sak Leung, Professor Man-Hon Wong and Professor Ting Hing-Fung for spending time examining my thesis. Their advice has not only improved the quality of this thesis, but also inspired me with new knowledge. Being inspired is always a key element for a researcher.

I am grateful for the financial support from the Department of Computer Science and Engineering for all conferences during my study. I would also like to thank the Department for providing their excellent research environment and facilities.

Special thanks to Dr. Dickson Chiu and Professor Wai-Kuen Cham, who have provided me with their valuable advice on my PhD study before my admission to CUHK.

Finally, I would like to express gratitude to my family: my wife Chu-Ji as well as my sons Ching and Kwong. You are my source of happiness and confidence, and I can scarcely believe how lucky I am. Thank you for everything, and you will always be my beloved ones.

Chun-Kit Au
31 August 2014
Hong Kong.

To Chu-Ji, Ching and Kwong.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Knowing others is intelligence; knowing yourself is true wisdom. Mastering others is strength; mastering yourself is true power.

Lao Tzu

1.1 Motivations

The complexity of real-world problems have been increasing rapidly. This demands the need for more powerful simulations, modelling frameworks, as well as algorithms. Although computational power has been exponentially growing in recent years, there is still a permanent demand for robust algorithms to solve these real-world problems. The resolutions of sophisticated real-world problems are always expensive in terms of time and money. Many common real-world problems require expensive function evaluations even for a single candidate solution. As a result, there is a strong demand for designing robust algorithms which can efficiently take advantage of increasing computational power.

Among these real-world problems, optimization problems have been one of the most important topics in various disciplines, including computer science, engineering and economics. Theoretical computer science has brought forward a rich set of compli-

cated algorithms to solve optimization problems, especially with regard to continuous optimization. In continuous optimization, the variables of the underlying problems are in a domain that is typically a set of real numbers. Searching for the optimal candidate solutions is always difficult, because 1) objective functions always have the nonlinear properties and multiple local optima; 2) the derivatives of the objective functions are not easily computed. More importantly, most of these optimization problems is black box problems in the sense that there are no available explicit mathematical formulation. As a result, designing well-performing optimization algorithms is a highly active research area.

Algorithms for solving continuous optimization problems include the use of analytical methods and the design of approximation algorithms. In analytical methods, an exact global or local optimal solution is guaranteed and is theoretically proven for the optimality of the solution when sufficient time is given [164, 91]. However, these methods always require either exhaustive searches or symbolic mathematical computations which make them become impractical for complex real-world optimization problems. On the other hand, the approximation algorithms are algorithms which approximate the solutions for optimization problems. Common methods include derivative based approximation method [206], heuristic algorithms [113] and direct search methods [60]. Derivative based approximation methods include steepest descent [35], conjugate gradient method [206] and Newton method [167]. They must evaluate the objective functions and its respective derivative information, which are often unavailable in real-world optimization problems. Direct search methods, including simplex method [149], Powell's method [177] and pattern search [213], do not require derivative information. Yet they are developed in the domains of mathematically programming.

In this context, evolutionary algorithms (EA) are thoroughly investigated because they can solve complex optimization problems by using simple problem-specific variation operators and selection operators. The very early evolutionary algorithm is genetic algorithms (GA) and was established in the United States by Holland [110]. The genetic algorithms at that time generated a binary encoding for the problem. Mutation, recombination, and selection were applied to a multi-set "population" of solutions in an iterative process. Selection is done according to the "fitness" of a solution, i.e. the objective value of a maximization problem. Given sufficient time, the process is able to evolve good solutions with high fitness, which is similar to natural evolution. In about the same time, evolution strategies (ESs) were founded in Germany [184, 198, 199]. Unlike genetic algorithms which heavily relied on recombinations, evolution strategies used mutations only and were formulated for optimization in continuous space. Nowadays, both genetic algorithms and evolution strategies are known by the name of evolutionary algorithms and are a highly active and vivid research area known as Evolutionary Computation.

Evolutionary algorithms have been applied to a wide variety of practical real-world or artificial problems. The popularity of evolutionary algorithms can be explained by their implementation. In addition, we can apply complex settings to these algorithms for optimization problems which are not well understood. In extreme cases, optimization problems are a black box to the optimization algorithms. Information and knowledge of the problems can only be obtained by simulations or evaluations of candidate solutions. In this context, evolutionary algorithms can easily be applied to these problems where time and computational resources are constrained, and when knowledge of the problem is limited. Specifically, evolutionary algorithms can address a wide range of optimization problems with a variety

of properties: unimodal or multi-modal objective functions; ill-conditioned optimization problems; noisy or noiseless objective functions; large scale optimization problems; constrained optimization problems; stationary or dynamic optimization problems and multiple objective optimization problems.

The Evolutionary Computation community has been developing a variety of approaches and methods to address all types of optimization problems. Theoretically, according to the No Free Lunch Theorem [227], we cannot always expect evolutionary algorithms to be a general-problem algorithm which can outperform any problem-specific algorithms. These results underline the capabilities as well as the limitations of evolutionary algorithms. From a theoretical perspective, the fundamental research on simple algorithms for simple problems can demonstrate the usefulness of evolutionary algorithms on more solid theoretical foundations. From the practical perspective, some evolutionary algorithms are known to be the most efficient ones with respect to the current benchmark or real-world problems. However, some other evolutionary algorithms are considered to be more general, more flexible or more elegant.

The main goal of this thesis is to design new techniques and configure them in high performing algorithms for continuous optimization. In recent years, practitioners have developed a wide variety of different extensions to evolutionary algorithms. Instead of inventing completely new algorithms for continuous optimization, this thesis focuses on the enhancements of the current existing promising methods. By systematically engineering new variants of these optimization algorithms, we show that we can improve significantly their performances and apply them to new domains, particularly on sets of problem classes with problem-specific properties. One such enhancement is the use of new sampling methods where evolutionary algorithms sample their new candidate solutions. Samples in evolutionary algo-

rithms are always *random* and *independent* of each other. We believe that evolutionary algorithms on optimization problems with problem-specific properties, which employs a degree of randomness as parts of its logics, can be further improved by *derandomizing* their randomness. One of the popular derandomization methods is to replace the independent samples with *dependent* ones: sampling methods for new candidate solutions can be dependent on the previous candidate solutions which are in a good quality. In addition, this shows that the derandomization of evolutionary algorithms work particularly well on some problem-specific properties.

Another goal of this thesis is to study the state-of-the-art evolutionary algorithms (EA) for dynamic optimization problems. Dynamic optimization problems are an active research topic and have increasingly attracted interest from the Evolutionary Computation community. This research area is relatively young as most of the studies have only been made in the last few years. Therefore, there still has many open areas with open research questions, on which one of the most important questions is about how well we understand the state-of-the-art evolutionary algorithms perform for dynamic optimization problems. The main purpose of this thesis is to investigate this important question and to experimentally understand the limits and capabilities of the state-of-art evolutionary algorithms for different classes of dynamic optimization problems.

1.2 Main Contributions

In general, the contributions presented in this thesis can be divided into two parts:

1. In the first part (Chapters 4 to 6), we propose two new sampling methods: *halfspace sampling* and *eigenspace sampling*. We will study the behavior of the simple evolu-

tion strategies that use these two sampling methods and apply these methods to the efficient and prominent Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [103, 105] for single objective static optimization problems:

- Halfspace sampling in evolution strategies (Chapters 4);
- Halfspace sampling in evolution gradient search (Chapters 5);
- Eigenspace sampling in evolution strategies (Chapters 6).

2. In the second part (Chapters 7 to 8), we present two state-of-art evolutionary algorithms: Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [103, 105] and Cooperative Coevolutionary Algorithms (CCEA) [172]. Both algorithms and their variants are studied for dynamic environments and evaluated on a standard suite of the benchmark problems.

   - CMA-ES in dynamic environments (Chapters 7);
   - CCEA in dynamic environments (Chapters 8);

In details, we highlight the contributions of each chapter as follows:

Halfspace Sampling in Evolution Strategies

A novel halfspace sampling method is proposed in a single parent elitist ES for unimodal functions. In halfspace sampling, the supporting hyperplane that goes through a parent separates the search space into a positive halfspace and a negative halfspace. If an offspring lies in the negative halfspace, it will be reflected with respect to the parent so that the offspring lies in the positive halfspace. The convergence rates of a scale-invariant step

size (1+1)-ES with halfspace sampling are derived on spherical functions that are in finite and infinite dimensions. It is also proven that the lower bounds of convergence rates are improved by a factor of 2 when strategies sample their offspring in the optimal positive halfspace. Halfspace sampling is implemented in a (1+1) CMA-ES by introducing the concept of evolution halfspaces. Evolution halfspaces accumulate the significant information of the previous successful and unsuccessful steps in order to estimate the optimal positive halfspace. The (1+1)-CMA-ES with halfspace sampling is benchmarked on the Black-Box Optimization Benchmarking (BBOB) noise-free testbed and experimentally compared with the standard (1+1)-CMA-ES.

Halfspace Sampling in Evolution Gradient Search (EGS)

The log-linear convergence of the Evolution Gradient Search (EGS) with halfspace sampling is proven theoretically. When halfspace sampling is used in the EGS, the search space is divided into two halfspaces with respect to a parent. A random sample is used only when it lies in the positive halfspace. Otherwise, its reflection with respect to the parent is used. All resulting random vectors in an iteration are used to estimate the optimal halfspaces in order to improve the local performance. The log-linear convergences of the scale-invariant step size EGS with and without halfspace sampling are proven. The convergence rates are expressed in terms of expectations of random variables. By means of Monte-Carlo simulations, we numerically computed the convergence rates and compared the lower bounds in finite and infinite dimensions, for different numbers of offspring. The EGS with halfspace sampling always converges faster than the EGS without halfspace sampling. An improvement of 42% to 68% is observed asymptotically.

Eigenspace Sampling in Evolution Strategies

A modification to the standard CMA-ES is proposed. The resulting algorithm is called ϵ-CMA-ES. The objective of the modification is to improve the CMA-ES by reducing the number of function evaluations needed for covariance matrix adaptation. To improve the time required for covariance matrix adaptation, the ϵ-CMA-ES identifies the minor eigenspace in the Hessian matrix of the underlying objective functions, which has repeated or clustered eigenvalues. The ϵ-CMA-ES always evaluates all the directions of the dominant eigenspace and the dominated eigenspaces. It also randomly evaluates each direction in the minor eigenspace. The ϵ-CMA-ES is investigated on a set of common unimodal benchmark problems, including ill-conditioned functions and functions that have a few repeated eigenvalues in their Hessian matrices. The advantages are most pronounced in objective functions with minor dominated eigenspace, such as the Cigar function, Tablet function and Twoaxes function. In these functions, the variances of the mutation distribution in the directions of the minor eigenspace are reduced much faster where the ϵ-CMA-ES randomly evaluates the minor eigenspace. However, limited benefits are observed for other objective functions when the eigenspectra are evenly distributed, such as the Ellipsoid function and the Rosenbrock function. When the problem dimension is 80, the improvement in the ϵ-CMA-ES ranges from none (on the Ellipsoid and Rosenbrock functions) to more than 40% (on Twoaxes function).

CMA-ES in Dynamic Environments

The state-of-the-art CMA-ES variants for dynamic optimization are empirically studied. The variants include the elitist (1+1)-CMA-ES, the standard (μ,λ)-CMA-ES and the sep-(μ,λ)-CMA-ES. We first briefly review the CMA-ES variants in the context

of static optimization, and discuss the latest dynamic optimization benchmark problems that are used in our simulations. In one out of the six dynamic functions, the elitist (1+1)-ES with the one-fifth rule and the (1+1)-CMA-ES have the best performance. These two elitist strategies are statistically equivalent in our simulations. The non-elitist strategies, including the standard (μ,λ)-CMA-ES and the sep-CMA-ES, are outperformed by the elitist variants. The results are consistent with dynamic changes with different severities. However the elitist strategies, which are point-based search algorithms, underperform for higher dimensional problems. The population-based strategies, including the standard (μ,λ)-CMA-ES and the sep-CMA-ES, perform the same as the elitist (1+1) variants do.

CCEA in Dynamic Environments

The behaviour of CCEAs on the state-of-art dynamic optimization benchmarks is also investigated. We first review the background of the CCEAs for static optimization, and then discuss the four major approaches used in EAs for the dynamic optimization. One major difference between CCEA individuals and EA individuals is that the CCEA individual has to collaborate with another $n-1$ CCEA individuals for its fitness evaluation. We formally discuss the two major collaboration methods used in the CCEAs: 1) the best collaboration method, in which the CCEA individual always chooses the best individuals in terms of fitness, and 2) the random collaboration method, in which the CCEA individual always randomly select other individuals without considering their fitness. The previous study shows that using the best collaborations for static optimization always yields the best performance. We extend this study to the context of dynamic optimization, and investigate whether the choices for collaboration methods is the same. Our simulation results show that the CCEAs using the best collaboration method outper-

form the CCEAs using the random collaboration method. The results are consistent with the types of dynamic changes and the problem dimensions.

1.3 Thesis Outline

Chapter 2 introduces continuous black box optimization problems. We summarize the common properties that can make the black box optimization difficult. The methods that evaluate optimization algorithms are also discussed. We review the benchmark testbeds in the literature enabling us to compare various optimization algorithms. Lastly, we discuss continuous black box optimization in dynamic environments, their common properties, the methods of evaluations as well as the benchmark testbeds in the literature.

Chapter 3 reviews the evolutionary algorithms studied in this thesis. Firstly, the history of evolutionary computations is reviewed, followed by a comprehensive overview on variants of evolutionary algorithms. Secondly, we review the details in the family of evolution strategies, including the single parent elitist (1+1)-ES and populated non-elitist (μ,λ)-ES and CMA-ES. Lastly, we review CCEAs which are commonly used for large scale optimization.

Chapter 4 presents the details of halfspace sampling and is divided into two parts. The first part focuses on the use of halfspace sampling in a simple scale-invariant step size (1+1)-ES. The second part focuses on its practical implementation in a (1+1)-CMA-ES. The log-linear convergence of a scale-invariant step size (1+1)-ES with halfspace sampling is derived and simulated by means of Monte-Carlo methods.

Chapter 5 investigates in theory the gain brought by halfspace sampling to EGS. The linear convergence of EGS with scale-invariant step sizes is proven on spherical functions. Con-

vergence rates of EGS with and without halfspace sampling in finite and infinite dimensions are derived. By means of Monte-Carlo simulations, it is shown that the lower bounds of convergence rates of EGS with halfspace sampling are better than those without halfspace sampling, regardless of dimensionality and the number of offspring.

Chapter 6 presents the details of eigenspace sampling. It first defines the eigenspace in the search space and discusses the motivations of eigenspace sampling. Then the modifications to the (μ,λ)-CMA-ES $(1,\lambda)$-CMA-ES are presented. The experimental results on convex quadratic functions are also presented and discussed. Some suggestions on the use of eigenspace sampling are also given.

Chapter 7 empirically investigates the state-of-the-art CMA-ES variants in the literature and studies their performance for dynamic optimization. The variants of the CMA-ES are reviewed. Then, the dynamic optimization benchmark problems in our simulations are discussed, and the empirical comparisons are presented. Additionally, possible future works on CMA-ES variants for dynamic optimization are reviewed.

Chapter 8 studies CCEAs for dynamic optimization. The background of CCEAs is reviewed, and the uses of two new types of individuals in CCEAs are described. The benchmark problems, the experimental setup and the parameter settings in our simulations are described. The experimental results and the scores in the benchmark problems are reported.

Chapter 9 concludes this thesis, summarizes our contributions and outlines future research directions.

1.4 Related Publications

This thesis is based on the following publications.

Referred Journal

- **Chun-Kit Au**, Ho-Fung Leung. Cooperative Coevolutionary Algorithms for Dynamic Optimization: An Experimental Study. Springer Evolutionary Intelligence. [12] (http://dx.doi.org/10.1007/s12065-014-0117-3)

Refereed conference papers

- **Chun-Kit Au**, Ho-Fung Leung. Halfspace Sampling in Evolution Strategies. To appear in Genetic and Evolutionary Computation Conference (GECCO) 2014, 12-16 July 2014, Vancouver, BC, Canada. [17] (http://doi.acm.org/10.1145/2576768.2598335)

- **Chun-Kit Au**, Ho-Fung Leung. An Empirical Comparison of CMA-ES in Dynamic Environments. In Proceedings of Parallel Problem Solving from Nature PPSN XII - 12th International Conference, 1-5 September 2012, Taormina, Italy. [16] (http://dx.doi.org/10.1007/978-3-642-32937-1_53)

- **Chun-Kit Au**, Ho-Fung Leung. Eigenspace sampling in the mirrored variant of (1,λ)-CMA-ES. In Proceedings of IEEE World Congress on Computational Intelligence (WCCI) 2012, 10-15 June 2012, Brisbane, Australia. [11] (http://dx.doi.org/10.1109/CEC.2012.6256650)

- **Chun-Kit Au**, Ho-Fung Leung. Improving CMA-ES by Random Evaluation on Minor Eigenspace. In Proceedings of IEEE World Congress on Computational Intelligence (WCCI) 2010, 18-23 July 2010, Barcelona, Spain. [15] (http://dx.doi.org/10.1109/CEC.2010.5586553)

- **Chun-Kit Au**, Ho-Fung Leung. Investigating the Collaboration Methods of Random Immigrant Scheme in Coop-

erative Coevolution. In Proceedings of IEEE Congress on Evolutionary Computation (CEC) 2009, 18-21 May 2009, Trondheim, Norway. [14] (http://dx.doi.org/10.1109/CEC.2009.4983281)

- **Chun-Kit Au**, Ho-Fung Leung. On the Behavior of Cooperative Coevolution in Dynamic Environments. In Proceedings of IEEE World Congress on Computational Intelligence (WCCI) 2008, 1-6 June 2008, Hong Kong. [13] (http://dx.doi.org/10.1109/CEC.2008.4631177)

□ **End of chapter.**

Part I

Background

Chapter 2

Continuous Black Box Optimization

Pure mathematics is, in its way, the poetry of logical ideas.

Albert Einstein

2.1 Continuous Black Box Optimization

2.1.1 Continuous Optimization

In optimization, we need to find the best solutions to optimization problems. Formally, optimization is defined as finding the *best solution* $\mathbf{x}^*$ in a search space $\mathcal{S}$, which satisfies:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{S}} f(\mathbf{x}) \tag{2.1}$$

where $f : \mathcal{S}\mapsto \mathbb{R}$ is called the *objective function* (also known as *fitness* function). The search space $\mathcal{S}$ and the objective function f together define the *optimization problem*. The equivalent formulation describes the maximization case where we need to search $\mathbf{x}$ that maximizes f. In that case, the objective function is often called the *cost function*. The elements of the search space $\mathcal{S}$ composed of multiple variables are called *decision variables*.

Continuous optimization is the class of optimization problems, where $\mathcal{S}$ is a subset of $\mathbb{R}^n$ and n is the dimension of the search space. The term "continuous" refers to the continuous decision variables, i.e. the components of $\mathbf{x}$, and does not imply that the objective function f is continuous. The fundamental problem of continuous optimization is that the search space is infinite. Therefore, finding the best solution $\mathbf{x}^*$ is not guaranteed. In fact, many objective functions in the real-world applications are relatively smooth. This allows one to take advantage of it by exploiting the necessary information. For instance, the classical example is the gradient descent methods.

In the literature, there are other subfields of optimization, where their solution space are different from that of the continuous optimization problems. For instance, in *integer optimization*[76, 40, 195], the solution space is a subset of integers $\mathbb{Z}$. In *combinatorial optimization*[150, 163], the solution space is a finite set of objects. The classic combinatorial optimization problems are the Travelling Salesman Problem[127] and the Minimum spanning tree[88]. There are optimization problems where the solution space is *constrained*. In constrained optimization[38, 178], the solution space is bounded by a collection of equalities or inequalities which define the set of feasible solutions. The existence of constraints gives rise to challenges for optimization algorithms, especially when the best solution lies on the search space boundary.

2.1.2 Black Box Optimization

Originally, optimization was studied for objective functions that are known. In most cases, the optimum cannot be calculated analytically. In some cases, the objective functions themselves are unknown. This creates another subfield of optimization called *black box optimization*. The objective of *black box optimiza-*

tion is to find the *best* solutions, those with values closest to the optimum function value f_{opt}, with as few function evaluations as possible. In black box optimization, the optimization algorithms interpret the objective functions as a black box. No specific assumptions on objective functions are made. All information about the black box objective functions, except the dimension n, comes from evaluations of candidate solution $\mathbf{x}$ on f. The number of function evaluations is usually denoted as the *cost* of black box optimization.

Black box optimization is sometimes known as *direct search* [111, 123] or *derivative free optimization* [60], since the derivatives are not explicitly used. It is also closely related to the field of *metaheuristics* [83, 158, 211, 3] in which the goal is to find *sufficiently good* solutions on the assumptions of incomplete information or limited computation capacity. Many metaheuristics have implemented some form of *randomized search methods*, so the solution found depends on the set of random variables generated. Examples of randomized search methods include *Simulated Annealing* [215], *Particle swarm optimization* [120], *Ant colony optimization* [69] and *evolutionary computation* [30, 78, 31, 65].

2.1.3 Basic Properties

Several basic properties of objective functions characterize black box optimization and make it difficult for the designers of optimization algorithms. The following lists a few common properties that can be found in the literature. For details, readers can refer to the works [108, 109].

Multi-modality

A *local optimum* of an objective function f is an n-dimensional vector $\mathbf{x}^l \in \mathbb{R}^n$ with a *neighbourhood* $\mathcal{S}(\mathbf{x}^l)$ of $\mathbf{x}^l$ such that

$$\forall \mathbf{x} \in \mathcal{S}(\mathbf{x}^l) \wedge \mathbb{R}^n, f(\mathbf{x}^l) \leq f(\mathbf{x}) \tag{2.2}$$

A *global optimum* is a vector $\mathbf{x}^g \in \mathbb{R}^n$, such that there is no other $\mathbf{x}^l$ that is better than $\mathbf{x}^l$. Formally, we can write

$$\forall \mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}^g) \leq f(\mathbf{x}) \tag{2.3}$$

An objective function is *multi-modal* if it has more than one optimum. Multi-modal functions are difficult to be optimized because there is no guarantee of finding the global optima. Analytical methods are frequently not applicable, and the use of numerical solution strategies often results in major challenges. In order to increase the chance of finding the global optima, the designers of optimization algorithms always have to run several optimization runs or *restarts* within a single run. In terms of difficulty for optimization algorithms, multi-modal functions are usually more complicated than the unimodal one because there are usually a large number of basins of attractions, which are located around the local optima.

High Dimensionality

The volume of the search space increases exponentially with problem dimension n or with the number of decision variables. The term *curse of dimensionality* coined by Richard Bellman [36, 37] refers to the problems caused by the exponential increase in volume that is associated with adding extra dimensions to the solution space. The following example can illustrate the challenges of high dimensionality. Consider a one-dimensional real-value space, and we place 100 points into the space. If we were to achieve a similar coverage in a ten-dimensional real-value space, it would require $100^{10} = 10^{20}$ points. Consequently,

optimization algorithms that work in small dimensions may not be useful in the large dimensional problems.

It is also important to mention the dimension ranges of problems when we compare optimization algorithms. Many optimization algorithms do not scale up well for large dimensions, because the computational complexity and the memory requirement grow quickly with problem dimensions.

Non-separability

An objective function f is said to be *separable* if the optimum of the function f can be found by performing independent one-dimensional searches along each independent coordinate. An objective function f is said to be *non-separable* if the optimum of the function *cannot* be found by independent one-dimensional searches. It is said to be *partially separable* if f has a *groups of coordinates* that can be optimized separately. Many real-world problems are partially separable. One usually decomposes a problem into sub-problems so we can separately solve the sub-problems of the large partially separable problem. Decomposition is a very common approach for such complex optimization problems.

Ill-conditioning

A convex-quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$ is ill-conditioned if the condition number of $\mathbf{H}$ is much larger than 1, where $\mathbf{H}$ is the symmetric positive definite. The condition number of $\mathbf{H}$ is the ratio between its largest and smallest eigenvalue and is also equal to the squared rate between the longest and shortest principal axes of the ellipsoid $\{\mathbf{x}|\mathbf{x}^T\mathbf{H}\mathbf{x}\}$. The optimization algorithms have to perform different lengths of search steps in the respective searching directions of the decision space, in order to produce the same improvements for the objective function. In most cases,

the order of differences in the search steps can be very large.

Ill-conditioned problems are difficult for optimization because we often make too short or too long search steps in all directions. Variable metric methods such as quasi-Newton methods BFGS [54, 55, 75, 87, 202] learn the inverse of the Hessian of f and are able to renormalize the search steps. However, these algorithms often estimate the gradient using finite-difference methods. There are numerical instabilities to round-off the errors when the condition number is large. Evolutionary variable-metric algorithms such as CMA-ES [105, 103] are usually more stable because they do not need to estimate the gradient to learn an appropriate metric.

Dynamic

An objective function $f(\mathbf{x}, t)$ is *dynamic* if the objective values of candidate solutions depend on the time-step t. Basically, it means that the objective function changes over time. The change can be as simple as a shift or a rotation of the search space. In *dynamic environments* where the objective function f is changing, the objective is to find the global optima of f for a time-step t and to predict the global optima at the time steps when $T > t$. In practice, this is always difficult to do. Therefore a less ambitious objective is to track the moving optimum in the space as closely as possible [118, 10, 8]. The dynamic objective functions are usually difficult for black box optimization. This is especially true, when the optimization algorithms have explored parts of the search space and excluded those for later stages of optimization, which is naturally normal for static objective functions. Therefore, it is necessary to develop robust methods which handle the dynamic of these dynamic objective functions in order to find the best solutions in time.

Noisy

In many real-world problems, it is unrealistic to assume that we can measure the objective function f precisely as it can be corrupted by measurements or noises. An objective function $\tilde{f}$ is *noisy* if different function values $\tilde{f}(\mathbf{x})$, which are perturbed by random component ξ, are observed. Optimizating the noisy $\tilde{f}$ is called *noisy optimization* [7, 9]. Generally, there are two cases for noisy optimization. In *multiplicative noises*, the variances of the noises decrease to zero when approaching the optimum. Mathematically, it can be formally defined as $\tilde{f}(\mathbf{x}) = f(\mathbf{x})(1 + \xi)$. The other case is to have *additive noise* in the objective functions where the variances of the noises are lower-bounded. Formally, it is defined as $\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \xi$. Both cases make optimization difficult because the information obtained from one function evaluation is less precise than the one from noise-free objective functions.

Multi-objective

An optimization problem is *multi-objective* if there are m objectives $f_i(\mathbf{x})$ for $i = 1, \ldots, m$, which have to be optimized. The desirable output of multi-objective optimization is usually a Pareto set of optimal solutions[1], instead of a single solution. Multi-objective optimization [67] is also called *multiple criteria decision making*[240] in some context and is generally difficult. This is because the multi-objective problem inherits similar properties from the single-objective optimization. The optimization itself needs to take into account the criteria of all multi-objective, which is less obvious than in the single-objective case.

[1] A Pareto set is the set of parameterizations or allocations that are all Pareto efficient.

2.1.4 Evaluating Optimization Algorithms

Performance Measure in Practice

There are two performance measures that we can practically evaluate and compare different optimization algorithms in the same continuous black-box optimization problems:

1. The *expected total number of function evaluations* to find the optimum.
2. The *expected total computational time* to find the optimum.

In all of the above cases, the optimum does not need to be the global optimum $\mathbf{x}^g$. It can be a local optimum $\mathbf{x}^l$ or an ϵ-approximation of the optimum, *i.e.*, $f(\mathbf{x}^*) \leq f(\mathbf{x}^\epsilon) + \epsilon$. For some applications, one performance measure will be more useful than others. In most cases, one assumes that the function evaluations are mostly expensive and are used as the performance measure of optimization algorithms. In addition, using the first performance measure is simple. It is independent of the implementation and hardware used by optimization algorithms. The second performance measure is useful when one wants to analyze the computational complexity for a single iteration of optimization algorithms, especially when algorithms are applied to optimize problems which are *high-dimensional* or *large scale*[135, 205, 82].

Convergence Analysis

In the theoretical aspect, we can compare and evaluate algorithms by studying the *convergence order* of optimization algo-

rithms by:

$$\lim_{k\to\infty} \frac{|f(\mathbf{x_{k+1}}) - f(\mathbf{x}^*)|}{|f(\mathbf{x_k}) - f(\mathbf{x}^*)|^q} = \mathrm{CR_f} \tag{2.4}$$

$$\lim_{k\to\infty} \frac{||\mathbf{x_{k+1}} - \mathbf{x}^*||}{||\mathbf{x_k} - \mathbf{x}^*||^q} = \mathrm{CR_x} \tag{2.5}$$

where the rates of convergence $\mathrm{CR_f}$ and $\mathrm{CR_x}$ are $0 \leq \mathrm{CR_f}, \mathrm{CR_x} < 1$, and $\mathbf{x}_k$ denotes the candidate solution in iteration k. Notice the difference of two equations: the first one focuses on the convergence rates in terms of function values; the second one focuses on the norm distance of $\mathbf{x}$ to the optimum. On different values of q, we can define different *orders of convergence*. For instance, if $q > 1$, then it is a *superlinear convergence*. If $q = 1$ and $\mathrm{CR_f}, \mathrm{CR_x} \in (0, 1)$, it is a *linear convergence*. If $q = 1$ and $\mathrm{CR_f}, \mathrm{CR_x} = 1$, it is a *sublinear convergence*

2.1.5 Benchmarking Function Testbeds

Evaluating and comparing optimization algorithms in a single problem is not sufficient for us to understand its strengths and weaknesses, and to determine its quality. One should never use the result out of a single problem to generalize for a large class of problems. This is why it is important to test and evaluate a single optimization on the common benchmarks and testbeds so the practitioners are able to determine which algorithms are the best suited to their optimization needs. In the literature, there are several benchmark function testbeds to achieve this goal.

CEC Real Parameter Optimization Testbed

In Congress on Evolutionary Computation (CEC) 2005, there was a session "Session on Real-Parameter Optimization (CEC-2005)". It focused on a more rigorous investigations of 11 Evolutionary Algorithms on 25 unimodal and multi-modal benchmark

problems [210]. Subsequent sessions were then hosted in CEC 2013 [133] and CEC 2014 [133]. The testbed has now evolved to have a total of 30 benchmark problems, 3 unimodal and 27 multimodal functions with problem dimensions up to 100. Each of the benchmark problem have different properties and make the optimization algorithms difficult and challenging for optimization.

Another similar testbed introduced in CEC, is for *large scale optimization* [212]. Its focus is on providing a systematic evaluation platform to compare the *scalability* of optimization problems. The benchmark problems in this testbed have dimensions up to 1000 which are difficult. In addition, the benchmark problems consist of problems which are separable, partially separable and non-separable. In fact, many real-world optimization problems are most likely be partially separable and consist of different groups of parameters, which have with strong dependencies within but little interactions between the groups. This is reflected in benchmark problems in order to ensure that the optimization algorithms are used in the same scenario.

The last two testbeds that are closely related to the continuous black box optimization and held in the Congress on Evolutionary Computation (CEC) is related to *constrained optimization* [134, 137] and real-world optimization [63]. The first one in [134, 137] aims towards benchmarking optimization algorithms when the optimization problems have different types of constraints in the solution space. The testbed initially had 24 benchmark functions in 2006 and in 2010, developed into benchmark functions that have separable or non-separable constraints. The second one includes in [63] consists of real-world problems that come from the problem domains like electromagnetic, power systems engineering, bioinformatics and computational biology. The aim is to get a better understanding of optimization algorithms in real-world problems because outstanding performances

on a set of artificial benchmarks may not guarantee a similar performance in every practical optimization problem.

GECCO Black-Box Optimization Benchmarking (BBOB)

COmparing Continuous Optimizers [98] (COCO) is a recent and successful benchmark testbed used to quantify and compare the performance of optimization algorithms in a scientifically decent and rigorous way. COCO provides a common platform to run optimization algorithms on a single-objective benchmark function testbed, generates data output, post-processes and then presents the results in graphs and tables. The practitioners can run their black box real-parameter optimzation algorithms in a few dimensions a few hundreds times and execute the provided post-processing scripts. COCO has been used for the Black-Box-Optimization- Benchmarking (BBOB) [98] workshops during the Genetic and Evolutionary Computation Conference (GECCO) in 2009, 2010, 2012 and 2013.

BBOB provides two testbeds which have 24 noiseless [100] functions and 30 noisy [101] functions. Each benchmark function has different properties: separable, non-separable, unimodal, multi-modal, ill-conditioned and deceptive. There are also functions with and without a weak global structure.

COCO uses the *expected running time*(ERT) as the performance measure. If an optimization algorithm succeeds in reaching the target precision value $f_{\text{target}} - f_{\text{opt}}$ in a single run, its runtime (RT) is the number of function evaluations used, where f_{target} and f_{opt} denote the target function value and the optimal function value respectively.If the algorithm fails, it can be restarted and run again. The ERT is the expected number of function evaluations to reach a target function value for the first time. The ERT [98] is defined as:

$$\text{ERT}(f_{\text{target}}) = \frac{\#\text{FEs}(f_{\text{best}} \geq f_{\text{target}})}{\#\text{succ}} \tag{2.6}$$

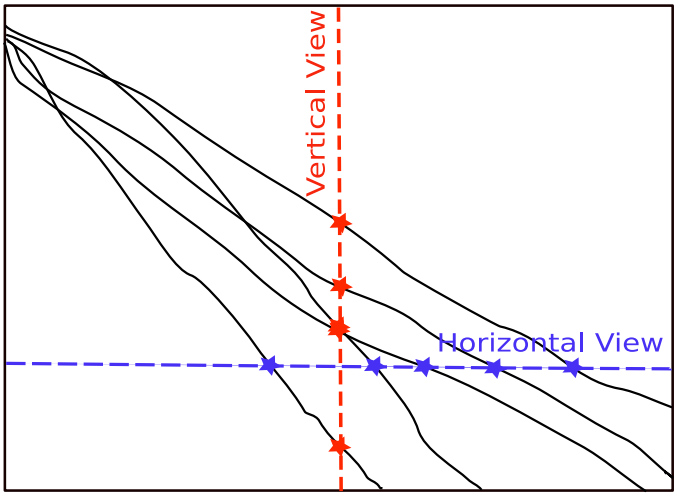

Figure 2.1: Illustration of fixed-cost (vertical cuts) and fixed-target (horizontal cut) view in [98]. Black lines depict the best function values plotted against the number of function evaluations.

where $\#\text{FEs}(f_{\text{best}} \geq f_{\text{target}})$ denotes the number of function evaluations conducted in all trials when the best function value f_{best} is not smaller than f_{target} during the trial. The #succ is the number of successful trials.

There are two approaches in COCO to collect data and make measurements from the output of experiments:

1. *Fixed-cost scenario (vertical cuts).* In this scenario, the number of function evaluations is fixed. This corresponds to fixing a cost for optimization and measuring the function values when the optimization algorithm has reached a given number of function evaluations. We can picture the scenario by drawing a vertical line on the convergence graph in Figure 2.1.

2. *Fixed-target scenario (horizontal cuts)*: In this scenario, the target function value is fixed. The number of function evaluations required to reach this target function value is also

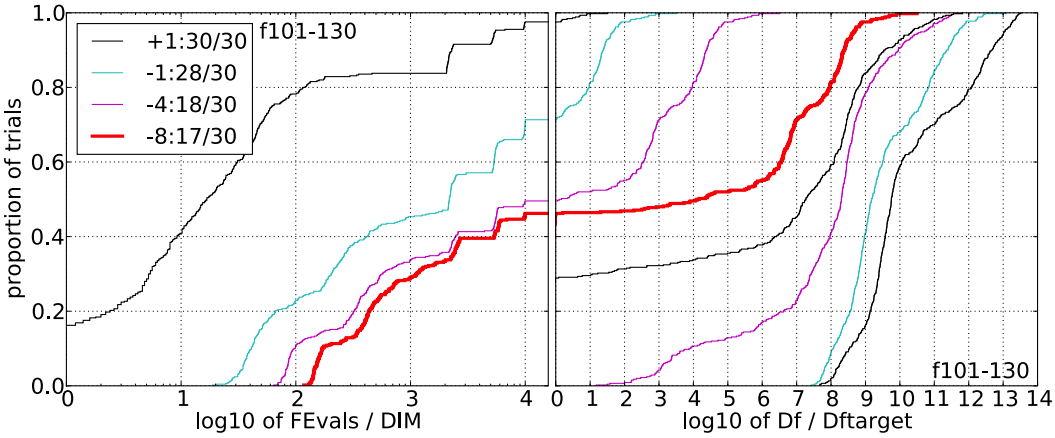

Figure 2.2: Illustration of empirical (cumulative) distribution functions (ECDF) [98] of running length (left) and precision (right) arising respectively from the fixed-target and the fixed-cost scenarios.

measured. This is shown by drawing an horizontal line in the convergence graph in Figure 2.1.

Empirical Cumulative Distribution Functions (ECDFs) are used in the COCO to summarize the results from a set of benchmark problems. The ECDF function $F : \mathbb{R} \mapsto [0, 1]$ is defined for a given set of real-valued data $\mathcal{S}$, such that $F(x)$ equals the fraction of elements in $\mathcal{S}$ which are smaller than x. The function F is monotonous and a lossless representation of the (unordered) set $\mathcal{S}$ [98]. For example, the thick red graph in Figure 2.2 shows on the left the distribution of the running length in terms of the number of function evaluations for Las-Vegas algorithms [112] for reaching precision $\Delta f = 10^{-8}$ (horizontal cut). The graph on the right shows the vertical cut for the maximum number of function evaluations, showing the distribution of the best achievedΔ f values, divided by 10^{-8}. The run length distributions on the left show different target precisionsΔ f. The precision distributions on the right show different fixed numbers of function evaluations. The y-value at the transition between the left and right subplots corresponds to the success probabil-

ity. In the example, there is about a 50% for precision 10^{-8} (thick red) and over 70% for precision 10^{-1} (cyan).

2.2 Dynamic Environments

2.2.1 Definition

In dynamic environments, the objective function changes during the course of optimization. At any given time $t \in \mathbb{T}$, one needs to find the *best solutions* $\mathbf{x}^*$ such that

$$\forall \mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}^*, t) \leq f(\mathbf{x}, t). \tag{2.7}$$

where $f : \mathbb{R}^n \times \mathbb{T} \rightarrow \mathbb{R}$ is the objective function of a minimization problem and n is the problem dimension. In dynamic optimization problems (DOPs), the fitness functions, the design variables and the environmental conditions change from time to time. Examples of real-world problems include random arrivals of new tasks, machine faults and degradation, climate change, market fluctuation and economic factors. Alternatively, DOPs can be defined as sequences of static problems linked together by some dynamic rules [5, 187, 188, 216, 221], or as problems that have time-dependent parameters in their mathematical expressions [28, 43, 226] without explicitly mentioning whether the problems are solved online by an optimization algorithm or not. It is important to distinguish the DOPs from general time-dependent problems because from the perspective of an optimization algorithm, a time-dependent problem is only different from a static problem if it is solved in a dynamic way. This results in optimization algorithms for DOPs having to take into account of changes during the optimization process [46, 145].

The simplest way to solve these dynamic optimization problems is to consider every change as an arrival of a new static optimization problem. However the time and the resources for

optimization are always limited, and using an explicit restart approach becomes unfeasible. Therefore, more complicated strategies to reduce the computational effort and to maintain the best qualities of the output solutions are required. One methods to address dynamic environments is to collect useful information on the dynamic environments. When the environment changes from time to time, optimization algorithms can collect the information in the previous environments, in a hope that the information is useful in the current environments. This assumes that the new environments are correlated to the previous environments. If the environments are correlated, it is reasonable to explore the search space "near" the previous optima. However, there is also a disadvantage for this approach. The success of the approach depends on the nature of dynamic problems. If the change is radical and the correlations between changes are low, the restart approach may be the only viable alternative and reusing the information in the previous changes may deteriorate the performance of the search algorithm. For most of the real-world problems, we usually assume that the changes are smooth. The two open questions are what kind of useful information the algorithms should collect and in what way the algorithms should use the collected information.

Using EAs for dynamic optimization problems has always been an active research area [46, 145, 233, 237, 218]. Many researchers always consider EAs as having a great potential to be the efficient optimizers for dynamic problems. In fact, there are many fundamental challenges for EAs. If an EA is being used in dynamic optimization problems, it has to adapt to the changing objective functions. The solutions found by an EA in the earlier time do not imply they are the same solutions at a later time. Therefore one major goal for the EAs is to continuously and efficiently adapt to the changing environment. This requires the EAs not only to find the global optima under a specific envi-

ronments, but also to continuously track the changing optima under different dynamic environments.

In the literature, there are works to develop the new approaches for dynamic optimization problems. The works [118, 153] summarize the major approaches:

- *Increasing diversity after dynamic changes*: When a change is detected, the EA will increase the diversity of its populations. Hypermutation [58, 144] is one of the approaches. The basic idea is to increase the mutation probability of the individuals so the candidate solutions can converge to the new global optima in a short time. One disadvantage for this approach is that increasing the diversity will replace the information of the previously successful individuals. It is also difficult to determine the optimal level of diversity. Having a large diversity will resemble a restart while too little diversity does not solve the problem in a desirable time.

- *Maintaining diversity throughout the runs*: Maintaining diversity during the course of optimization is important. Using random immigrants [89] is one of the methods to randomly generate individuals in every generation. Though the approach of maintaining diversity is interesting and simple to be implemented, having too high diversity can slow down the convergence rate to the optima.

- *Using memory approaches*: There are two main categories for the memory approaches. In the *explicit memory approach* [45, 236], the algorithms store the useful information in the previous generations. A pre-defined rules are designed so the algorithms know what information they have to store and what information they have to retrieved from the memory. The other category is the *implicit memory* [64]. It uses the redundant representations to store the

useful information in the previous generations. The diploid representations [86] are an example that attempts to mimic the diploid genes in the nature.

- *Using multi-populations approaches*: When an EA uses the multi-populations [225, 47, 235], it simultaneously tracks the multiple optima in the search space. When there is a change in the dynamic problem, the multi-populations are able to store the information for different regions of the search space. This can therefore increase the probabilities of finding the optimal candidate solutions in the new dynamic environments.

2.2.2 Properties

In the literature, there are many dynamic optimization problems that are used to compare different optimization algorithms. These dynamic test problems have different characteristics and can be classified into different groups. According to survey works by [153, 237], we can classify the DOPs into the groups below:

1. *Time-linkage*: The future behavior of the dynamic optimization problem depends on the current and/or the previous solutions found by the optimization algorithms. In the literature, there are a few of general purpose time linkage DOPs [43, 155] and problem-specific time linkage DOPs [155, 44].

2. *Predictability*: The changes in the dynamic optimization problems follow a regular pattern. For instance, the optima moves in fixed step sizes, periodically or in predictable time intervals.

3. *Visibility*: The changes in the dynamic optimization problems are visible to optimization algorithms. For instance,

optimization algorithms can re-evaluate the objective functions or the constraint functions in search space to detect whether a change has occurred. Most DOPs in the literature have the properties of visibility, except the work in [59, 214, 154, 186].

4. *Constrained problem*: The dynamic optimization problem can be constrained, and the constraints can change over time [154, 186].

5. *Multiple objectives*: The dynamic optimization problems can have multi-objective functions.

6. *Cyclic, Periodical or recurrent changes*: Changes in the dynamic optimization problems can be cyclic, periodical or recurrent.

7. *Changing Factors*: There are other factors that are changeable from time to time. They include objective functions, domain of variables, number of variables, constraints, or other parameters.

2.2.3 Performance Measures

Measuring the performance of optimization algorithms for DOPs is vital. According to the work [153, 237], we can classify the performance measures into *optimality-based performance measures* and *behaviour-based performance measures*.

Optimality-based performance measures measures the ability of optimization algorithms in finding the solution with the best objective/fitness values (fitness-based measures) or finding the solutions that are closest to the global optimum (distance-based measures). Common measures in this category includes:

1. Best-of-Generation [28, 58, 81, 89, 90];

2. Best-Error-Before-Change [214];

3. Modified Offline Error and Offline Performance [46, 48];
4. Relative error [73, 217]
5. Normalised Scores [152];
6. Distance-Based Measures [219, 194]

Behaviour-based performance measures are to evaluate optimization algorithms if they exhibit certain behaviours that are useful in dynamic environments. Examples of such behaviours are maintaining high diversity throughout the runs; quickly recovering from a drop in performance when a change happens, and limiting the fitness drops when changes happen. These measures are usually used complementarily with optimality-based measures when we analyze optimization algorithms in dynamic environments. Common measures in this category includes:

1. Diversity-based Measure [142, 157, 181, 232, 146, 46, 84];
2. Performance Drop after Changes [217, 181];
3. Convergence Speed after Changes [217, 154, 152];
4. Fitness Degradation over Time [4, 154]

In the following, we will review the two common performance measures that will be used in this thesis and analyze the scenario where one performance measure is preferable to others.

Best-of-Generation

This measure is calculated as the averages for the best objective values in each generation of the optimization problems. This measure has been used in many early research [28, 58, 81, 89, 90] and is still the most commonly used measure. Formally, the

work [234] describes this performance measure as

$$\bar{f}_{\mathrm{BOG}} = \frac{1}{k n_{\mathrm{run}}} \sum_{i=1}^{k} \left(\sum_{j=1}^{n_{\mathrm{run}}} f_{\mathrm{BOG}_{ij}} \right) \tag{2.8}$$

where $\bar{f}_{\mathrm{BOG}}$ is the mean best-of-generation fitness, k is the number of generations, n_{run} is the total number of runs, and $f_{\mathrm{BOG}_{ij}}$ is the best-of-generation fitness of the ith generation in the jth run on a particular problem.

The best-of-generation performance measures can enable the designers of optimization algorithms to quantitatively compare the performance of different algorithms. However, one drawback of using this measure is that they are not normalized and therefore they can be biased by the difference of the fitness landscapes at different periods of change. For example, if at a certain period of change the overall fitness values of the landscape is particularly higher than those at other periods of changes, the final $\bar{f}_{\mathrm{BOG}}$ might be biased toward the high fitness values in the particular period and hence might not correctly reflect the overall performance of the algorithm.

Best-Error-Before-Change

This measure [214] calculates the average of the best errors, i.e the difference between the optimum value and the value of the best individual at the end of each change period. It is also used for one of the performance measures in the CEC 2009 competition on dynamic optimization [128]. Formally, when there is T number of changes in a DOP, it can be described as

$$E_{\mathrm{best}} = \frac{1}{T} \sum_{i=1}^{T} |f(\mathbf{x}_{\mathrm{best}}(t_i), t_i) - f(\mathbf{x}^*(t_i), t_i)| \tag{2.9}$$

where the vector $\mathbf{x}_{\mathrm{best}}(t_i)$ is the best solutions found by the optimization algorithms at time t_i, and the vector $\mathbf{x}^*(t_i)$ is the location of the global optimum at time t_i.

This measure is useful in situations where we are interested to compare the final outcome of different algorithms found before the changes. However, there are drawbacks in this performance measure. First, the measure is not suitable if someone are interested in the overall performance of optimization algorithms whether there are changes or not. Second, this measure is also not normalised and is biased toward periods where the errors are relatively very large. Third, the measure requires that the global optimum is known.

2.2.4 Generalized Dynamic Benchmark Generator (GDBG)

Over the years, there have been a number of dynamic test problems to compare the EAs in dynamic environments. These include the moving peak benchmark (MPB) proposed by Branke [45], the DF1 generator proposed by Morrison and De Jong [143], the single and multi-objective dynamic test problem generator by Jin and Sendhoff[119] and exclusive-or (XOR) operator by Yang and Yao [229, 235, 236]. MPB and DF1 consist of multi-dimensional landscapes where the heights, the widths and the positions of the peaks can be changed during the course of optimization. Although a number of DOP generators exist in the literature, there is no unified approach of constructing dynamic problems across the binary space, real space and combinatorial space . The generalized dynamic benchmark generator (GDBG) [129, 128] constructs dynamic environments for all the three solution spaces to evaluate the performance of optimization algorithms. The benchmark problem was used in CEC 2009 and CEC 2014 competitions to evaluate the state-of-the-art algorithms for dynamic optimization. It differs from the MPB and the DF1 benchmarks, and uses the rotation method instead of shifting the positions of peaks. Using the rotation method can prevent the unequal challenge in every change when the posi-

tions of peaks bounce back from the boundary of the search space.

There are two categories for the dynamic changes in the GDBG: dimensional changes and non-dimensional changes. In the dimensional changes, GDBG increases or decreases the number of dimensions for the dynamic problems. In the non-dimensional changes, the variables are changed such that the fitness landscape is changed. An example is to increase the number of the peaks during the course of optimization. There are six types of the non-dimensional changes, including small step changes, large step changes, s random changes, chaotic changes, recurrent changes and recurrent changes with noise. We name them C_1 to C_6. The dimensional changes are generally regarded as difficult challenges for algorithms since there are no relationships between the dimensions when these dimensional changes occur.

Formally, we can describe the dynamic changes as follows:

$$\phi(t+1) = g(\phi(t), \Delta\phi) \tag{2.10}$$

where $\phi(t)$ is the system control parameters, $\Delta\phi$ is the derivation from the current system control parameters, and $g(\cdot)$ is the function to change the system control parameters. At time t, the new environment at time $t+1$ can be expressed as:

$$f(x,\phi\,,t+1) = f(x, g(\phi(t), \Delta\phi), t) \tag{2.11}$$

The optima of dynamic problems are determined by the system control parameters and they can be different from one instance at time t to another instance at time $t+1$. The six non-dimensional changes are:

- C_1 Small step change:

$$\Delta\phi = \alpha \cdot ||\phi|| \cdot r \cdot \phi_{severity} \tag{2.12}$$

- C_2 Large step change:

$$\Delta\phi = ||\phi|| \cdot (\alpha \cdot sgn(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \phi_{severity} \tag{2.13}$$

- C_3 Random step change:

$$\Delta\phi = \mathcal{N}(0,1) \cdot \phi_{severity} \tag{2.14}$$

- C_4 Chaotic change:

$$\phi(t+1) = A \cdot \phi(t) \cdot (1 - \frac{\phi(t)}{||\phi||}) \tag{2.15}$$

- C_5 Recurrent change:

$$\phi(t+1) = \phi_{min} + ||\phi|| \cdot \frac{\big(sin(\frac{2\pi t}{P} + \varphi) + 1\big)}{2} \tag{2.16}$$

- C_6 Recurrent change step with noise:

$$\begin{aligned}\phi(t+1) = \phi_{min} + ||\phi|| \cdot \frac{\big(sin(\frac{2\pi t}{P} + \varphi) + 1\big)}{2} \\ +\mathcal{N}(0,1) \cdot \phi_{noisyseverity}\end{aligned} \tag{2.17}$$

where $||\phi||$ is the range of ϕ, $\phi_{severity} \in (0,1)$ is the change severity of ϕ, ϕ_{min} is the minimum value of ϕ, $\phi_{noisyseverity} \in (0,1)$ is the noisy severity in the recurrent changes with noise. The parameters $\alpha \in (0,1)$ and $\alpha_{max} \in (0,1)$ are the constant values in C_1 small step change and C_2 large step change respectively. A logistics function is used in C_4 chaotic change, where A is a positive constant between $(1.0, 4.0)$. If ϕ is a vector, the initial values in ϕ will be different within $||\phi||$ in C_4 chaotic change. P is the period in the C_5 recurrent step change and the C_6 recurrent step change with noise, φ is the initial phase, r is a random number between -1 and 1. The function $sgn(x)$ returns 1 when x is greater than 0, returns -1 when x is less than 0, otherwise returns 0. Finally, $\mathcal{N}(0,1)$ returns a normal distributed random number.

There are two instances in the GBDB benchmark: rotation DBG and composition DBG. In the rotation DBG, the fitness

Table 2.1: Parameter settings of the GBDB benchmark problem

For all test functions	
Dimension:	$n \in [5, 15]$ or $\mathbf{x} \in [-5, 5]^n$
Change Frequency:	$\tau = 2 \cdot 10^3 \cdot n \cdot \text{FES}$
Number of Change:	$T = 60$
Step of severity:	$\alpha = 0.040$
Maximum value of step of severity:	$\alpha_{max} = 0.1$
Period:	$p = 12$
Severity of recurrent with noise:	$\phi_{noisyseverity} = 0.8$
Chaotic constant:	$A = 3.67$
Height range:	$h \in [10, 100]$
Height severity:	$\phi_{h_{severity}} = 5.0$
For Composition DBG functions	
Number of basic functions:	$m = 10$
Coverage range factor:	$\sigma_i = 1.0, \forall i = 1, 2, \ldots, n$
Constant factor:	$C = 2000$
For Rotation DBG functions	
Number of basic function:	$m = 10$
Width Range:	$w \in \{1, 10\}$
Width Severity:	$\phi_{w_{severity}} = 0.5$

landscape consists of the multiple peaks that are controlled by tuning the system control parameters. The heights, the widths and the positions of each peak are changed in the six change types. If the dynamic problem is $f(x,\phi\,,t)$, the set of system control parameters is $\phi = (\mathbf{H}, \mathbf{W}, \mathbf{X})$, where $\mathbf{H}$,$\mathbf{W}$ and $\mathbf{X}$ are the peak heights, the peak widths and the peak positions respectively. Formally, the function $f(x,\phi\,,t)$ is

$$f(x,\phi\,,t) = \min\Bigg\{\mathbf{H}_i(t) + \mathbf{W}_i(t)\left(\exp\left(\sqrt{\sum_{j=1}^{n}\frac{(x_j - \mathbf{X}_j^i(t))^2}{n}}\right) - 1\right)\Bigg\}_{i=1}^{m} \tag{2.18}$$

where m is the number of the peaks, n is the problem dimension. The height and the width of the peaks are changed by:

$$\mathbf{H}(t+1) = \text{DynamicChanges}\left(\mathbf{H}(t)\right) \tag{2.19}$$

$$\mathbf{W}(t+1) = \text{DynamicChanges}\left(\mathbf{W}(t)\right) \tag{2.20}$$

where $\phi_{h_{severity}}$ and $\phi_{w_{severity}}$ are the change severities of the height and the width respectively. The ranges of the height and the width are denoted by $||\phi_h||$ and $||\phi_w|$ respectively.

Finally, instead of shifting the peak locations, a rotation matrix is used to change the peak locations in GDBG. This rotation matrix $R_{ij}(\theta)$ is obtained by rotating the projection of $\mathbf{x}$ in the plane $i-j$ by an angle θ from the i-th axis to the j-th axis. The peak position $\mathbf{X}$ is changed by the following steps:

1. Randomly select l dimensions from the n-dimensions and compose a vector $r = [r_1, r_2, \ldots, r_l]$, where l is an even number.

2. For each pair of dimensions $r[i]$ and the dimension $r[i+1]$, construct a rotation matrix $R_{r[i],r[i+1]}\left(\theta(t)\right)$, where $\theta(t) = \text{DynamicChanges}(\theta(t-1))$,

3. A transformation matrix $A(t)$ is obtained by

$$A(t) = R_{r[1],r[2]}(\theta(t)) \cdot R_{r[3],r[4]}(\theta(t)) \cdot \cdots R_{r[l-1],r[l]}(\theta(t)), \theta(t) \in (0, 2\pi) \quad (2.21)$$

4. $\mathbf{X}(t+1) = \mathbf{X}(t) \cdot A(t)$

where the change severity of θ and the range of θ are denoted by $\phi_{\theta_{severity}}$ and ϕ_θ respectively. The range of θ is between 0 and 2π.

Another instance of GDBG benchmark is the composition DBG. It constructs more challenging benchmark functions with the randomly located optima. By shifting, rotating and composing the optima of the standard functions, the functions that possess many desirable properties can be obtained. Formally, the composition DBG can be described as:

$$F(x,\phi\,,t) = \sum_{i=1}^{m} \left\{ w_i^{'} \cdot \left(f_i^{'} \left(\frac{(x - \mathbf{O}_i(t) + \mathbf{O}_{iold}) \cdot \mathbf{M}_i}{\lambda_i} \right) + \mathbf{H}_i(t) \right) \right\} \quad (2.22)$$

where $\phi = (\mathbf{O}, \mathbf{M}, \mathbf{H})$ is the system control parameter, $F(x)$ is the composition function, $f_i(\mathbf{x})$ is the i-th basic function[2] used to construct the composition function, m is the number of the basic functions, $\mathbf{M}_i$ is the orthogonal rotation matrix for each $f_i(\mathbf{x})$, $\mathbf{O}_i$ and $\mathbf{O}_{iold}$ are the shifted optimum and the old optimum respectively for each $f_i(\mathbf{x})$. The weight w_i for each $f_i(\mathbf{x})$ is defined as follows:

$$w_i = \exp\left(-\sqrt{\frac{\sum_{k=1}^{n} (x_k - o_i^k + o_{iold}^k)^2}{2n\sigma^2}}\right) \quad (2.23)$$

[2]For details of basic function used in the real composition DBG, please refer to [129, 128].

$$w_i = \begin{cases} w_i & \text{if } w_i = \max(w_i) \\ w_i \cdot (1 - \max(w_i)) & \text{if } w_i \neq \max(w_i) \end{cases} \tag{2.24}$$

$$w_i^{'} = \frac{w_i}{\sum_{i=1}^{m} w_i} \tag{2.25}$$

where σ_i is the coverage range factor of $f_i(\mathbf{x})$, λ_i is the stretch factor for each $f_i(\mathbf{x})$ which is defined as;

$$\lambda_i = \sigma_i \cdot \frac{X_{max} - X_{min}}{x_{max}^i - x_{min}^i} \tag{2.26}$$

where $[X_{min}, X_{max}]^n$ is the search range of $F(\mathbf{x})$ and $[x_{min}^i, x_{max}^i]^n$ is the search range of $f_i(\mathbf{x})$. In addition, the variable $f_i^{'}(\mathbf{x})$ is defined as

$$f_i^{'}(\mathbf{x}) = \frac{C \cdot f_i(\mathbf{x})}{|f_{max}^i|} \tag{2.27}$$

where C is a predefined constant. f_{max}^i is the estimated maximum value of $f_i(\mathbf{x})$ and can be calculated as:

$$f_{max}^i = f_i(x_{max} \cdot M_i) \tag{2.28}$$

In the composition DBG, $\mathbf{M}$ is randomly initialized and remains unchanged during the course of optimization. The dynamics of the system control parameters $\mathbf{H}$ and $\mathbf{O}$ can be described by:

$$\mathbf{H}(t+1) = \text{DynamicChanges}(\mathbf{H}(t)) \tag{2.29}$$

$$\mathbf{O}(t+1) = \text{DynamicChanges}(\mathbf{O}(t)) \tag{2.30}$$

Five basic functions that are used in the real composition DBG are outlined in Table 6.1, and the test function generated by GDBG is listed in Table 2.3.

Table 2.2: The basic functions in the composition DBG and the range of the search space

Name	Function	Range
Sphere	$f_{\text{Sphere}}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
Rastrigin	$f_{\text{Rastrigin}}(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5, 5]^n$
Weierstrass	$f_{\text{Weierstrass}}(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - n \sum_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$, $a = 0.5, b = 3, k_{max} = 20$	$[-0.5, 0.5]^n$
Griewank	$f_{\text{Griewank}}(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-100, 100]^n$
Ackley	$f_{\text{Ackley}}(\mathbf{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$

Table 2.3: The test functions generated by GDBG

Name	Number of Peaks / Basic Functions	Basic functions used
F_1 Rotation peak	10	-
F_2 Composition of Sphere's function	10	$\forall i = 1, \ldots, 10 : f_i = f_{\text{Sphere}}$
F_3 Composition of Rastrigin's function	10	$\forall i = 1, \ldots, 10 : f_i = f_{\text{Rastrigin}}$
F_4 Composition of Griewank's function	10	$\forall i = 1, \ldots, 10 : f_i = f_{\text{Griewank}}$
F_5 Composition of Ackley's function	10	$\forall i = 1, \ldots, 10 : f_i = f_{\text{Ackley}}$
F_6 Hybrid Composition function	10	$f_{1,2} = f_{\text{Sphere}}$, $f_{3,4} = f_{\text{Ackley}}$, $f_{5,6} = f_{\text{Griewank}}$, $f_{7,8} = f_{\text{Rastrigin}}$, $f_{9,10} = f_{\text{Weierstrass}}$

2.3 Discussion

Although the descriptions for the continuous black box optimization in Section 2.1 and 2.2 are short, we intend to give a general high-level comprehensive overview rather than details by details analysis on the topics. The basic properties of optimization problems in Section 2.1.3 and 2.2.2 contain the most commonly used ones in the literature. Interested readers can refer to the much more comprehensive works in [141, 191, 153, 237].

In black-box optimization testbeds, we have discussed a few popular testbeds in the literature. In the next chapters when we evaluate optimization algorithms, our experimental results are all tested under the framework of COCO [98] and GDBG [129, 128]. Specifically, when we evaluate the sampling methods we proposed to improve the state-of-the-art algorithms, it is important to leverage a platform where classes of problems can be more accessible to the community. It is also necessary to study the characteristics of problems when these proposed methods are suitable for these state-of-the-art algorithms. These two benchmark testbeds can definitely achieve this objective.

The review in this chapter showed that not many of the assumptions above are backed up by evidence from real-world applications. Therefore, this leads to the open questions of whether these academic assumptions still hold in real-world optimization, whether it is a static environment described in Section 2.1 or in dynamic environments in Section 2.2. Even the assumptions are true, we are still unable to tell whether these assumptions are representative in real-world applications and in what type of applications they will hold. There is very little research aiming at connecting the gap between these artificial benchmark problems and the real world optimization problems. There are certain gaps between current numerical optimization and real-world applications. In future research, we need to further inves-

tigate so as to close these gaps and to bring the practical use of the proposed methods to the realistic scenarios.

□ **End of chapter.**

Chapter 3

Evolutionary Algorithms

If I were again beginning my studies, I would follow the advice of Plato and start with mathematics.

Galileo Galilei

3.1 Overview of Evolutionary Computation

3.1.1 History

Evolutionary Computation [30, 78, 77, 31, 65, 85, 125] is a research field where the nature inspired or evolution inspired computational methods are used for solving the real-world problems. It has been growing rapidly since the first introduction of evolutionary algorithms [2]. The very first optimization algorithms that were closely related to evolution in the literature were proposed in the works [34, 33, 80, 49, 50, 182, 183] during 1950s and the early 1960s. Later, Hans-Paul Schwefel simulated different versions of two membered (1+1)-evolution strategies (ESs) in which one parent generates one offspring. The simulation was conducted in the first digital computer Zuse Z23 of Technical University of Berlin [198]. Ingo Rechenberg later theoretically analyzed these strategies and proposed the first μ multi-membered strategies $(\mu + 1)$ in his doctoral thesis [71].

Another sub-field of Evolutionary Computation, Evolutionary Programming was proposed by Lawrence J. Fogel in his work [79] during the 1960s. Fogel proposed to evolve the population of finite state machines (FSMs) to solve problems of prediction and control, defined as a set of sequences from a finite alphabet. The process was named Evolutionary Programming in contrast to Linear Programming [62], Dynamic Programming [39] and Quadratic Programming [151].

Schumer and Steiglitz [196] proposed the Adaptive Step-Size Random Search (ASSRS), an adaptive version of Fixed Step-Size Random Search [182, 183, 148]. They proved that Optimal Step-Size Random Search (OSSRS) on unimodal functions and found out that:

1. the optimal step size is proportional to the distance to the optimum, similar results were published later for (1+1)-ES [24];

2. there is an optimal probability of improvements for ASSRS, similar methods one-fifth success rule was proposed in [71];

3. the average number of function evaluations to reach a target accuracy is linear in the problem dimension.

Since the distance to the optimum is usually unknown, only an approximation of the optimal step size can be estimated.

In 1975, John H. Holland proposed Genetics Algorithms (GA) in his book [110]. His work provides a simplified but rich theory of how adaptation and evolution work and provides a working algorithmic framework to simulate the evolution for solving real-world problems. Holland also proposed the so-called Schema Theorem to provide the theoretical evidence of GA convergence.

3.1.2 Modern Approaches

In the Evolutionary Computation community, there are many modern approaches that are inspired by the evolution and the nature for solving optimization problems. The following subsection summarizes the most widely used approaches and give their brief comparison analysis.

Evolution Strategies

Evolution strategies were used to solve the continuous optimization [200, 71] since its introduction. There were works that ES was used to optimize mixed integer optimization [32, 130, 96, 131] but most theoretical and empirical experimental works are on continuous optimization. In this thesis, we focus on different variants in the family of evolution strategies. We give a detailed description in Section 3.2 and Section 3.3.1.

Genetic Algorithms

Genetic Algorithms (GAs) are the most common approach used for the initial studies of evolution-inspired optimization. The first GA was proposed by Holland in his work [110] and was called "Simple Genetic Algorithm" (SGA). The candidate solutions are binary coded parent solutions competing with offspring solutions, generated after multi-point crossovers and bit-flip mutations of parents. Another representative genetic algorithms in the literature used the real-value representation of decision variables and enriched the set of variation operators [228, 66]. The key to designing a successful GA is that the algorithm designers pick the right choice of the representation of candidate solutions for a given problem, such that the variation operators favour successful exploration and exploitation of the search space. The Cooperative Coevolutionary Algorithms (CCEAs)

[173], described in Section 3.4 later, can be categorized into genetic algorithms as they were first proposed in [172]. We will give more details of CCEAs in Section 3.4.

Estimation of Distribution Algorithms

In Estimation of Distribution Algorithms (EDA) [126], unlike other approaches where the distribution of candidate solutions is defined implicitly by variation operators, the sampling of new solutions in EDA is explicitly defined by a chosen probabilistic model, e.g. multivariate normal distributions. Examples of EDA in the literature include Univariate Marginal Distribution Algorithm (UMDA) [147], Estimation of Multivariate Normal Algorithm (EMNA) [126] and Cross-Entropy Method (CEM) [192].

Ant Colony Optimization

Ant Colony Optimization (ACO) [69, 70] was proposed by Marco Dorigo in his doctoral thesis [68]. At the beginning, ACO was an optimization algorithm which imitated the behaviour of ants to solve usual combinatorial optimization problems. The artificial ants seek the path between their colonies and a source of the food and lay down pheromone trails such that other ants will more likely to follow these trails. This may reinforce attractiveness of the shortest path to the food. Trail evaporation can occur in reducing pheromone values of all trails over time and preserving some diversity in the optimal path seeking. Later, there is progress to generalize ACO and use it for continuous optimization [204].

Genetic Programming

Genetic Programming (GP) was first proposed by the work [61] and later popularized by John R. Koza [125, 124]. GP is an

evolution-inspired technique which imitates artificial evolution of computer programs that perform a user-predefined task. GP can be considered as a GA when we used tree structure to represent individuals. Unlike other approaches, the number of variables or components representing a candidate solution is usually dynamic during the search. This often favours extensive search in the space of possible topologies. As a result, the final candidate solutions are too complicated even for a relatively simple problem. This leads to the development of multi-objective version of GP [125] to control the "bloat issue".

Differential Evolution

Differential Evolution (DE) [209] was proposed by Rainer Storn and Kenneth Price [208]. It has become popular in the recent few years because of its simplicity and efficiency. DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple parameters, and then keeping whichever candidate solution has the best fitness. If the new candidate solution has an improvement, it is accepted and becomes part of the population, otherwise the candidate solution is discarded.

The DE typically has only a few parameters and several mutation strategies. Selecting the DE parameters that yield a good performance has therefore been the subject of much research [207, 179]. In the literature, there are also mathematical convergence analyses regarding parameter selections [166, 165] as well as different variants [179, 136, 180, 51].

Particle Swarm Optimization

Particle Swarm Optimization (PSO) [120] optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search space

according to simple mathematical formulas over the particle's position and velocity. Each particle's movement is influenced by its local best known positions but, is also guided toward the best known positions in the search-space, which are updated when the better positions are found by other particles. This is expected to move the swarm towards the best solutions.

3.2 Evolution Stratgies

In this section, we describe the key variants of evolution strategies in the literature. We first outline the (1+1)-ES that is the historically first version of an evolution strategy. We then describe a multi-membered (μ,λ)-ES in which μ number of parent generates λ number of offsprings. We also review the step size adaptation methods that are commonly used in the literature.

3.2.1 Single Parent Elitist (1+1)-ES

The (1+1)-ES was developed in the work [198, 71] and was inspired by biological evolution. In the first version, a very simple evolution loop without any algorithm parameters was used. This algorithm generates a single offspring $\mathbf{x}'$ by

$$\mathbf{x}' = \mathbf{x} + \sigma \cdot \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{I}) \tag{3.1}$$

If the offspring performs better than its parent (in terms of fitness), it becomes the new parent. Otherwise, the parent remains. The standard deviation σ of the normal distribution was a fixed scalar value. The work [71] has shown that the step size σ can be chosen such that the (1+1)-ES can be made successful when a candidate solution is sampled. This leads to the development of step size adaptation such that the step size σ can be "adapted" so as to improve the local performance on two objective functions, namely the corridor function and the sphere

function. A theoretical study was conducted to show that the best convergence rate is achieved when one-fifth of all mutations are successful. This leads to the classical step size adaptation the so-called "one-fifth success rule". If about one-fifth of all mutations are successful, the step size is optimal and no adaptation is required. If the success rate falls below one-fifth, the step size needs to be reduced. If it grows above one-fifth, the step size needs to be increased. To obtain the new step size, the current step size σ is either multiplied or divided by a constant c. The recommendation on the value of c is 0.817 [199]. The step size adaptation is applied in each k iterations of the algorithm, and the success rate p_{succ} is measured over a sliding window of the last $10 \cdot k$ mutations.

Algorithm 1: Pseudo Code of the (1+1)-ES with one-fifth success rule.

1 **Given:** $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{X}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma_1 > 0$
2 Initialise $\mathbf{X}_1$, σ_1, $k = 1$
3 **repeat**
4 $\quad \mathbf{X}_k^{'} \leftarrow \mathbf{X}_k + \sigma_k \boldsymbol{\mathcal{N}}(\mathbf{0}, \boldsymbol{I})$ `/* sample an offspring */`
5 $\quad$ **if** $f(\mathbf{X}_k^{'}) \leq f(\mathbf{X}_k)$ **then**
6 $\quad\quad \mathbf{X}_{k+1} \leftarrow \mathbf{X}_k^{'}$ `/* replace the parent */`
7 $\quad\quad \sigma_{k+1} \leftarrow 1.5\sigma_k$ `/* increase step size */`
8 $\quad$ **else**
9 $\quad\quad \mathbf{X}_{k+1} \leftarrow \mathbf{X}_k$ `/* keep the parent */`
10 $\quad\quad \sigma_{k+1} \leftarrow 1.5^{-1/4}\sigma_k$ `/* decrease step size */`
11 $\quad k \leftarrow k + 1$ `/* iteration counter */`
12 **until** *termination condition is met*

Algorithm 1 shows the pseudo-code of a (1+1)-ES with the one-fifth rule, where at iteration k the parent $\mathbf{X}_k$ generates an offspring $\mathbf{X}_k^{'}$ (line 4) by Gaussian mutation, defined by the step-size σ_k and an identity covariance matrix $\mathcal{C} = \boldsymbol{I}$. By setting the covariance matrix to the identity, the variations of all variables are independent of each other, and it implies that they are uncorrelated.

The offspring replaces its parent (lines 6) if it has fitness at least as good as its parent's. The update of the mutation step-size (line 7) is increased by a factor of 1.5. If otherwise, the offspring is worse than the parent, the current parent will become the parent in the next iteration $k+1$ (line 9). The mutation step size is decreased by a factor of $1.5^{-1/4}$. According to [18], the choices between increasing and decreasing the factors are basically implementing the idea of the one-fifth success rule.

3.2.2 Population-based Non-elitist (μ,λ)-ES

The (μ,λ)-ES is the evolution strategy that uses the mutative σ-self adaptation for step size adaptation scheme. It is basically a non-elitist, multi-membered strategy that uses *comma selection*. That means that the parents never survive to the next iteration and μ best out of λ offspring are chosen to become parents in the next generation. The self-adaptation for the mutative step sizes is common in the family of ES. There have been a lot of theoretical works to study the performance of ES with the use of mutative σ-self adaptation [41, 42, 92, 139, 140]. The underlying idea of mutative step size adaptation is based on the assumption that individuals with good settings of strategy parameters will generate good offspring, such that the good strategy parameters survive the selection. Recombination of candidate solutions and strategy parameters is performed through global intermediate recombination, i.e. by averaging all of the μ parents.

Algorithm 2 shows the pseudo code of a (μ,λ)-ES with mutative σ-self adaptation. The algorithm starts by initializing μ individuals in the parental population $\mathcal{P}_k$ (line 2), where $\mathcal{P}_k$ denotes the parental population at iteration k, and $(\mathbf{X}_k^i, f(\mathbf{X}_k^i), \boldsymbol{\sigma}_k^i)$ is the ith offspring at iteration k. Note that each individual maintains its own mutative step size $\boldsymbol{\sigma}_k^i \in \mathbb{R}^n$ in the (μ,λ)-ES. First, the candidate solutions and the mutative step size are

recombined separately (lines 4 and 5) to form an intermediate parent. Each mutative step size is updated by σ-mutative self adaptation (line 8). The local and global learning rates τ and τ' are set to $\frac{1}{\sqrt{2\sqrt{n}}}$ and $\frac{1}{2\sqrt{n}}$ respectively [200, 27], where n is the problem dimension. Offspring are generated by using the newly updated mutative step size (line 9). After generating λ offspring, a comma selection takes place to select the best μ out of the λ offsprings and they become the parental population in the next iteration $\mathcal{P}_{k+1}$ (line 11).

Algorithm 2: Pseudo Code of the (μ,λ)-ES with mutative σ-self adaptation.

1 **Given:** $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{X}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma_1 > 0$
2 Initialise $\mathcal{P}_1 := \{(\mathbf{X}_k^1, f(\mathbf{X}_k^1), \boldsymbol{\sigma}_k^1), \ldots, (\mathbf{X}_k^\mu, f(\mathbf{X}_k^\mu), \boldsymbol{\sigma}_k^\mu)\}$, $k = 1$
3 **repeat**
4 $\quad \mathbf{X}_k^p \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} \mathbf{X}_k^i$ `/* recombination to form a parent */`
5 $\quad \boldsymbol{\sigma}_k^p \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} \boldsymbol{\sigma}_k^i$ `/* recombination to form a global step size */`
6 $\quad i \leftarrow 1$ `/* offspring counter */`
7 $\quad$ **while** $i \leq \lambda$ **do**
8 $\quad\quad \boldsymbol{\sigma}_k^i = \boldsymbol{\sigma}_k^p \exp\{\tau \mathcal{N}_i(0,1) + \tau' \boldsymbol{\mathcal{N}}_i(\mathbf{0}, \mathbf{I})\}$ `/* adapt step size */`
9 $\quad\quad \mathbf{X}_k^i \leftarrow \mathbf{X}_k^p + \boldsymbol{\sigma}_k^i \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ `/* sample an offspring */`
10 $\quad\quad i \leftarrow i + 1$
11 $\quad$ Select the best μ out of $\{(\mathbf{X}_k^i, f(\mathbf{X}_k^i), \boldsymbol{\sigma}_k^i)\}$ for $1 \leq i \leq \lambda$ into $\mathcal{P}_{k+1}$
12 $\quad k \leftarrow k + 1$ `/* iteration counter */`
13 **until** *termination condition is met*

3.3 Covariance Matrix Adaptation Evolution Strategies (CMA-ES)

In recent years, there has been research work that have contributed to the state-of-the-art covariance matrix adaptation evolution strategies (CMA-ES) [103, 105, 106, 104, 102] used to solve many black-box optimization problems. CMA-ES usually

optimizes the real value objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ in the continuous domain. The CMA-ES iteratively evolves the populations of individuals by variations and selections. The quality of the individuals is determined by their fitness, assigned by evaluating the underlying objective functions. Selection chooses the best individuals to become the parents in the next generations. Variations are done by mutations and recombinations. Many works have demonstrated the success of CMA-ES in several domains including optimization, machine learning and the real world applications.

On ill-conditioned problems, covariance matrix adaptation can accelerate the rate of convergence of evolution strategies by orders of magnitude. For example, a successful covariance matrix adaptation can enable strategies to generate candidate solutions predominantly in the directions of narrow valleys. The CMA-ES is able to learn the appropriate covariance matrix from successful steps that the algorithm has taken. It exhibits invariance properties that make it suitable for solving non-separable optimization problems. The CMA-ES obtains the information about the successful search steps and uses the information to update the covariance matrix of the mutation distribution in a derandomised mechanism. The covariance matrix is updated such that variances in directions of the search space that have previously been successful are increased, and those in other directions are decreased. Even in a small population, the accumulation of information over a number of successful steps can reliably adapt the covariance matrix.

The CMA-ES is basically a stochastic search algorithm that samples its new candidate solutions from a multivariate normal distribution and adapts its mean and covariance matrix after each iteration. Many studies [121, 93] have shown it to have a superior performance over other optimization algorithms in continuous optimization problems. The CMA-ES had the best

performance in the benchmark problems in the CEC 2005 competition and the COCO benchmarks workshop. There has been a lot of work to further improve the performance of the standard CMA-ES. These include the restart CMA-ES [23], the local CMA-ES [22], the CMA-ES using active covariance matrix [116], the (1+1) CMA-ES using "incremental Cholesky update" [114], the functionally specialized CMA-ES [1], the L-CMA-ES [122], the sep-CMA-ES [189], and the BI-population CMA-ES [94]. In the following subsections, we outline the three variants of CMA-ES studied in this thesis.

3.3.1 The standard (μ,λ) CMA-ES

In each iteration k of the standard (μ,λ)-CMA-ES [106, 104], λ number of candidate solutions are generated by sampling a multi-variate normal distribution $\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C})$ with mean $\mathbf{0}$ and a $n \times n$ covariance matrix $\mathbf{C}$. The μ best solutions are selected to update the distribution parameters for the next iteration step k+1. The standard CMA-ES employs the concept of cumulative step adaptation (CSA). There are two evolution paths $\mathbf{p}^{\sigma}$ and $\mathbf{p}^{c}$, both of them are two n-dimensional vectors used to accumulate information about the recent steps of the strategy. The learning of the accumulating information is controlled by three independent learning rates c_{σ}, c_1 and c_c that change the global step size σ and the covariance matrix $\mathbf{C}$.

Algorithm 3 shows the pseudo code of the standard (μ,λ) CMA-ES. The algorithms starts by initializing a few parameters including the distribution mean $\mathbf{m}_1$, the step size $\boldsymbol{\sigma}_k$, two evolution paths $\mathbf{p}_1^{\sigma}$ and $\mathbf{p}_1^{c}$, and the covariance matrix $\mathbf{C}_1$. From line 4 to line 7, the CMA-ES samples its offspring by the equation $\mathbf{m}_k + \sigma_k \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}_k)$. The notation $\sigma_k \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}_k)$ denotes the random vector in n-dimensional drawn from a multivariate normal distribution and is equivalent to $\boldsymbol{\mathcal{N}}(\mathbf{0}, (\sigma_k)^2 \mathbf{C}_k)$. The equation

in line 5 can basically be rewritten into $\mathbf{x}_k^i \leftarrow \boldsymbol{\mathcal{N}}(\mathbf{m}_k, (\sigma_k)^2\mathbf{C}_k)$. The new mean distribution $\mathbf{m}_{k+1}$ is updated in line 8; it is essentially the weighted sum of the best μ out of the λ candidate solutions. The symbol $\mathbf{x}_{i:\lambda}$ represents the i-th best of the candidate solutions $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ and therefore we have $f(\mathbf{x}_{1:\lambda}) \leq \cdots \leq f(\mathbf{x}_{\lambda:\lambda})$.

The step-size σ_k is updated using cumulative step-size adaptation (CSA), sometimes also referred to path length control. The evolution path $\mathbf{p}_k^\sigma$, which is also called the search path, is updated first in line 9. The first term $(1-c_\sigma)\mathbf{p}_k^\sigma$ on the right hand side of the equation represents the discounted evolution path $\mathbf{p}_k^\sigma$. The constant c_σ controls how much information is accumulated in the backward time horizon. The second term in the equation represents the current update based on the displacement of the new mean distribution $\mathbf{m}_{k+1}$. The term $\sqrt{c_\sigma(2-c_\sigma)\mu_w}$ is the complement for discounted variance. The constant μ_w is the variance effective selection mass for $1 \leq \mu_w \leq \mu$. The ${\mathbf{C}_k}^{-\frac{1}{2}}$ denotes the unique symmetric square root of the inverse of $\mathbf{C}_k$, such that the equation ${\mathbf{C}_k}^{-\frac{1}{2}} = \sqrt{\mathbf{C}_k}^{-1} = \sqrt{{\mathbf{C}_k}^{-1}}$ holds. Alternatively, if $\mathbf{BDB}^T = \mathbf{C}$ is an eigen-decomposition into an orthogonal matrix $\mathbf{B}$ with a diagonal matrix $\mathbf{D}$, we have ${\mathbf{C}_k}^{-\frac{1}{2}} = \mathbf{B}\mathbf{D}^{-\frac{1}{2}}\mathbf{B}^T$.

The new evolution path for covariance matrix $\mathbf{p}_{k+1}^c$ (line 11) is as the weight sum of the current $\mathbf{p}_k^c$ and the displacement of the new mean distribution $\mathbf{m}_{k+1}$. Similarly, the constant c_c controls how much information from the previous steps is needed. The indicator function in line 10 evaluates whether the length of evolution path exceeds a threshold value or not. If it does, the evolution path for the covariance matrix will be updated as the new mean distribution $\mathbf{m}_{k+1}$. The purpose is to stall the update of $\mathbf{p}^c$ when σ increases rapidly.

The covariance matrix $\mathbf{C}$'s update consists of two parts (line 13): a rank one update and a rank-μ update. The term $c_\mu +$

Table 3.1: Default parameter values of the (μ,λ)-CMA-ES

$$
\begin{aligned}
&\lambda = 4 + \lfloor 3 \ln(n) \rfloor \\
&\mu = \lfloor \lambda/2 \rfloor \\
&w_{i=1,\ldots,\mu} = \frac{\ln(\mu+\frac{1}{2}) - \ln(i)}{\sum_{j=1}^{\mu} \left(\ln(\mu+\frac{1}{2}) - \ln(j)\right)} \\
&\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \\
&c_\sigma = \frac{\mu_w+2}{n+\mu_w+3} \\
&d_\sigma = 1 + 2 \cdot \max\left(0, \sqrt{\frac{\mu_w-1}{n+1}} - 1\right) + c_\sigma \\
&c_c = \frac{4}{n+4} \\
&c_1 = \frac{\min(2,\lambda/3)}{(n+1.3)^2+\mu_w} \\
&c_\mu = \frac{2(\mu_w - 2 + \frac{1}{\mu_w})}{(n+2)^2+\mu_w}
\end{aligned}
$$

$c_1 \mathbf{p}_{k+1}^c {\mathbf{p}_{k+1}^c}^T +$ in line 13 is basically the rank-one update that includes the use of the evolution path $\mathbf{p}^c$. The constant c_1 is the learning rate for the rank-one update of the covariance matrix. The term $c_\mu \mathbf{C}^\mu$ refers to the rank-μ-update that computes a covariance matrix $\mathbf{C}^\mu$ as a weighted sum of the covariances of successful steps of the μ best individuals. The constant c_μ is the learning rate for the rank-μ update and must not exceed $1 - c_1$. The update of $\mathbf{C}$ itself is a replacement of the previously accumulated information by a new one with corresponding weights of importance.

The step size σ in line 14 is increased if and only if $||\mathbf{p}_{k+1}^\sigma||$ is larger than the expectation of $\mathrm{E}||\boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})||$, and is decreased if it is smaller. The constant d_σ is the damping parameter and is usually close to one. Table 3.1 shows the default parameters used in the standard (μ,λ)-CMA-ES.

3.3.2 sep-CMA-ES

In the standard CMA-ES, the full learning task scales roughly with n^2 and can dominate most of the search cost. This is one

Algorithm 3: Pseudo Code of the standard (μ,λ) CMA-ES

1 **Given:** $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma_1 > 0$,$\mathbf{m}_k \in \mathbb{R}^n$,$\mathbf{p}_k^\sigma, \mathbf{p}_k^c \in \mathbb{R}^n$,$\mathbf{C}_k \in \mathbb{R}^{n\times n}$
2 Initialise $\mathbf{m}_1$,σ_1, $\mathbf{p}_1^\sigma = \mathbf{0}$, $\mathbf{p}_1^c = \mathbf{0}$, $\mathbf{C}_1 = \mathbf{I}$,$k = 1$
3 **repeat**
4 **while** $i \leq \lambda$ **do**
5 $\mathbf{x}_k^i \leftarrow \mathbf{m}_k + \sigma_k \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{C}_k)$
6 $f_i = f(\mathbf{x}_k^i)$
7 $i \leftarrow i + 1$
8 $\mathbf{m}_{k+1} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$
9 $\mathbf{p}_{k+1}^\sigma = (1 - c_\sigma)\mathbf{p}_k^\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_w}\mathbf{C}_k{}^{-\frac{1}{2}} \frac{\mathbf{m}_{k+1} - \mathbf{m}_k}{\sigma_k}$
10 $h_\sigma = 1_{\left\{||\mathbf{p}_{k+1}^\sigma|| < \sqrt{1-(1-c_\sigma)^{2(g+1)}}(1.4 + \frac{2}{n+1})\mathrm{E}||\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I})||\right\}}$
11 $\mathbf{p}_{k+1}^c = (1 - c_c)\mathbf{p}_k^c + h_\sigma \sqrt{c_c(2 - c_c)\mu_w} \frac{\mathbf{m}_{k+1} - \mathbf{m}_k}{\sigma_k}$
12 $\mathbf{C}^\mu = \sum_{i=1}^{\mu} w_i \frac{\mathbf{x}_{i:\lambda} - \mathbf{m}_k}{\sigma_k} \times \frac{(\mathbf{x}_{i:\lambda} - \mathbf{m}_k)^T}{\sigma_k}$
13 $\mathbf{C}_{k+1} = (1 - c_1 - c_\mu)\mathbf{C}_k + c_1 \mathbf{p}_{k+1}^c {\mathbf{p}_{k+1}^c}^T + c_\mu \mathbf{C}^\mu$
14 $\sigma_{k+1} = \sigma_k \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\mathbf{p}_{k+1}^\sigma||}{\mathrm{E}||\boldsymbol{\mathcal{N}}(\mathbf{0},\mathbf{I})||} - 1\right)\right)$
15 $k \leftarrow k + 1$
16 **until** *termination condition is met*

of the major limitations of the standard CMA-ES because of the high degree of freedom $\frac{n^2+n}{2}$ in the covariance matrix. One of the solutions is to reduce the degree of freedom from $\frac{n^2+n}{2}$ to n where only the diagonal of the covariance matrix is adapted. The resulting algorithm is called "sep-CMA-ES" [189]. There are two simple changes undertaken in sep-CMA-ES. First, the covariance matrix $\mathbf{C}$ is constrained to be diagonal. Second, the learning rate c_μ is increased. This means that the mutation distribution is sampled independently in the given coordinate system using n individual variances. For sep-CMA-ES, the changes to the updated equations in Algorithm 3 and Table 3.1 are:

1. The original update of the covariance matrix in line 13 is changed to

$$c_{k+1}^{jj} = (1 - c_\mu)c_k^{jj} + \frac{c_\mu}{\mu_w}(\mathbf{p}_{k+1}^c)_j^2 + c_\mu(1 - \frac{1}{\mu_w})(\mathbf{C}^\mu)^{jj} \quad (3.2)$$

for $j = 1, \ldots n$ where c_{jj} and $(\mathbf{C}^{\mu})^{jj}$ denotes the jth diagonal elements of the covariance matrix $\mathbf{C}$ and the matrix $\mathbf{C}^{\mu}$ respectively. The terms $\left(\mathbf{p}_{k+1}^{c}\right)_j$ represents the jth element of the evolution path $\mathbf{p}^{c}$.

2. The learning rate for rank-μ update is updated by:

$$c_{\mu}^{\text{sep}-\text{CMA}-\text{ES}} = \frac{n+2}{3} \cdot c_{\mu} \tag{3.3}$$

3.3.3 (1+1) CMA-ES

The (1+1)-CMA-ES is a new variant that has been recently proposed [115] as an extension of the (1+1)-ES with the one-fifth success rule [184]. It differs from the standard CMA-ES variant in that: (1) it is an elitist algorithm, and (2) the step size and the covariance matrix associated to the search distribution are adapted. The experimental results in [115] show that the (1+1)-CMA-ES is about 1.5 times faster than the standard CMA-ES on unimodal functions.

We follow the principles introduced in [115] and the (1+1)-CMA-ES is summarized in Algorithm 4. A candidate solution $\mathbf{x'}_k$ (line 6) is sampled by perturbing the current solution $\mathbf{x}_k$ by adding a normal distributed vector with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{C}_k$, scaled by the mutation step-size σ_k. This candidate solution is accepted only if $f(\mathbf{x'}_k) < f(\mathbf{x}_k)$. The mutation step-size (line 8) is adapted using the average success rate p^{succ} such that it is increased if the success rate is strictly larger than the target probability $p_{\text{target}}^{\text{succ}}$, and decreased if it is strictly smaller. If $f(\mathbf{x'}_k) < f(\mathbf{x}_k)$, the covariance matrix (line 11) is adapted by adding the matrix $\mathbf{p}_{k+1}{\mathbf{p}_{k+1}}^{T}$ a multiple of $\mathbf{C}_k$ the rank-one update where ${\mathbf{p}_{k+1}}^{T}$ is the transpose of $\mathbf{p}_{k+1}$. We will use the same default settings as in [115] for all strategy parameters. For completeness, the default values of parameters are

Table 3.2: Default parameter values of the (1+1)-CMA-ES

$p^{\text{succ}}_{\text{target}} = \frac{2}{11}, c_p = \frac{1}{12}$
$c_c = \frac{2}{N+2}, c_\mu = \frac{2}{N^2+6}$
$p^{\text{thresh}} = 0.44$

shown in the Table 3.2.

Algorithm 4: Pseudo Code of the (1+1)-CMA-ES

Given: $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma_1 > 0, p^{\text{succ}} \in \mathbb{R}$, $\mathbf{p}_1 \in \mathbb{R}, \mathbf{C}_1 \in \mathbb{R}^{n \times n}$
Initialise $\mathbf{x}_1$, σ_1, $p^{\text{succ}}_k = p^{\text{succ}}_{\text{target}}, k = 1$
repeat
  $\mathbf{A}_k = \texttt{chol(}\mathbf{C}_k\texttt{)}$ where $\texttt{chol(}\cdot\texttt{)}$ is the Cholesky decompositions such that $\mathbf{C} = \mathbf{A}\mathbf{A}^T$
  $\mathbf{z}_k \leftarrow \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}'_k \leftarrow \mathbf{x}_k + \sigma_k \mathbf{A}_k \mathbf{z}_k$
  $p^{\text{succ}}_{k+1} = (1 - c_p) p^{\text{succ}}_k + c_p 1_{\{f(\mathbf{x}'_k) \le f(\mathbf{x}_k)\}}$
  $\sigma_{k+1} = \sigma_k \exp\left(\frac{p^{\text{succ}}_{k+1} - p^{\text{succ}}_{\text{target}}}{n \cdot (1 - p^{\text{succ}}_{\text{target}})}\right)$
  if $f(\mathbf{x}'_k) \le f(\mathbf{x}_k)$ **then**
    $\mathbf{p}_{k+1} \leftarrow (1 - c_c)\mathbf{p}_k + 1_{\{p^{\text{succ}}_{k+1} < p^{\text{thresh}}\}} \sqrt{c_c(2 - c_c)} \mathbf{A}_k \mathbf{z}_k$
    $\mathbf{C}_{k+1} \leftarrow$
    $\left(1 - c_\mu + c_\mu 1_{\{p^{\text{succ}}_{k+1} > p^{\text{thresh}}\}} c_c(2 - c_c)\right) \mathbf{C}_k + c_\mu \, \mathbf{p}_{k+1} \, \mathbf{p}_{k+1}{}^{\mathrm{T}}$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}'_k$
  else
    $\mathbf{p}_{k+1} \leftarrow \mathbf{p}_k$
    $\mathbf{C}_{k+1} \leftarrow \mathbf{C}_k$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
  $k \leftarrow k + 1$
until *termination condition is met*

3.3.4 CMA-ES with restarts

The multi-modality of a black-box optimization problem can lead to the premature convergence of optimization algorithms. In order to increase the probability of finding the best optima,

two variants of CMA-ES with restart strategies are proposed. In the IPOP-CMA-ES [23], the CMA-ES is restarted by doubling the population size every time when the stopping criterion is met. In BIPOP-CMA-ES [95], there are two regimes. One regime is a CMA-ES with a large population and its population size is doubled in every restart. Another regimes is a CMA-ES with small population and the population size is randomly assigned at each restart. BIPOP-CMA-ES restarts the regimes sequentially, such that two regimes have almost the same budget of function evaluations.

3.4 Cooperative Coevolutionary Algorithm (CCEA)

3.4.1 Background

Coevolutionary algorithms are one of the popular evolutionary algorithms (EAs) and are fundamentally different from the traditional EAs. Fitness evaluations in the coevolutionary algorithms always depend on the outcomes of interactions between individuals. Traditionally, the coevolutionary algorithms can be classified into the competitive coevolution [190] and the cooperative coevolution [173]. In a competitive coevolutionary algorithm, the individuals compete against others. The increase in the fitness of an individual will decrease the fitness of another individual. In a cooperative coevolutionary algorithm (CCEA), the fitness demonstrates how well the individuals perform when they collaborate. Higher fitness is given to the individuals when they perform well. Lower fitness is given when they perform poorly. Intuitively, one may consider the CCEAs more superior to the traditional EAs because the CCEAs decompose the search space when they search the optima for optimization problems. Each population of a CCEA only requires a search of the projection of a n-dimensional problem and it is therefore natural

to assume that the CCEAs perform better. However, the work [222] reported that every CCEA population easily searches for components of candidate solutions that are robust to the other components. Most importantly, the combinations of these components are not always globally optimal. Therefore there are many works in the literature that aim towards understanding the behaviour of the CCEAs [171] and improving the performance of the CCEAs for static optimization [161].

The CCEA [173] differs from the traditional EAs in their ways of solving problems. In a CCEA, an individual only represents a single component of a candidate solution. The individuals that represent the same component form a CCEA population. An individual is evaluated by collaborating with the individuals from other populations. To reduce the noise for the fitness evaluations, multiple evaluations are required. An CCEA individual does not have one individual for its fitness evaluations, but multiple sets of individuals for multiple fitness evaluations. The final fitness of the individual can be the results of the maximum fitness, the minimum fitness or the average fitness over all these multiple evaluations. The effects of using different collaboration methods in a CCEA for static optimization are studied in [224, 223]. In the literature, there are many real-world applications that used the cooperative coevolutionary approaches. These applications include the optimization for the inventory control systems [72], learning the constructive neural networks [173], the multi-agent systems [57, 176], and the rule learning [174, 175].

If a CCEA is used in static optimization, the CCEA optimizes the functions differently. For instance we now optimize a two argument function $f(x, y)$. A common EA usually consists of one population. An EA individual is made up of the genes that represent two dimensions. For problems like these, a CCEA would usually have two populations. The genes of the CCEA

individuals represent one dimension. The individuals in the first and the second populations represent x and y arguments respectively. Both populations evolve separately. The only difference is in their fitness evaluations. A CCEA individual from a population chooses an individual from the other population and then they are evaluated by the objective function $f(x, y)$.

There are a few collaboration methods in the literature. One simple way for an individual to choose collaborators is to select the best-fit individuals from the previous rounds of evaluations. This method is called "best collaboration". Another method is called "random collaborations'. In the random collaboration, an individual randomly chooses the collaborators. The fitness of the selected individuals is never considered. Another common method is "complete collaboration". In the complete collaboration, an individual collaborates with all individuals from other populations. Moreover, an individual can use different collaboration methods to select its collaborators. For instance, an individual can use the best collaboration method to choose the best-fit collaborator and use the random collaboration to select four random collaborators [223].

In the literature, the work [222] reports that the CCEAs lose a lot of information by projecting the search space into the separated components. The fitness of the components is sensitive to the components with which they are chosen. As a result, the CCEAs are easily trapped and driven toward the sub-optimal solutions. One way to tackle it is to provide the CCEA individuals with more information about the best collaborators. Another method is to allow an individual to collaborate with a large number of individuals and to take the maximum fitness from these collaborations. The early works [56, 223] have shown that the performance is better when taking the maximum fitness of these evaluations, rather than taking the average fitness or the minimum fitness. The advantages of using multiple col-

laborators are graphically illustrated in the work [162]. The experimental study [159] has also demonstrated that the performance of a CCEA can be further improved if the collaboration methods are changing during the course of optimization. Other recent works [203, 160] focus on an archive methods to store the useful collaborators in every generation. The method produces an archive of individuals that provides a good assessment with the whole CCEA population. The size of the archive can be very small and result in a significant reduction in numbers of fitness evaluations. However, there are two difficulties in this archive-based collaboration method. Firstly, it is difficult to decide what and when the collaborators are chosen into the archive Secondly, it is difficult to determine the optimal sizes of the archive. A large-sized archive will certainly improve the accuracy of finding the best collaborators, but having too large an archive will cost too much in terms of numbers of function evaluations.

There are many empirical studies that focus on various aspects to better understand the dynamic of a CCEA for the static optimization. It includes the collaboration methods [223], the interaction frequency for the CCEA populations [170] and the updating timings in the sequential and parallel variants of a CCEA [169]. The most prominent study is the one by [171, 168]. In the study, the authors analyze the dynamic of a CCEA by investigating the trajectories of the best-of-generation individuals and by studying the best-response curves. It is found that the performance of a CCEA is greatly correlated to the best-response curves[1] that are fundamentally the basic properties of the underlying optimization problems. The settings of a CCEA, including the collaboration methods, the use of the elitist individuals and the interaction frequency for the CCEA populations,

[1]The term "best response curves" was originally used in a two-populations CCEA [171, 168]. The authors also mentioned that when more than two populations are used, the best-response curves become the best-response surfaces or even the best-response subspaces in the higher dimensions.

are dependent on the intersections of the best-response curves. For instance, the experimental results show that a CCEA using the best collaborations can outperform a CCEA using the random collaborations as the best-response curves are perpendicular to each other.

3.4.2 The $(\mu \dot{+} \lambda)$-CCEA

We used the well-known $(\mu \dot{+} \lambda)$-selection scheme from evolution strategies (ES), applied them in a CCEA and investigate its performance in our simulations. Fundamentally, an ES individual differs from a CCEA individual. In a $(\mu \dot{+} \lambda)$-ES, an individual, a, is a 3-tuple $a = [\mathbf{x}_a, \boldsymbol{\sigma}_a, f_a(\mathbf{x}_a)]$, comprising of its candidate solution vector $\mathbf{x}_a \in \mathbb{R}^n$, the mutation step size $\boldsymbol{\sigma}_a \in \mathbb{R}^n_+$ and the fitness computed by the objective function being optimized $f : \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$. In a $(\mu \dot{+} \lambda)$-CCEA, a CCEA individual, b, is a 4-tuple $b = [x_b, \sigma_b, \mathbf{c}_b, f_b(\mathbf{c}_b)]$, comprising of a component of the candidate solution[2] $x_b \in \mathbb{R}$, the mutation step size $\sigma_b \in \mathbb{R}_+$, the collaboration vector $\mathbf{c}_b \in \mathbb{R}^n$ and the fitness computed by the objective function that takes the collaboration vector $\mathbf{c}_b$ as the input argument. The major difference is that an ES individual directly uses the candidate solution $\mathbf{x}_a$ to evaluate its fitness while a CCEA individual has to first combine the component of its candidate solution x_b with the components of candidate solutions in other individuals to form a collaboration vector $\mathbf{c}_b$ then use the vector $\mathbf{c}_b$ to evaluate the fitness of the CCEA individual b.

Algorithm 5 describes the details of the $(\mu \dot{+} \lambda)$-CCEA. The CCEA first initializes n number of CCEA populations $\mathcal{Q}^i_k, \forall i \in 1, \ldots, n$. Each CCEA population $\mathcal{Q}^i_k$ consists of μ number of parents that are first evaluated by the random collaboration method (line 5). The best collaboration method cannot be used

[2] We can define a CCEA individual such that the number of dimensions for a component of candidate solution is greater than 1, i.e. $\mathbf{x}_b \in \mathbb{R}^d$ for $d > 1$.

in the beginning, because we need the "best individuals" but none of the individuals are evaluated when the algorithm starts. Algorithm 6 shows how a CCEA individual selects its collaborators randomly. In line 7 of the Algorithm 6, the CCEA individual b selects the collaborators from other populations by using a uniform distribution $\mathcal{U}(1, |\mathcal{Q}^i|)$ where $|\mathcal{Q}^i|$ represents the population size of $\mathcal{Q}^i$ and $\mathcal{U}(1, |\mathcal{Q}^i|)$ uniformly returns the random integer between 1 and $|\mathcal{Q}_i|$. Every individual of the same population has the same probability of becoming the component of the collaboration vector $\mathbf{c}$ for the CCEA individual b. Note that in `RandomCollaborate()`, the fitness of individuals is not considered. Another collaboration method is the *best collaboration method* and it is shown in Algorithm 7. The major difference between `RandomCollaborate()` and `BestCollaborate()` is step 7, where the individuals with the maximum fitness are selected. Both of the procedures return the collaboration vector $\mathbf{c} \in \mathbb{R}^n$ that is used to evaluate the fitness of the CCEA individual b. Finally the collaboration methods used in this thesis are *sequential*, meaning that the CCEA always uses the updated individuals in the populations $\mathcal{Q}^1, \ldots \mathcal{Q}^n$.

After selecting the collaborators in line 5, the CCEA enters the main loop. It then assigns the parents into the population for the next generation $\mathcal{Q}^i_{k+1}$ if the plus selection is used (line 9), otherwise an empty set is assigned (line 11). The next step is to generate λ number of offsprings sequentially. An offspring is first cloned from its parent (line 14). If the mutative σ-self adaptation is used, its mutation step size is updated (line 16). After the mutation step size is updated, the offspring is generated (line 17) by adding a normal distribution random scalar $\mathcal{N}(0, 1)$ (scaled by $\sigma_{q'}$) to $x_{q_{ij}}$. The collaboration vector $\mathbf{c}_{b'}$ is formed by calling either one of the collaboration procedures: either `RandomCollaborate()` and `BestCollaborate()`. The off - spring is then evaluated (line 19) and is added to the population

$\mathcal{Q}^i_{k+1}$ (line 20).

After all the individuals are generated, the selection takes place in step 22. When the plus selection and the comma selection are used, $\mu + \lambda$ number and λ number of individuals are ordered accordingly for their fitness. The best μ individuals in each population $\mathcal{Q}^i$ are selected for the next generation. The algorithm iterates until the termination condition is met.

Algorithm 5: Pseudo Code of the $(\mu \overset{+}{,} \lambda)$-CCEA

1 **Given:** $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma > 0$
2 Initialise $\mathcal{Q}^i_1, \forall i \in \{1, \ldots, n\}, k = 1$
3 **for** $i = 1$ *to* n **do**
4 **for** $j = 1$ *to* μ **do**
5 $\mathbf{c}_{q_{ij}} \leftarrow$ `Collobrate(`$q_{ij}, \mathcal{Q}^1_k, \ldots, \mathcal{Q}^n_k$`)`
6 $f_{q_{ij}} \leftarrow f\left(\mathbf{c}_{q_{ij}}, k\right)$ `/* initialize parent's fitness */`
7 **repeat**
8 **if** *Plus-selection* **then**
9 $\mathcal{Q}_{k+1} \leftarrow \mathcal{Q}_k$
10 **else**
11 $\mathcal{Q}_{k+1} \leftarrow \emptyset$
12 **for** $i = 1$ *to* n **do**
13 **for** $j = 1$ *to* λ **do**
14 $q' \leftarrow q_{ij} \in \mathcal{Q}^i_k$
15 **if** *Self-adaptive* **then**
16 $\sigma_{q'} \leftarrow \sigma_{q_{ij}} \exp\left(\tau \mathcal{N}_i(0,1) + \tau' \mathcal{N}_j(0,1)\right)$
17 $x_{q'} \leftarrow x_{q_{ij}} + \sigma_{q'} \cdot \mathcal{N}(0,1)$
18 $\mathbf{c}_{q'} \leftarrow$ `Collobrate(...)`
19 $f_{q'} \leftarrow f\left(\mathbf{c}_{q'}, k\right)$
20 $\mathcal{Q}^i_{k+1} \leftarrow \mathcal{Q}^i_{k+1} \cup \{q'\}$
21 **for** $i = 1$ *to* n **do**
22 $\mathcal{Q}^i_{k+1} \leftarrow \left\{ b_{j:|\mathcal{Q}^i_{k+1}|} | 1 \leq j \leq \mu \right\}$
23 $k \leftarrow k + 1$
24 **until** *termination condition is met*

Algorithm 6: Random Collaboration

```
RandomCollaborate(b, Q^1, Q^2, ..., Q^{n-1}, Q^n)
Given: c ∈ R^n
for i = to n do
    if b ∈ Q^i then
        c_i ← x_b
    else
        k ← U(1, |Q^i|)
        c_i ← x_{b_k} for b_k ∈ Q^i
Return c
```

Algorithm 7: Best Collaboration

```
BestCollaborate(b, Q^1, Q^2, ..., Q^{n-1}, Q^n)
Given: c ∈ R^n
for i = to n do
    if b ∈ Q^i then
        c_i ← x_b
    else
        b_k ← arg max_{b∈Q^i} f_b
        c_i ← x_{b_k} for b_k ∈ Q^i
Return c
```

3.5 Discussion

3.5.1 Why CMA-ES?

In this thesis, we use CMA-ES as a baseline optimization algorithm when comparing the proposed sampling method in Chapters 4, 5 and 6. We also study its behaviour for dynamic environment in Chapter 7. It is a starting point to further improve optimization techniques in black-box optimization. CMA-ES is chosen in this thesis because of its invariance properties. The importance of invariances in science has long been acknowledged. The invariance of an algorithm with respect to a given transformation of the problem domain is a source of robustness, as any

theoretical or empirical result for a given problem instance can be extended to the whole class of problems obtained by applying the transformation. From a theoretical perspective, this invariance property is a source of robustness. From an algorithmic perspective, it removes the need to tune the algorithm hyperparameters according to some (generally unknown) scale of the fitness function. CMA-ES exhibits the invariances against the rank-preserving transformation of the objective function f, the variance against angle preserving transformation of the search space (rotation, reflection, translation) and the scale invariance. The key invariance property which contributes the most into the good performance of the CMA-ES is invariance with respect to the rotation of the search space. Most of Evolutionary Algorithms lack this property and are therefore outperformed by CMA-ES on problems, where the algorithm may benefit from the learning of correlations between the variables.

3.5.2 Why CCEA?

CCEAs have also been used in the large scale optimization problems [238, 132]. In these works, the CCEA populations are "grouped" in such a way that the interacting variables are jointly optimized while the non-interacting variables are optimized as separable sub-problems in a lower dimension. The works consider each variable as one dimensional sub-problem. The algorithms using different grouping strategies [156, 138] are proposed. All these algorithms attempt to divide the variables such that the non-separable sub-problems are grouped together while the other separable sub-problems are optimized separately. In this way, the CCEA can optimize the high-dimensional problems using a divide-and-conquer method to tackle the curse of dimensionality. However, these works focus on large scale *static* optimization and whether these methods are applicable in the

large scale dynamic optimization has yet to be investigated.

□ **End of chapter.**

Part II

Samplings

Chapter 4

Halfspace Sampling in Evolution Strategies

Natural selection is anything but random.

Richard Dawkins

4.1 Motivations

Evolution Strategies (ESs) are robust random search algorithms. They are designed to minimise objective functions that map a continuous search space $\mathbb{R}^n$ into $\mathbb{R}$. In a $(1+1)$-ES, an offspring is created from a single parent $\mathbf{X}_k \in \mathbb{R}$ at iteration k by adding an *independent* random vector $\boldsymbol{\mathcal{N}}_k$ to $\mathbf{X}_k$. If the offspring is better than its parent, it is then selected to be the next parent $\mathbf{X}_{k+1}$. Otherwise, offspring are iteratively generated until a better solution is found.

The elitist $(1+1)$-ES is the fastest and the most local variant in the ES family, and has been studied in various literature [185, 41]. In the local search scenario, the (1+1)-ES outperforms its non-elitist counterparts. Conventionally, the $(1+1)$-ES implements a step size adaptation - the so-called one-fifth success rule. The basic idea is to increase a step size after a successful step and decrease it after a failure. A history of success probabilities is maintained as a threshold during the course of

optimization so that the strategy increases its step size when the success probability is larger than 1/5, and decreases it otherwise. Intuitively, if "too many" steps are successful, this indicates that the search is too local. If "too few" steps are successful, this indicates that the step size used for sampling is too large. A recent development on $(1+1)$-ES is the implementation of the covariance matrix adaptation (CMA) with rank-one update [115]. In the (1+1)-CMA-ES, it adapts not only the step size σ_k but also the covariance matrix $\mathbf{C}_k \in \mathbb{R}^{n\times n}$ associated with the search distribution of mutation vectors based successful steps.

This chapter introduces a novel sampling method that aims at improving the convergence rates of ES on *unimodal* functions. We propose to divide the search space into two halfspaces in $\mathbb{R}^n$ such that the hyperplane bounding the halfspaces passes through the parent $\mathbf{X}_k$. A proposed ES has a tendency to sample candidate solutions in the positive halfspace in which the previous successful steps are located. All the unsuccessful steps in the recent iterations are also used to estimate the positive halfspace. In order to estimate the optimal positive halfspace that contains as many better solutions as possible, we propose to maintain a new vector $\mathbf{n}_k \in \mathbb{R}^n$. This vector is a normal vector to the hyperplane bounding the halfspaces and is updated as the exponential weighted sum of the previous successful and unsuccessful steps. The idea is similar to that of evolution paths in [107], but here it mainly focuses on the evolution of halfspaces via accumulation of significant information from steps in recent iterations.

We also investigate theoretically the gain we can expect from halfspace sampling on the spherical functions. Linear convergence of the scale-invariant step size $(1+1)$-ES with halfspace sampling is derived and its lower bounds are compared with those of the standard $(1+1)$-ES. We express the convergence rates in terms of expectations of random variables in finite di-

mensions of the search space. Lastly, we derive the expressions for the asymptotic normalised convergence rates when the dimension goes to infinity.

The objective of this chapter is to present the concept of halfspace sampling in a $(1 + 1)$-ES. In detail, the contributions of this chapter are

- to introduce halfspace sampling in the elitist $(1 + 1)$-ES,
- to theoretically investigate algorithms' convergence rates on the spherical functions in finite and infinite dimensions,
- to experimentally compare convergence rates to evaluate the impact of halfspace sampling,
- to implement halfspace sampling in the $(1 + 1)$-CMA-ES and present the empirical performance results.

4.2 Halfspaces

We start with formally defining halfspaces.

Definition 1. *Let* $\mathbf{n}$ *and* $\mathbf{x}_0$ *be two vectors in* $\mathbb{R}^n$*. The set* $H(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) = 0\right\}$ *is a hyperplane in* $\mathbb{R}^n$ *passing through* $\mathbf{x}_0$ *and it has a normal vector* $\mathbf{n}$*. Corresponding to* H*, there is a pair of halfspaces: a positive closed halfspace* $H_+(\mathbf{n}, \mathbf{x}_0)$ *and a negative closed halfspace* $H_-(\mathbf{n}, \mathbf{x}_0)$*:*

$$H_+(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) \geq 0\right\} \tag{4.1}$$

$$H_-(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) \leq 0\right\} \tag{4.2}$$

A halfspace is a convex set and its boundary is the hyperplane H that separates the space into two halfspaces. It defines a reflection that fixes the hyperplane and interchanges two halfspaces. By the definition, we can see that a hyperplane H is

orthogonal to its normal vector $\mathbf{n}$. Any vector $\mathbf{x}$ with respect to the point $\mathbf{x}_0$ forms an acute angle with $\mathbf{n}$ when $\mathbf{x}$ lies in the positive closed halfspace, i.e. $\mathbf{x} \in H_+(\mathbf{n}, \mathbf{x}_0)$. Similarly, it forms an obtuse angle if $\mathbf{x}$ lies in the negative closed halfspace, i.e. $\mathbf{x} \in H_-(\mathbf{n}, \mathbf{x}_0)$. A positive closed halfspace can be derived from its respective negative closed halfspace by reversing the sign of the vector $\mathbf{n}$:

$$H_+(\mathbf{n}, \mathbf{x}_0) = H_-(-\mathbf{n}, \mathbf{x}_0) = -H_-(\mathbf{n}, \mathbf{x}_0) \tag{4.3}$$

We can also derive for the negative halfspace similarly. Lastly, we define a lemma that is useful when describing the optimal halfspaces in the next section:

Lemma 1. *Let $\mathbf{x}, \mathbf{x}_0 \in \mathbb{R}^n$ be two vectors. Assume that $\mathbf{x}_0 + \mathbf{x}$ lies in the positive closed halfspace $H_+(\mathbf{n}, \mathbf{x}_0)$ with a normal vector $\mathbf{n} \in \mathbb{R}^n$. Then, the reflection $\mathbf{x}_0 - \mathbf{x}$ with respect to $\mathbf{x}_0$ lies in the negative closed halfspace $H_-(\mathbf{n}, \mathbf{x}_0)$.*

Proof. If $\mathbf{x}_0 + \mathbf{x}$ lies in the closed positive halfspace $H_+(\mathbf{n}, \mathbf{x}_0)$, $\mathbf{n}^T\mathbf{x}$ must be greater than or equal to zero. Putting a reverse of $\mathbf{x}$ inverts the sign and hence $-\mathbf{n}^T\mathbf{x}$ must be less than or equal to zero. Therefore, by definition, $\mathbf{x}_0 - \mathbf{x}$ must lie in the negative closed halfspace. □

In halfspace sampling, the basic idea is to generate new candidate solutions in a halfspace where the previous successful samples are located. At iteration k, the positive halfspace $H_+(\mathbf{n}, \mathbf{X}_k)$ with respect to the parent $\mathbf{X}_k$ is formed by accumulating the information of successful steps in recent iterations while the negative halfspace $H_-(\mathbf{n}, \mathbf{X}_k)$ is formed from the information of unsuccessful steps. The two halfspaces have the same supporting hyperplanes that passes through the parent $\mathbf{X}_k$. Instead of sampling offspring that can lie in the whole search space $\mathbb{R}^n$, the $(1+1)$-ES with halfspace sampling generates offspring that lies in the positive halfspace $H_+(\mathbf{n}, \mathbf{X}_k)$: If an offspring $\mathbf{X}_k + \boldsymbol{\mathcal{N}}$

lies in the positive halfspace, no action is required, where $\boldsymbol{\mathcal{N}}$ denotes a vector drawn from a standard multivariate normal distribution. Only if an offspring lies in the negative halfspace $\mathbf{X}_k + \boldsymbol{\mathcal{N}} \in H_-(\mathbf{n}, \mathbf{X}_k)$, by Lemma 1, we reverse the direction of the random vector and the offspring will then lie in the positive halfspace i.e. $\mathbf{X}_k - \boldsymbol{\mathcal{N}} \in H_+(\mathbf{n}, \mathbf{X}_k)$. Figure 4.1 illustrates a summary of the geometric interpretation for a parent, two halfspaces and the supporting hyperplane.

Algorithm 8 shows the pseudo code of the $(1 + 1)$-ES with halfspace sampling. We name the resulting strategy as $(1+1_{\text{hs}})$-ES. Consider a scalar objective function $f : \mathbb{R}^n \mapsto \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$; $\mathbf{X}_k$ denotes the current parent at iteration k, and $\sigma_k \in \mathbb{R}_+$ is the step size. The vector $\mathbf{n}_k$ denotes the normal vector of the hyperplane H that passes through the parent $\mathbf{X}_k$. The algorithm starts by initialising $\mathbf{n}_k$ as a null vector $\mathbf{0}$. An instantiation of a multivariate normal distribution $\mathbf{Z}_k \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ with mean vector $\mathbf{0}$ and identity covariance matrix is generated. The vector $\mathbf{Z}_k$ is used if it lies in the positive halfspaces $H_+(\mathbf{n}_k, \mathbf{0})$, otherwise $-\mathbf{Z}_k$ is used.

4.3 Optimal Halfspaces in Convex Sublevel Sets

In a black box optimization scenario, the location of the optimum is unknown. It is necessary to understand the optimal halfspaces, which contains as many better solutions as possible. Understanding the optimal halfspaces is also important for convergence analysis in the next section as it affects the optimal bounds for convergence rates of the $(1+1_{\text{hs}})$-ES. We now derive the optimal halfspaces at any point $\mathbf{x} \in \mathbb{R}^n$ when we optimise objective functions with convex sublevel sets. To achieve this, we first define the relationship between the supporting hyperplane and the tangent hyperplane at any point $\mathbf{x}$.

Definition 2. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$ be an objective function with convex sublevel sets and $\mathbf{x}_0 \in \mathbb{R}^n$ be a point in n-dimensional space. At the point $\mathbf{x}_0$ with an objective function value $f(\mathbf{x}_0)$, the supporting hyperplane contains all the points in the convex sublevel set $C_-(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$ with objective function values smaller than or equal to $f(\mathbf{x}_0)$. If f is differentiable at $\mathbf{x}_0$, then the tangent hyperplane $T(\mathbf{x}_0)$ at $\mathbf{x}_0$ is the supporting hyperplane.*

With this definition, we can derive the below lemma that the optimal halfspaces at a point $\mathbf{x}_0$ are delimited by the supporting hyperplane H at the point $\mathbf{x}_0$.

Lemma 2. *Let f be an objective function with convex sublevel sets and assume f is differentiable. At the point $\mathbf{x}_0 \in \mathbb{R}^n$ with an objective function value $f(\mathbf{x}_0)$, the optimal halfspaces $H_+^*(\mathbf{n}^*, \mathbf{x}_0)$ and $H_-^*(\mathbf{n}^*, \mathbf{x}_0)$ with a normal vector $\mathbf{n}^*$ are delimited by the tangent hyperplane $T(\mathbf{x}_0)$. The optimal positive halfspace $H_+^*(\mathbf{n}^*, \mathbf{x}_0)$ contains all the points in $C_-(\mathbf{x}_0)$.*

Proof. We prove on a geometrically based argumentation. Assume a minimisation for the objective function f. Given a point $\mathbf{x}_0 \in \mathbb{R}^n$, by definition, the tangent hyperplane $T(\mathbf{x}_0)$ at $\mathbf{x}_0$ separates the whole space into two halfspaces. The positive halfspace $H_+(\mathbf{n}, \mathbf{x}_0)$ contains all the points in $C_-(\mathbf{x}_0)$ with objective function values smaller than or equal to $f(\mathbf{x}_0)$ while the negative halfspace contains the points with larger or equal objective values. Hence the optimal halfspaces for the better solutions are delimited by the tangent hyperplane $T(\mathbf{x}_0)$ at $\mathbf{x}_0$. □

By this lemma, the optimal positive halfspace delimited by the tangent hyperplane contains all the better solutions if we are optimising an objective function that has convex sublevel sets. We can additionally establish another lemma that the gradient descent direction at a point $\mathbf{x}_0$ is the normal of the supporting hyperplane for the optimal halfspaces at $\mathbf{x}_0$.

Lemma 3. *Let* $H^*_+(\mathbf{n}^*, \mathbf{x}_0)$ *and* $H^*_-(\mathbf{n}^*, \mathbf{x}_0)$ *be the optimal halfspaces at the point* $\mathbf{x}_0$ *as in Lemma 2. If* f *is differentiable and has the convex sublevel sets, the normal vector* $\mathbf{n}^*$ *is in the form of* $\left(-\frac{\partial f(\mathbf{x}_0)}{\partial x_1}, \ldots, -\frac{\partial f(\mathbf{x}_0)}{\partial x_n}\right)$ *with a gradient descent direction at* $\mathbf{x}_0$.

Proof. By Lemma 2, the tangent hyperplane $T(\mathbf{x}_0)$ separates the space into the optimal halfspaces. By definition of a tangent hyperplane, the normal of a tangent hyperplane at $\mathbf{x}_0$ is $\left(\frac{\partial f(\mathbf{x}_0)}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{x}_0)}{\partial x_n}\right)$. Reversing the direction of this normal gives the normal of the supporting hyperplane that contains all the points in $C_-(\mathbf{x}_0)$. □

Algorithm 8: Pseudo Code of a Simple (1+1)-ES with Halfspace Sampling

```
Initialise X_1, σ_1, n_1 = 0, k = 1
repeat
    Z_k ~ N(0, I)
    if Z_k ∉ H_+(n_k, 0) then
        Z_k = −Z_k
    X'_k = X_k + σ_k Z_k
    if f(X'_k) ≤ f(X_k) then
        X_{k+1} ← X'_k
    else
        X_{k+1} ← X_k
    k = k + 1
    updateHalfSpaces(n_k, Z_k, f(X'_k), f(X_k))
until termination condition is met;
```

4.4 Linear Convergence on Spherical Functions

In this section, we investigate in theory the gain brought by halfspace sampling. We are particularly interested in the linear

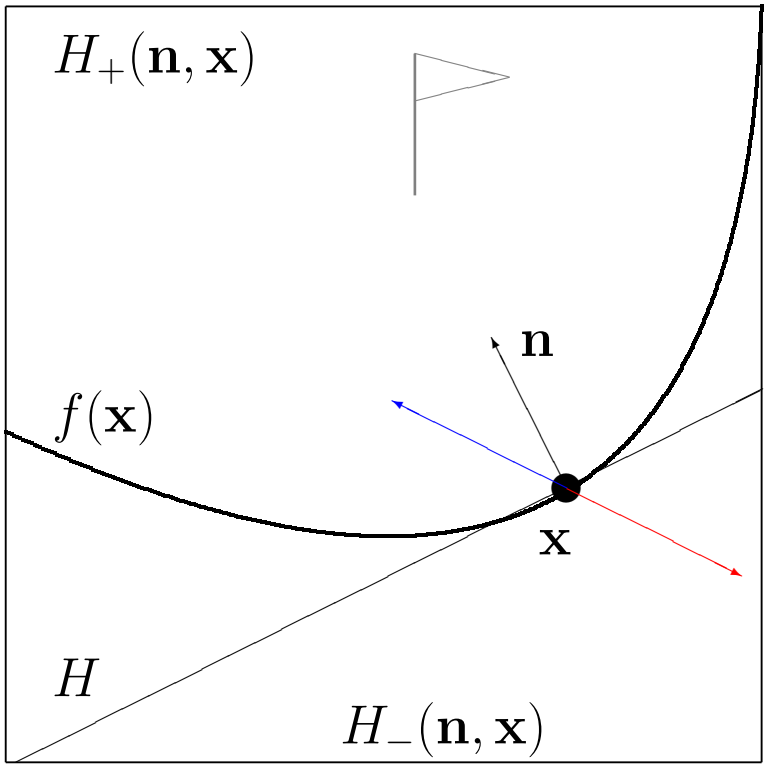

Figure 4.1: Halfspace sampling on an objective function f with convex sublevel sets. The flag in the diagram represents the optimum of the function. The curve represents the points having function value $f(\mathbf{x})$. The supporting hyperplane H (black solid line) with a normal vector $\mathbf{n}$ separates the space into the positive halfspace $H_+(\mathbf{n}, \mathbf{x})$ (the area above the line) and the negative halfspace $H_-(\mathbf{n}, \mathbf{x})$ (the area below the line). If a candidate solution (the red vector) lies in the negative halfspace, it will be discarded. Its reflection (the blue vector) with respect to $\mathbf{x}$, which lies in the positive halfspace, will be used.

convergence rates when the optimal halfspaces are used, i.e., $\mathbf{n}^*$ is known. In practice, the optimal halfspaces are not known and can be estimated only during the course of optimization. However, it is still of interest as we can understand the maximum gain brought by halfspace sampling as well as the optimal bounds of convergence rates of the $(1 + 1_{\text{hs}})$-ES. We refer the resulting ES as the $(1 + 1_{\text{hs}})$-ES* where the optimal halfspaces are known.

Here we use the same definition of the linear convergence rate as [117, 25] and investigate convergence rates on spherical functions with the optimum in zero $g(||\mathbf{x}||), g \in \mathcal{M}$ where $\mathcal{M}$ denotes the set of functions $g : \mathbb{R} \mapsto \mathbb{R}$ that are strictly increasing. We study the case of the scale-invariant step size where $\sigma_k = \sigma||\mathbf{X}_k||$. We refer to the resulting ES as scale-invariant step size ES. For different variants of algorithms with scale-invariant step size, the linear convergence is proved by the following reasons. First, there exists a CR $\in \mathbb{R}$ such that for all $k, k_0 \in \mathbb{R}$ with $k > k_0$

$$\frac{1}{\Lambda}\frac{1}{k - k_0}\mathrm{E}\left[-\ln\frac{||\mathbf{X}_k||}{||\mathbf{X}_{k_0}||}\right] = \mathrm{CR} \tag{4.4}$$

where Λ is the number of function evaluations in each iteration. The convergence rate CR of the algorithm is compatible with the almost sure convergence rate [25]. Therefore, we can prove that with scale-invariant step size, almost surely

$$-\frac{1}{\Lambda}\frac{1}{k - k_0}\ln\frac{||\mathbf{X}_k||}{||\mathbf{X}_{k_0}||} \xrightarrow[k\to\infty]{} \mathrm{CR}$$

For details of proof, please refer to [117, 25].

4.4.1 Preliminaries

Before establishing the main results, we derive some technical results and introduce some useful definitions. In each iteration of

a scale-invariant step size (1+1)-ES, a random independent vector following a standard multivariate normal distribution $\boldsymbol{\mathcal{N}}$ is sampled to create an offspring $\mathbf{X}_k^{'} = \mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$. The distribution of $\sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$ depends in prior on $\mathbf{X}_k$. However, since the distribution of $\boldsymbol{\mathcal{N}}$ is spherical, the distribution of $\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$ will be the same if we start from any vector with unit norm, so WLOG in the scale-invariant step size, the distribution is independent of $\mathbf{X}_k$ and is determined by $||\mathbf{e}_1 + \sigma\boldsymbol{\mathcal{N}}||$.

Lemma 4. *Let $H_+^*(\mathbf{n}^*, \mathbf{e}_1)$ and $H_-^*(\mathbf{n}^*, \mathbf{e}_1)$ be the optimal halfspaces at the point $\mathbf{e}_1$. On spherical functions, the normalised normal vector $\mathbf{n}^*$ is $-\mathbf{e}_1$ and has the gradient descent direction to the optimum.*

Proof. By Lemma 3, the normal vector $\mathbf{n}^*$ has the gradient descent direction and is a form of $\left(-\frac{\partial f(\mathbf{x}_0)}{\partial x_1}, \ldots, -\frac{\partial f(\mathbf{x}_0)}{\partial x_n}\right)$. At the point $\mathbf{e}_1$, the gradient descent direction is $-2\mathbf{e}_1$. After normalisation, it gives $-\mathbf{e}_1$. □

The lemma establishes that the tangent hyperspace with a normal vector $-\mathbf{e}_1$ separates the space into two halfspaces, and the convex sublevel sets $C_-(\mathbf{e}_1)$ lies in the positive halfspaces only. We now establish another lemma that simplifies the event of a candidate solution lying in the optimal halfspaces.

Lemma 5. *Let $H_+^*(\mathbf{n}^*, \mathbf{e}_1)$ and $H_-^*(\mathbf{n}^*, \mathbf{e}_1)$ be the optimal halfspaces at the point $\mathbf{e}_1 = (1, 0, \ldots, 0)$ where $\mathbf{n}^*$ is the vector $-\mathbf{e}_1$. Also let $\mathbf{x}_{\mathbf{e}_1}^+$ be a candidate solution $\mathbf{x}_{\mathbf{e}_1}^+ = \mathbf{e}_1 + \sigma\boldsymbol{\mathcal{N}}$ sampled from the parent $\mathbf{e}_1$. On spherical functions, the two events $\left\{\mathbf{x}_{\mathbf{e}_1}^+ \in H_+^*(\mathbf{n}^*, \mathbf{e}_1)\right\}$ and $\left\{\mathbf{x}_{\mathbf{e}_1}^+ \in H_-^*(\mathbf{n}^*, \mathbf{e}_1)\right\}$ can be written as $\{\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 \leq 0\}$ and $\{\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 \geq 0\}$ respectively.*

Proof. By definition of a positive halfspace, if $\mathbf{x}_{\mathbf{e}_1}^+ \in H_+^*(\mathbf{n}^*, \mathbf{e}_1)$, then $(\mathbf{n}^*)^T\left(\mathbf{x}_{\mathbf{e}_1}^+ - \mathbf{e}_1\right) \geq 0$. By Lemma 4, substituting $-\mathbf{e}_1$ for $\mathbf{n}^*$ and $\mathbf{e}_1 + \sigma\boldsymbol{\mathcal{N}}$ for $\mathbf{x}_{\mathbf{e}_1}^+$ gives $-\mathbf{e}_1^T\sigma\boldsymbol{\mathcal{N}} \geq 0$. Projecting the $\boldsymbol{\mathcal{N}}$ onto the coordinate $\mathbf{e}_1$ and rearranging the terms, the result

$\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 \leq 0$ holds. The event $\left\{\mathbf{x}_{\mathbf{e}_1}^+ \in H_-^*(\mathbf{n}^*, \mathbf{e}_1)\right\}$ can be derived similarly. □

We can establish a similar lemma for the candidate solution $\mathbf{x}_{\mathbf{e}_1}^- = \mathbf{e}_1 - \sigma\boldsymbol{\mathcal{N}}$.

Lemma 6. *Let $\mathbf{x}_{\mathbf{e}_1}^-$ be a candidate solution $\mathbf{x}_{\mathbf{e}_1}^- = \mathbf{e}_1 - \sigma\boldsymbol{\mathcal{N}}$ sampled from the parent $\mathbf{e}_1$. On spherical functions, the two events $\left\{\mathbf{x}_{\mathbf{e}_1}^- \in H_+^*(\mathbf{n}^*, \mathbf{e}_1)\right\}$ and $\left\{\mathbf{x}_{\mathbf{e}_1}^- \in H_-^*(\mathbf{n}^*, \mathbf{e}_1)\right\}$ can be written as $\{\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 \geq 0\}$ and $\{\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 \leq 0\}$ respectively.*

Proof. The proof follows similarly the proof of Lemma 5. □

With lemma 5 and 6, we can establish that on spherical functions, an offspring $\mathbf{x}_{\mathbf{e}_1}^+$ (or $\mathbf{x}_{\mathbf{e}_1}^-$) sampled from the parent $\mathbf{e}_1$ lies in the positive halfspace only if the scalar projection of $\sigma\left[\boldsymbol{\mathcal{N}}\right]_1$ onto the first coordinate $\mathbf{e}_1$ has a negative sign. That implies that a better solution is found only if $\sigma\left[\boldsymbol{\mathcal{N}}\right]_1$ is negative. Otherwise, a worse solution is sampled in the negative halfspace that does not contain the convex sublevel sets $C_-(\mathbf{e}_1)$.

We now need to use the indicator function for the event of an offspring lying in the optimal positive halfspace $H_+^*(\mathbf{n}^*, \mathbf{e}_1)$. We also need the indicator function for the event of an offspring being better than its parent $\mathbf{e}_1$. Therefore, we define the random variables U,V W^+ and W^-:

Definition 3. *Let $U = \sigma\left[\boldsymbol{\mathcal{N}}\right]_1$,$V = -2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma||\boldsymbol{\mathcal{N}}||^2$, $W^+ = 2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma||\boldsymbol{\mathcal{N}}||^2$ and $W^- = -2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma||\boldsymbol{\mathcal{N}}||^2$*

We express the indicators for the events of $\mathbf{x}_{\mathbf{e}_1}^+$ or $\mathbf{x}_{\mathbf{e}_1}^-$ lying in the optimal positive halfspace with respect to $\mathbf{e}_1$ as

$$1_{\left\{\mathbf{x}_{\mathbf{e}_1}^+ \in H_+^*(\mathbf{n}^*, \mathbf{e}_1)\right\}} = 1_{\{U \leq 0\}} \tag{4.5}$$

$$1_{\left\{\mathbf{x}_{\mathbf{e}_1}^- \in H_+^*(\mathbf{n}^*, \mathbf{e}_1)\right\}} = 1_{\{U \geq 0\}} \tag{4.6}$$

and the indicator for the events of $\mathbf{x}_{\mathbf{e}_1}^+$ or $\mathbf{x}_{\mathbf{e}_1}^-$ being better than $\mathbf{e}_1$ as

$$1_{\{\mathbf{x}_{\mathbf{e}_1}^+ \text{ is better than } \mathbf{e}_1\}} = 1_{\{W^+ \leq 0\}} \tag{4.7}$$

$$1_{\{\mathbf{x}_{\mathbf{e}_1}^- \text{ is better than } \mathbf{e}_1\}} = 1_{\{W^- \leq 0\}} \tag{4.8}$$

We now establish one more technical lemma that is useful when we derive the linear convergence rate of $(1 + 1_{\text{hs}})$-ES*.

Lemma 7. *The following two equations hold*

$$\begin{aligned} &\mathrm{E}\left[\ln(1 + \sigma W^+ 1_{\{U \leq 0, W^+ \leq 0\}} + \sigma W^- 1_{\{U > 0, W^- \leq 0\}})\right] \\ =&\mathrm{E}\left[\ln(1 + \sigma W^+ 1_{\{W^+ \leq 0\}} + \sigma W^- 1_{\{W^- \leq 0\}})\right] \end{aligned} \tag{4.9}$$

$$\begin{aligned} &\mathrm{E}\left[\ln(1 + \sigma W^+ \, 1_{\{U \leq 0, W^+ \leq 0\}} + \sigma W^- 1_{\{U > 0, W^- \leq 0\}})\right] \\ &\qquad = \mathrm{E}\left[\ln(1 + \sigma V 1_{\{V \leq 0\}})\right] \end{aligned} \tag{4.10}$$

where $U = \sigma \left[\boldsymbol{\mathcal{N}}\right]_1$, $V = -2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma||\boldsymbol{\mathcal{N}}||^2$, $W^+ = 2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma||\boldsymbol{\mathcal{N}}||^2$ *and* $W^- = -2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma||\boldsymbol{\mathcal{N}}||^2$ *and* $\boldsymbol{\mathcal{N}}$ *is a random vector following a standard multivariate distribution.*

Proof. Consider the event $\{U \leq 0, W^+ \leq 0\}$ to prove equation (4.9). Given $\sigma \in \mathbb{R}^+$, the event $\{W^+ \leq 0\}$ happens only if $U \leq 0$. Hence the event $\{U \leq 0, W^+ \leq 0\}$ can be written as $\{W^+ \leq 0\}$. Similarly, we can write the event $\{U > 0, W^- \leq 0\}$ as $\{W^- \leq 0\}$. Replace the terms in equation (4.9) and the result holds.

To prove equation (4.10), consider the event $\{U \leq 0, W^+ \leq 0\}$. If U is negative, W^+ is the same as V. Similarly in the event $\{U > 0, W^- \leq 0\}$, if U is positive, then W^- is the same as V. Putting these together, the equation (4.10) is simplified to

$$\mathrm{E}\left[\ln(1 + \sigma V 1_{\{U \leq 0, V \leq 0\}} + \sigma V 1_{\{U > 0, V \leq 0\}}).\right]$$

Adding up the second and third terms in the logarithm gives $\sigma V 1_{\{V \leq 0\}}$ and hence the result holds. □

4.4.2 The (1+1)-ES with Optimal Halfspaces

We now prove the linear convergence of the $(1 + 1_{\text{hs}})$-ES* with scale-invariant step size, focus on investigation on the lower bounds of convergence rates for $(1 + 1_{\text{hs}})$-ES*. By comparing its lower bounds with those of a standard $(1 + 1)$-ES, we derive that the maximum gain brought by halfspace sampling is a factor of 2 in finite dimensions.

In a $(1 + 1_{\text{hs}})$-ES* with scale-invariant step size, a single offspring $\mathbf{X}_k^+ = \mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$ is sampled from the parent $\mathbf{X}_k$, where $\boldsymbol{\mathcal{N}}$ is a standard multivariate normal distribution independent of $\mathbf{X}_k$. If the offspring $\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$ lies in the optimal negative halfspace $H_-^*(\mathbf{n}^*, \mathbf{X}_k)$, the direction of $\boldsymbol{\mathcal{N}}$ is reversed and the offspring becomes $\mathbf{X}_k^- = \mathbf{X}_k - \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$. By Lemma 1, the new offspring lies in the optimal positive halfspace $H_+^*(\mathbf{n}^*, \mathbf{X}_k)$. Now, the update equation for $||\mathbf{X}_k||$ on spherical functions reads:

$$\begin{aligned}||\mathbf{X}_{k+1}|| = {} & ||\mathbf{X}_k^+|| \times 1_{\left\{\mathbf{X}_k^+ \in H_+^*(\mathbf{n}^*, \mathbf{X}_k), ||\mathbf{X}_k^+|| \leq ||\mathbf{X}_k||\right\}} \\ & + ||\mathbf{X}_k^-|| \times 1_{\left\{\mathbf{X}_k^+ \notin H_+^*(\mathbf{n}^*, \mathbf{X}_k), ||\mathbf{X}_k^-|| \leq ||\mathbf{X}_k||\right\}} \\ & + ||\mathbf{X}_k|| \times 1_{\left\{\mathbf{X}_k^+ \in H_+^*(\mathbf{n}^*, \mathbf{X}_k), ||\mathbf{X}_k^+|| > ||\mathbf{X}_k||\right\}} \\ & + ||\mathbf{X}_k|| \times 1_{\left\{\mathbf{X}_k^+ \notin H_+^*(\mathbf{n}^*, \mathbf{X}_k), ||\mathbf{X}_k^-|| > ||\mathbf{X}_k||\right\}} \end{aligned} \tag{4.11}$$

where $\mathbf{X}_k^+ = \mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$ and $\mathbf{X}_k^- = \mathbf{X}_k - \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}$. We now establish the following lemma before we prove the linear convergence of the $(1 + 1_{\text{hs}})$-ES* with scale-invariant step size:

Lemma 8. *Let Z_k be the sequence of random variables*

$$\begin{aligned} Z_k = \frac{1}{2} \ln \Big[& ||\mathbf{Y}_k^+||^2 \times 1_{\left\{\mathbf{Y}_k^+ \in H_+^*(\mathbf{n}^*, \mathbf{Y}_k), ||\mathbf{Y}_k^+|| \leq ||\mathbf{Y}_k||\right\}} \\ & + ||\mathbf{Y}_k^-||^2 \times 1_{\left\{\mathbf{Y}_k^+ \notin H_+^*(\mathbf{n}^*, \mathbf{Y}_k), ||\mathbf{Y}_k^-|| \leq ||\mathbf{Y}_k||\right\}} \\ & + ||\mathbf{Y}_k||^2 \times 1_{\left\{\mathbf{Y}_k^+ \in H_+^*(\mathbf{n}^*, \mathbf{Y}_k), ||\mathbf{Y}_k^+|| > ||\mathbf{Y}_k||\right\}} \\ & + ||\mathbf{Y}_k||^2 \times 1_{\left\{\mathbf{Y}_k^+ \notin H_+^*(\mathbf{n}^*, \mathbf{Y}_k), ||\mathbf{Y}_k^-|| > ||\mathbf{Y}_k||\right\}} \Big] \end{aligned} \tag{4.12}$$

where $\mathbf{Y}_k = \mathbf{X}_k/||\mathbf{X}_k||$, $\mathbf{Y}_k^+ = \mathbf{X}_k/||\mathbf{X}_k|| + \sigma\boldsymbol{\mathcal{N}}$ *and* $\mathbf{Y}_k^- = \mathbf{X}_k/||\mathbf{X}_k|| - \sigma\boldsymbol{\mathcal{N}}$. *Then* Z_k *are independent identically distributed as*

$$Z^{(1+1_{\text{hs}})} = \frac{1}{2}\ln\left[1 + \sigma V 1_{\{V\leq 0\}}\right]$$

Proof. Because of the isotropy of the distribution of $\boldsymbol{\mathcal{N}}$ and of sphere function, we can first replace $\mathbf{Y}_k^+$, $\mathbf{Y}_k^-$ and $\mathbf{Y}_k$ by $\mathbf{e}_1^+$, $\mathbf{e}_1^-$ and $\mathbf{e}_1$ respectively, where $\mathbf{e}_1^+ = \mathbf{e}_1 + \sigma\boldsymbol{\mathcal{N}}$ and $\mathbf{e}_1^- = \mathbf{e}_1 - \sigma\boldsymbol{\mathcal{N}}$. The distribution Z_k equals to

$$\begin{aligned} Z_k = \frac{1}{2}\ln\Big[&||\mathbf{e}_1^+||^2 \times 1_{\left\{\mathbf{e}_1^+ \in H_+^*(\mathbf{n}^*,\mathbf{e}_1), ||\mathbf{e}_1^+||\leq 1\right\}} \\ &+ ||\mathbf{e}_1^-||^2 \times 1_{\left\{\mathbf{e}_1^+ \notin H_+^*(\mathbf{n}^*,\mathbf{e}_1), ||\mathbf{e}_1^-||\leq 1\right\}} \\ &+ 1_{\left\{\mathbf{e}_1^+ \in H_+^*(\mathbf{n}^*,\mathbf{e}_1), ||\mathbf{e}_1^+||> 1\right\}} \\ &+ 1_{\left\{\mathbf{e}_1^+ \notin H_+^*(\mathbf{n}^*,\mathbf{e}_1), ||\mathbf{e}_1^-||> 1\right\}}\Big] \end{aligned} \tag{4.13}$$

We then substitute the notations U, W^+ and W^- and expand $||\mathbf{e}_1^+||^2$ as $1 + 2\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma^2||\boldsymbol{\mathcal{N}}||^2$ and $||\mathbf{e}_1^-||^2$ as $1 + 2\sigma\left[\boldsymbol{\mathcal{N}}\right]_1 - \sigma^2||\boldsymbol{\mathcal{N}}||^2$. By Lemma 5 and 6, the term in the logarithm of equation (4.13) is simplified into

$$\begin{aligned} &1_{\{U\leq 0, W^+\leq 0\}} + \sigma W^+ 1_{\{U\leq 0, W^+\leq 0\}} + 1_{\{U>0, W^-\leq 0\}} \\ &+\sigma W^- 1_{\{U>0, W^-\leq 0\}} + 1_{\{U\leq 0, W^+>0\}} + 1_{\{U>0, W^->0\}} \end{aligned} \tag{4.14}$$

We substitute $1_{\{U\leq 0, W^+\leq 0\}} + 1_{\{U>0, W^-\leq 0\}} + 1_{\{U\leq 0, W^+>0\}} + 1_{\{U>0, W^->0\}} = 1$ and the equation (4.14) is simplified as

$$1 + \sigma W^+ 1_{\{U\leq 0, W^+\leq 0\}} + \sigma W^- 1_{\{U>0, W^-\leq 0\}} \tag{4.15}$$

By Lemma 7, we can simplify equation (4.15) into $1 + \sigma V 1_{\{V\leq 0\}}$. Injecting it into equation (4.13), we then obtain the result. □

We are now ready to prove the linear convergence of the $(1 + 1_{\text{hs}})$-ES* and express its convergence rate in terms of $Z^{(1+1_{\text{hs}})}$.

Theorem 1. *The* $(1 + 1_{\text{hs}})$*-ES** *with scale-invariant step size* $(\sigma_k = \sigma||\mathbf{X}_k||)$ *converge linearly on the class of spherical functions* $g(||\mathbf{x}||), g \in \mathcal{M}$*, and*

$$\lim_{k\to\infty} \frac{1}{k} \ln \frac{||\mathbf{X}_k||}{||\mathbf{X}_0||} = \text{CR}_{(1+1_{\text{hs}})}(\sigma) = \frac{1}{2}\text{E}\left[\ln(1 + \sigma V 1_{\{V\leq 0\}}\right] \quad (4.16)$$

where $V = -2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma||\boldsymbol{\mathcal{N}}||^2$ *and* $\boldsymbol{\mathcal{N}}$ *follows a standard multivariate normal distribution.*

Proof. We start with the equation (4.11), square it and normalise the equation by $||\mathbf{X}_k||$ and take the logarithm. We get

$$\begin{aligned}\frac{1}{2}\ln\frac{||\mathbf{X}_{k+1}||^2}{||\mathbf{X}_k||^2} = \frac{1}{2}\ln\Big[&||\mathbf{Y}_k^+||^2 \times 1_{\left\{\mathbf{Y}_k^+ \in H_+^*(\mathbf{n}^*,\mathbf{Y}_k), ||\mathbf{Y}_k^+||\leq||\mathbf{Y}_k||\right\}} \\ &+ ||\mathbf{Y}_k^-||^2 \times 1_{\left\{\mathbf{Y}_k^+ \notin H_+^*(\mathbf{n}^*,\mathbf{Y}_k), ||\mathbf{Y}_k^-||\leq||\mathbf{Y}_k||\right\}} \\ &+ ||\mathbf{Y}_k||^2 \times 1_{\left\{\mathbf{Y}_k^+ \in H_+^*(\mathbf{n}^*,\mathbf{Y}_k), ||\mathbf{Y}_k^+||>||\mathbf{Y}_k||\right\}} \\ &+ ||\mathbf{Y}_k||^2 \times 1_{\left\{\mathbf{Y}_k^+ \notin H_+^*(\mathbf{n}^*,\mathbf{Y}_k), ||\mathbf{Y}_k^-||>||\mathbf{Y}_k||\right\}}\Big]\end{aligned} \quad (4.17)$$

where $\mathbf{Y}_k = \mathbf{X}_k/||\mathbf{X}_k||$, $\mathbf{Y}_k^+ = \mathbf{X}_k/||\mathbf{X}_k|| + \sigma\boldsymbol{\mathcal{N}}$ and $\mathbf{Y}_k^- = \mathbf{X}_k/||\mathbf{X}_k|| - \sigma\boldsymbol{\mathcal{N}}$. According to Lemma 8, by isotropy of the standard multivariate normal distribution, the random variables in the RHS of equation (4.17) are independent and identically distributed as

$$Z^{(1+1_{\text{hs}})} = \frac{1}{2}\ln\left[1 + \sigma V 1_{\{V\leq 0\}}\right].$$

By applying law of large number (LLN) for independent random variable,

$$\frac{1}{k}\ln\frac{||\mathbf{X}_k||}{||\mathbf{X}_0||} = \frac{1}{2k}\sum_{i=0}^{k-1}\ln\frac{||\mathbf{X}_{i+1}||^2}{||\mathbf{X}_i||^2}$$

and it converges to $\text{E}\left[Z^{(1+1_{\text{hs}})}\right]$, hence the result. □

We have derived the linear convergence rate of a $(1+1_{\text{hs}})$-ES*. We now derive the gain brought from halfspace sampling by considering the ratio of $\text{CR}_{(1+1_{\text{hs}})}(\sigma)$ to $\text{CR}_{(1+1)}(\sigma)$.

Theorem 2. *The $(1+1_{\text{hs}})$-ES* with scale-invariant step size ($\sigma_k = \sigma||\mathbf{X}_k||$) converge linearly on the class of spherical functions $g(||\mathbf{x}||), g \in \mathcal{M}$, and the convergence rate is 2 times faster than that of the (1+1)-ES with scale-invariant step size, i.e.*

$$\frac{\text{CR}_{(1+1_{\text{hs}})}(\sigma)}{\text{CR}_{(1+1)}(\sigma)} = 2 \tag{4.18}$$

Proof. First we refer to the equation (7) in [19] for the definition of $\text{CR}_{(1+1)}(\sigma)$. Consider the LHS of the equation (4.18) and by Lemma 7, we can substitute equations (4.9) into (4.18),

$$\begin{aligned}
&\frac{\text{CR}_{(1+1_{\text{hs}})}(\sigma)}{\text{CR}_{(1+1)}(\sigma)} \\
=&\frac{\text{E}\left[\ln(1+\sigma V 1_{\{V\leq 0\}}\right]}{\text{E}\left[\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}}\right]} \\
=&\frac{\text{E}\left[\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}}+\sigma W^- 1_{\{W^-\leq 0\}})\right]}{\text{E}\left[\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}})\right]}
\end{aligned} \tag{4.19}$$

By Lemma 8 in [19],

$$\begin{aligned}
&\text{E}\left[\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}}+\sigma W^- 1_{\{W^-\leq 0\}})\right] \\
=&\text{E}\left[2\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}})\right]
\end{aligned}$$

we can further write the equation (4.19) as

$$\frac{\text{E}\left[2\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}})\right]}{\text{E}\left[\ln(1+\sigma W^+ 1_{\{W^+\leq 0\}})\right]}$$

Hence the result holds. □

4.5 Asymptotic Convergence Rates

In this section, we study how the finite dimension convergence rates derived previously for $(1 + 1_{\text{hs}})$-ES* behave when the dimension goes infinity. We first derive the limit of the probability of success $p^s_{(1+1_{\text{hs}})}(\sigma/d)$ and a technical lemma.

Lemma 9. *For all* $\sigma > 0$,

$$\lim_{n\to\infty} p^s_{(1+1_{\text{hs}})}(\frac{\sigma}{n}) = \Pr\left(|\left[\boldsymbol{\mathcal{N}}\right]_1| \geq \sigma/2\right) = \bar{\Phi}_{\text{HN}}(\frac{\sigma}{2}) \tag{4.20}$$

where $\bar{\Phi}_{\text{HN}}$ *is the complementary cumulative distribution of a standard half-normal distribution,* $\bar{\Phi}_{\text{HN}}(x) = \sqrt{\frac{2}{\pi}} \int_x^\infty \mathrm{e}^{-\frac{x^2}{2}}\,\mathrm{d}x$ *or, with the error function* erf, $\bar{\Phi}_{\text{HN}}(x) = 1 - \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)$.

Proof. We first start from the expression of $p^s_{(1+1_{\text{hs}})}(\sigma/d)$:

$$p^s_{(1+1_{\text{hs}})}(\sigma/d) = \Pr\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2 \leq 0\right) \tag{4.21}$$

$$= \mathrm{E}\left[1_{\left\{-2|[\boldsymbol{\mathcal{N}}]_1|+\frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2\leq 0\right\}}\right] \tag{4.22}$$

By LLN, we can derive that

$$-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2 \xrightarrow[n\to\infty]{} -2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma \tag{4.23}$$

Since $1_{\left\{-2|[\boldsymbol{\mathcal{N}}]_1|+\frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2\leq 0\right\}} \leq 1$, we can apply Lebesgue dominated theorem and therefore

$$\mathrm{E}\left[1_{\left\{-2|[\boldsymbol{\mathcal{N}}]_1|+\frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2\leq 0\right\}}\right] \xrightarrow[n\to\infty]{} \mathrm{E}\left[1_{\{-2|[\boldsymbol{\mathcal{N}}]_1|+\sigma\leq 0\}}\right] \tag{4.24}$$

RHS of equation (4.24) can be rewrite as

$$\begin{aligned}\mathrm{E}\left[1_{\{-2|[\boldsymbol{\mathcal{N}}]_1|+\sigma\leq 0\}}\right] &= \Pr\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma \leq 0\right)\\ &= \Pr\left(|\left[\boldsymbol{\mathcal{N}}\right]_1| \geq \sigma/2\right)\end{aligned}$$

Lastly $\Pr\left(|\left[\boldsymbol{\mathcal{N}}\right]_1| \geq \sigma/2\right) = 1 - \operatorname{erf}\left(\frac{\sigma}{2\sqrt{2}}\right) = \bar{\Phi}_{\text{HN}}(\frac{\sigma}{2})$ □

Lemma 10. *The following equation holds for all* $\sigma > 0$

$$\mathrm{E}\left[\left|\left[\boldsymbol{\mathcal{N}}\right]_1\right| 1_{\{|[\boldsymbol{\mathcal{N}}]_1|\geq\sigma/2\}}\right] = \sqrt{\frac{2}{\pi}}\exp(-\frac{\sigma^2}{8}) \tag{4.25}$$

Proof. First the LHS of equation (4.25) can be written as the density of a standard half-normal distribution as

$$\mathrm{E}\left[\left|\left[\boldsymbol{\mathcal{N}}\right]_1\right| 1_{\{|[\boldsymbol{\mathcal{N}}]_1|\geq\sigma/2\}}\right] = \sqrt{\frac{2}{\pi}}\int_{\sigma/2}^{\infty} x\exp(-\frac{x^2}{2})\,\mathrm{d}x \tag{4.26}$$

Integrating the RHS of equation (4.26) gives the result. □

We now derive the asymptotic convergence rate of the $(1+1_{\mathrm{hs}})$-ES* with scale-invariant step size.

Theorem 3. *Assuming the uniform integrability, for* $\sigma > 0$*, the convergence rate of the* $(1+1_{\mathrm{hs}})$*-ES** *with scale-invariant step size on spherical functions satisfies at the limit*

$$\lim_{n\to\infty} n\times\mathrm{CR}_{(1+1_{\mathrm{hs}})}\left(\frac{\sigma}{n}\right) = -\sqrt{\frac{2\sigma^2}{\pi}}\exp(-\frac{\sigma^2}{8})+\frac{\sigma^2}{2}\bar{\Phi}_{\mathrm{HN}}(\frac{\sigma}{2}) \tag{4.27}$$

Proof. We start by investigating the limit of the random variable of

$$\begin{aligned}&\mathrm{CR}_{(1+1_{\mathrm{hs}})}\left(\frac{\sigma}{n}\right)\\ &= \frac{1}{2}\mathrm{E}\left[\ln\left(1+\frac{\sigma}{n}\min\left(-2\left|\left[\boldsymbol{\mathcal{N}}\right]_1\right|+\frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2, 0\right)\right)\right]\end{aligned} \tag{4.28}$$

The following equation holds

$$\begin{aligned}&\lim_{n\to\infty} n\times\frac{1}{2}\ln(1+\frac{\sigma}{n}\min\left(-2\left|\left[\boldsymbol{\mathcal{N}}\right]_1\right|+\frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2, 0\right))\\ &\qquad\xrightarrow[n\to\infty]{} \frac{1}{2}\sigma\min\left(-2\left|\left[\boldsymbol{\mathcal{N}}\right]_1\right|+\sigma, 0\right)\end{aligned} \tag{4.29}$$

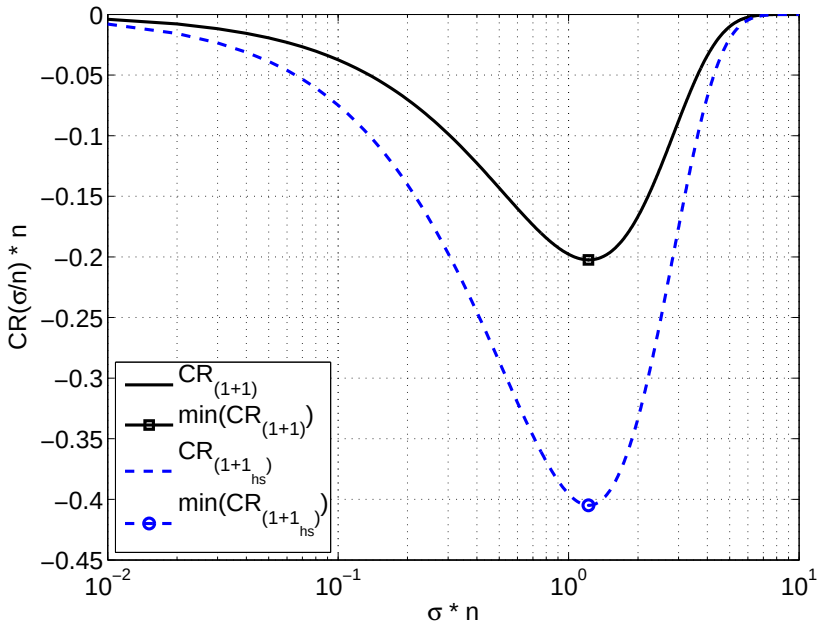

Figure 4.2: Theoretical limits results of the convergence rates for the $(1+1)$-ES in solid line (See Theorem 5 in [19]) and $(1+1_{\text{hs}})$-ES* (Equation (4.27)) in dashed line if the dimension n goes to infinity.

We can further derive that

$$\begin{aligned}
&\lim_{k\to\infty} n \times \mathrm{CR}_{(1+1_{\text{hs}})}\left(\frac{\sigma}{n}\right)\\
=&\frac{\sigma}{2}\mathrm{E}\left[\min\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1|+\sigma, 0\right)\right]\\
=&\frac{\sigma}{2}\mathrm{E}\left[\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1|+\sigma\right)1_{\{-2|[\boldsymbol{\mathcal{N}}]_1|+\sigma\le 0\}}\right]\\
=&-\sigma\mathrm{E}\left[\left(|\left[\boldsymbol{\mathcal{N}}\right]_1|\right)1_{\{-2|[\boldsymbol{\mathcal{N}}]_1|+\sigma\le 0\}}\right]\\
&\quad+\frac{\sigma^2}{2}\mathrm{Pr}\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1|+\sigma\le 0\right)\\
=&-\sigma\mathrm{E}\left[\left(|\left[\boldsymbol{\mathcal{N}}\right]_1|\right)1_{\{|[\boldsymbol{\mathcal{N}}]_1|\ge\sigma/2\}}\right]+\frac{\sigma^2}{2}\mathrm{Pr}\left(|\left[\boldsymbol{\mathcal{N}}\right]_1|\ge\sigma/2\right) \quad (4.30)
\end{aligned}$$

By Lemma 9 and 10, we substitute equations (4.20) and (4.25) into (4.30), then the result follows. □

By Putting together the expressions of asymptotic convergence rates and limits of probabilities of success, we can obtain

the gain brought by halfspace sampling in the $(1+1_{\text{hs}})$-ES* when the dimension n goes to infinity. With Lemma 9, we first derive the limit of the probability of success in the $(1+1_{\text{hs}})$-ES* is 2 times that of the $(1+1)$-ES.

Theorem 4. *For all $\sigma > 0$,*

$$\lim_{n\to\infty} \frac{p^s_{(1+1_{\text{hs}})}(\frac{\sigma}{n})}{p^s_{(1+1)}(\frac{\sigma}{n})} = 2 \tag{4.31}$$

where $p^s_{(1+1)}(\frac{\sigma}{n})$ is the limit of the probability of success of the $(1+1)$-ES.

Proof. By Lemma 13 in [19], the limit of the probability of success of the $(1+1)$-ES can be written as

$$p^s_{(1+1)}(\frac{\sigma}{n}) = \Phi\left(-\frac{\sigma}{2}\right) = \frac{1}{2}\left(1 + \text{erf}(-\frac{\sigma}{2\sqrt{2}})\right) \tag{4.32}$$

Moreover, by Lemma 9, the LHS of equation (4.31) can be simplified into

$$\lim_{n\to\infty} \frac{p^s_{(1+1_{\text{hs}})}(\frac{\sigma}{n})}{p^s_{(1+1)}(\frac{\sigma}{n})} = \frac{1 - \text{erf}(\frac{\sigma}{2\sqrt{2}})}{\frac{1}{2}\left(1 + \text{erf}(-\frac{\sigma}{2\sqrt{2}})\right)} \tag{4.33}$$

Since the error function has the property of

$$\text{erf}(-\frac{\sigma}{2\sqrt{2}}) = -\text{erf}(\frac{\sigma}{2\sqrt{2}}) \tag{4.34}$$

we substitute equation (4.34) into (4.33), we obtain the results. □

Theorem 5. *Assuming the uniform integrability, for $\sigma > 0$, the convergence rate of the $(1+1_{\text{hs}})$-ES* with scale-invariant step size on spherical functions converges at a rate that is two times faster than that of the $(1+1)$-ES with scale-invariant step size, i.e.*

$$\lim_{n\to\infty} \frac{\text{CR}_{(1+1_{\text{hs}})}\left(\frac{\sigma}{n}\right)}{\text{CR}_{(1+1)}\left(\frac{\sigma}{n}\right)} = 2$$

Proof. We start by substituting the CR into the expression as:

$$\begin{aligned}
&\lim_{n\to\infty} \frac{n \times \mathrm{CR}_{(1+1_{\mathrm{hs}})}\left(\frac{\sigma}{n}\right)}{n \times \mathrm{CR}_{(1+1)}\left(\frac{\sigma}{n}\right)} \\
&= \lim_{n\to\infty} \frac{n \ln(1+\frac{\sigma}{n} \min\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2, 0\right))}{n \ln(1+\frac{\sigma}{n} \min\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \frac{\sigma}{n}||\boldsymbol{\mathcal{N}}||^2, 0\right))} \\
&= \frac{\min\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma, 0\right)}{\min\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma, 0\right)}
\end{aligned} \tag{4.35}$$

Because of uniform integrability, we can write the equation as

$$\begin{aligned}
\lim_{n\to\infty} \frac{\mathrm{CR}_{(1+1_{\mathrm{hs}})}\left(\frac{\sigma}{n}\right)}{\mathrm{CR}_{(1+1)}\left(\frac{\sigma}{n}\right)} &= \frac{\mathrm{E}\left[\min\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma, 0\right)\right]}{\mathrm{E}\left[\min\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma, 0\right)\right]} \\
&= \frac{\mathrm{E}\left[\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma\right) 1_{\{-2|[\boldsymbol{\mathcal{N}}]_1|+\sigma\le 0\}}\right]}{\mathrm{E}\left[\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma\right) 1_{\{2[\boldsymbol{\mathcal{N}}]_1+\sigma\le 0\}}\right]} \\
&= \frac{\mathrm{E}\left[\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma\right) 1_{\{|[\boldsymbol{\mathcal{N}}]_1|\ge \sigma/2\}}\right]}{\mathrm{E}\left[\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma\right) 1_{\{[\boldsymbol{\mathcal{N}}]_1\le -\sigma/2\}}\right]}
\end{aligned} \tag{4.36}$$

Since

$$\begin{aligned}
&\left(-2|\left[\boldsymbol{\mathcal{N}}\right]_1| + \sigma\right) 1_{\{|[\boldsymbol{\mathcal{N}}]_1|\ge \sigma/2\}} \\
=&\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma\right) 1_{\{[\boldsymbol{\mathcal{N}}]_1\le -\sigma/2\}} + \left(-2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma\right) 1_{\{[\boldsymbol{\mathcal{N}}]_1\ge \sigma/2\}} \\
=&2\left[\left(2\left[\boldsymbol{\mathcal{N}}\right]_1 + \sigma\right) 1_{\{[\boldsymbol{\mathcal{N}}]_1\le -\sigma/2\}}\right]
\end{aligned} \tag{4.37}$$

Substitute equation (4.37) into (4.36), and the result is obtained. □

Geometrically, we can interpret Theorems 2, 5 and 4 as follows: Consider a point $\mathbf{x} \in \mathbb{R}^n$ in the search space. A new sample is drawn from a standard multivariate normal distribution $\boldsymbol{\mathcal{N}}$ with its center located at $\mathbf{x}$. Since $\boldsymbol{\mathcal{N}}$ is isotropic, the probability of sampling a new point in any pairs of halfspaces with respect to $\mathbf{x}$ are the same. By Lemma 2, the optimal negative halfspace does not contain any points that are better than

$\mathbf{x}$ if the underlying objective function has convex sublevel sets. With respect to the point $\mathbf{x}$, all the points in the negative halfspace are reflected into the positive halfspace. This is actually equivalent to *folding* over the probability mass in the negative halfspace and doubling the probability density in the positive halfspace.

We end this section by presenting a graph to compare the asymptotic convergence rate of a $(1+1)$-ES [19] and that of a $(1+1_{\text{hs}})$-ES* (Equation (4.27)). Figure 4.2 shows the normal convergence rates of both algorithms. The minimal convergence rates of the $(1+1)$-ES and $(1+1_{\text{hs}})$-ES* are approximately -0.202 and -0.404 respectively. By Theorem 5, halfspace sampling speeds up single parent elitist evolution strategies by a factor of 2, regardless of step size σ chosen when the dimensions goes to infinity.

4.6 Numerical Simulations on Convergence rates

We now compare the linear convergence rates of the $(1+1)$-ES with and without halfspace sampling. As the convergence rates are expressed in terms of the expectations of some random variables, we can simulate the convergence rates in finite dimensions by means of the Monte-Carlo method. For every convergence rate expression, we simulate each expectation of a random variable 10^6 times and then average to obtain an estimate of the expectation and hence the convergence rates for different σ. The step size σ is chosen such that it is in the range of $0.01 \leq \sigma \cdot n \leq 10$ and has steps of 0.01 in $\sigma \cdot n$.

Figure 4.3 shows the resulting convergence rates in our simulations versus σ in the dimensions from 2 to 160. Overall in the simulations, the step sizes for the best measured convergence rates for $(1+1_{\text{hs}})$-ES* are approximately the same as those in

$(1+1)$-ES in the corresponding dimensions. Figure 4.3 also shows the best observed convergence rates for the $(1+1_{\text{hs}})$-ES* and the $(1+1)$-ES against the dimensions. By Theorem 2, $(1+1_{\text{hs}})$-ES* converges at a rate which is twice that of a (1+)-ES. In all cases of our simulations in finite dimensions, the best observed convergence rates of the $(1+1_{\text{hs}})$-ES* was approximately 2 times of those of the $(1+1)$-ES. The observed ratios of $\text{CR}_{(1+1_{\text{hs}})}\left(\frac{\sigma}{n}\right)/\text{CR}_{(1+1)}\left(\frac{\sigma}{n}\right)$ are constantly 2.

4.7 Implementation in (1+1)-CMA-ES

In this section, we present the implementation of halfspace sampling in the $(1+1)$-CMA-ES [115]. The $(1+1)$-CMA-ES is basically an extension of the $(1+1)$-ES with a one-fifth success rule where a covariance matrix of the search distribution is adapted during the course of optimization.

A $(1+1)$-CMA-ES consists of a parental candidate solution $\mathbf{X}_k \in \mathbb{R}$, the search path $\mathbf{p}_k \in \mathbb{R}^n$, a global step size $\sigma_k \in \mathbb{R}^+$, the success probability estimate $P^{\text{succ}} \in \mathbb{R}$ and the covariance matrix $\mathbf{C}_k \in \mathbb{R}^{n\times n}$. Consider an objective function $f : \mathbb{R}^n \mapsto \mathbb{R}, \mathbf{X} \mapsto f(\mathbf{X})$. At iteration k, the $(1+1)$-CMA-ES repeats the following steps until the termination condition is met:

1. Determine the Cholesky factor $\mathbf{A}_k \in \mathbb{R}^{n\times n}$ such that $\mathbf{C}_k = \mathbf{A}_k\mathbf{A}_k^{\text{T}}$.
2. Generate the offspring candidate solution $\mathbf{X}_k^{'} = \mathbf{X}_k + \sigma\mathbf{A}_k\mathbf{Z}_k$ where the vector $\mathbf{Z}_k$ is an n-dimensional random vector draw from a standard multivariate normal distribution.
3. If the offspring candidate solution is better than its parent $f(\mathbf{X}_k^{'}) \leq f(\mathbf{X}_k)$,
   (a) Assign $\mathbf{X}_k^{'}$ to $\mathbf{X}_k$.

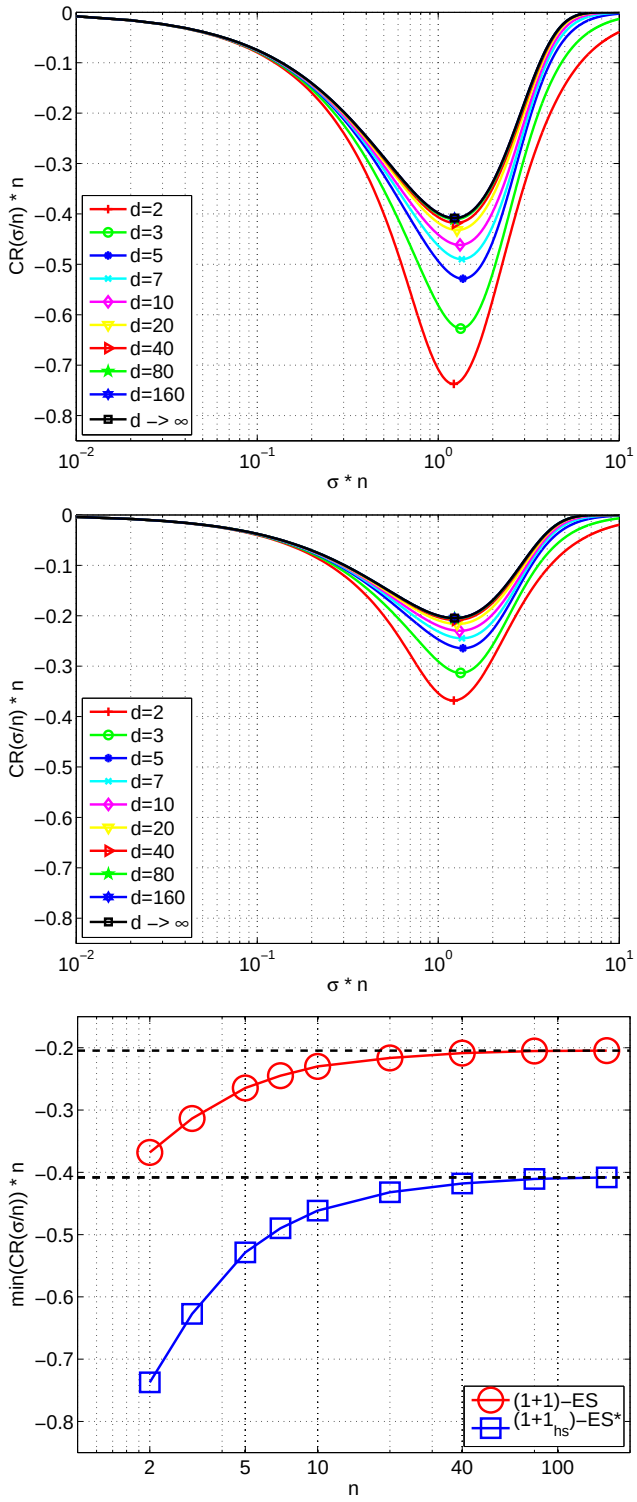

Figure 4.3: Convergence rates $\mathrm{CR}(\sigma)$ of the $(1{+}1_{\mathrm{hs}})$-ES* (top) and the (1+1)-ES (middle) for different dimensions n, all with scale-invariant step size. The bottom graph shows the best observed values of the convergence rates plotted against the dimensions and the dashed lines represent the theoretical limits when $n \to \infty$.

(b) Update the success probability estimate P_{k+1}^{succ} by $(1 - c_p)P_k^{\text{succ}} + c_p$, where c_p is a learning rate $(0 < c_p \leq 1)$.

(c) Update the search path $\mathbf{p}_{k+1}$ by

$$(1 - c_c)\mathbf{p}_k + \mathbf{1}_{\left\{P_{k+1}^{\text{succ}} < P_{\text{thresh}}\right\}} \sqrt{c_c(2 - c_c)}\mathbf{A}_k\mathbf{Z}_k$$

where c_c is the learning rate for the search path and P_{thresh} is a threshold set to 0.44.

(d) Update the covariance matrix $\mathbf{C}_{k+1}$ by

$$\begin{aligned}(1 - c_{\text{cov}} + c_{\text{cov}}\mathbf{1}_{\left\{P_{k+1}^{\text{succ}} < P_{\text{thresh}}\right\}} c_c(2 - c_c))\mathbf{C}_k \\ + c_{\text{cov}}\mathbf{p}_k\mathbf{p}_k^{\text{T}}\end{aligned}$$

where c_{cov} is the learning rate for covariance matrix.

4. Otherwise,

(a) Update the success probability estimate P_{k+1}^{succ} by $(1 - c_p)P_k^{\text{succ}}$.

5. Update the global step size σ_{k+1} by

$$\sigma_k \exp\left(\frac{1}{d}\frac{P_{k+1}^{\text{succ}} - P_{\text{target}}}{1 - P_{\text{target}}}\right)$$

where $d > 0$ is a damping constant and P_{target} is the target success probability.

The basic assumption of theorems in previous sections is that the optimal halfspaces are known. In order to estimate the optimal halfspaces, we introduce the concept of evolution halfspaces. A new vector $\mathbf{n}_k \in \mathbb{R}^n$ is added to the (1+1)-CMA-ES with halfspace sampling . The update of the evolution halfspaces is done by updating $\mathbf{n}_k$ of the hyperplane $H_+(\mathbf{n}_k, \mathbf{X}_k)$. We do not keep track of the evolution of the negative halfspace since the positive and negative halfspaces are symmetric with respect to the parent

$\mathbf{X}_k$. The update for the positive halfspace $H_+(\mathbf{n}_k, \mathbf{X}_k)$ depends on whether a better solution is found or not. The constants c_n^+ and c_n^- are the learning rates for the evolution halfspaces when a successful step and a unsuccessful step is found respectively. Both the factors $\sqrt{c_n^+(2-c_n^+)}$ and $\sqrt{c_n^-(2+c_n^-)}$ normalise the variance of the $\mathbf{n}_k$. The new normal vector $\mathbf{n}_{k+1}$ is a weighted mean of the old normal vector $\mathbf{n}_k$ and either the successful step $\mathbf{A}_k\mathbf{Z}_k$ or the reverse of an unsuccessful step $-\mathbf{A}_k\mathbf{Z}_k$.

Altogether, the $(1+1)$-CMA-ES with halfspace sampling differs from the algorithm given in the previous subsection only in that the following steps are updated or added:

2) Generate the vector $\mathbf{Z}_k$ that is a n-dimensional random vector draw from a standard multivariate normal distribution. If $\mathbf{A}_k\mathbf{Z}_k \in H_+(\mathbf{n}_k, \mathbf{0})$, generate a candidate solution $\mathbf{X}_k^{'} = \mathbf{X}_k + \sigma_k\mathbf{A}_k\mathbf{Z}_k$. Otherwise $\mathbf{X}_k^{'} = \mathbf{X}_k - \sigma_k\mathbf{A}_k\mathbf{Z}_k$.

3e) Update the normal vector by

$$\mathbf{n}_{k+1} = (1-c_n^+)\mathbf{n}_k + \sqrt{c_n^+(2-c_n^+)}\mathbf{A}_k\mathbf{Z}_k$$

where $c_n^+ \in \mathbb{R}^+$ $(0 < c_n^+ \leq 1)$ is the learning rate for the positive halfspace when a successful step is found.

4b) Update the normal vector by

$$\mathbf{n}_{k+1} = (1+c_n^-)\mathbf{n}_k - \sqrt{c_n^-(2+c_n^-)}\mathbf{A}_k\mathbf{Z}_k$$

where $c_n^- \in \mathbb{R}^+$ $(0 < c_n^- \leq 1)$ is the learning rate for the positive halfspace when an unsuccessful step is found.

The values of the constants in the algorithms are summarised in the Table 4.1. All settings for the constants originally in the $(1+1)$-CMA-ES are the same as the default values in [115]. The two new learning rates c_n^+ and c_n^- are obtained by running experiments on sphere function of search space dimensionalities from $n = 2$ to $n = 40$, in each instance choosing the best median performance, and fitting the nonlinear regression curve through the resulting data points against the dimensions.

Table 4.1: Parameter Settings of (1+1)-CMA-ES with halfspace sampling.

$d = 1 + \frac{n}{2}$	$c = \frac{2}{n+2}$	$P_{\text{target}} = \frac{2}{11}$	$c_{\text{p}} = \frac{1}{12}$
$c_{\text{cov}} = \frac{2}{n^2+6}$	$c_n^+ = \frac{4.06}{n^{1.85}+5.5}$	$c_n^- = \frac{0.87}{n^{1.47}-0.61}$	

4.7.1 Results on Noiseless BBOB TestBed

In order to evaluate the benefits of halfspace sampling in (1+1)-CMA-ES, we compared it with a corresponding strategy that does not use halfspace sampling. The $(1+1)$-CMA-ES is tested on a set of noiseless black-box optimization problems from the BBOB-2013 framework [99]. Preliminary results from experiments follow those in [97] on the benchmark functions given in [74, 99]. On unimodal functions, halfspace sampling shows a consistent advantage over the $(1 + 1)$-CMA-ES, except in the f_7 (Step-ellipsoid) and f_{12} (Bent Cigar). In 20D, we found statistically significant differences on f_2 (Ellipsoid Seperable), f_{10} (Ellipsoid) and f_{11} (Discus). In 5D, similar significant differences were found with an addition of f_{14} (Sum of Diff. Power). On multimodal function, halfspace sampling is neither harmful nor beneficial to $(1 + 1)$-CMA-ES, except in f_{21} (Gallagher 101 peaks) and f_{22} (Gallagher 21 peaks) where slight differences in success rate were observed in 20D. Generally, from these preliminary results, halfspace sampling is never observed to be significantly slower on any functions.

4.8 Conclusion and Future Perspective

In this chapter, we have analysed the $(1 + 1)$-ES with halfspace sampling. In halfspace sampling, the whole search space is divided into two halfspaces with respect to the parent. On the objective functions with convex sublevel sets, we prove that the positive halfspace contains the better candidate solutions

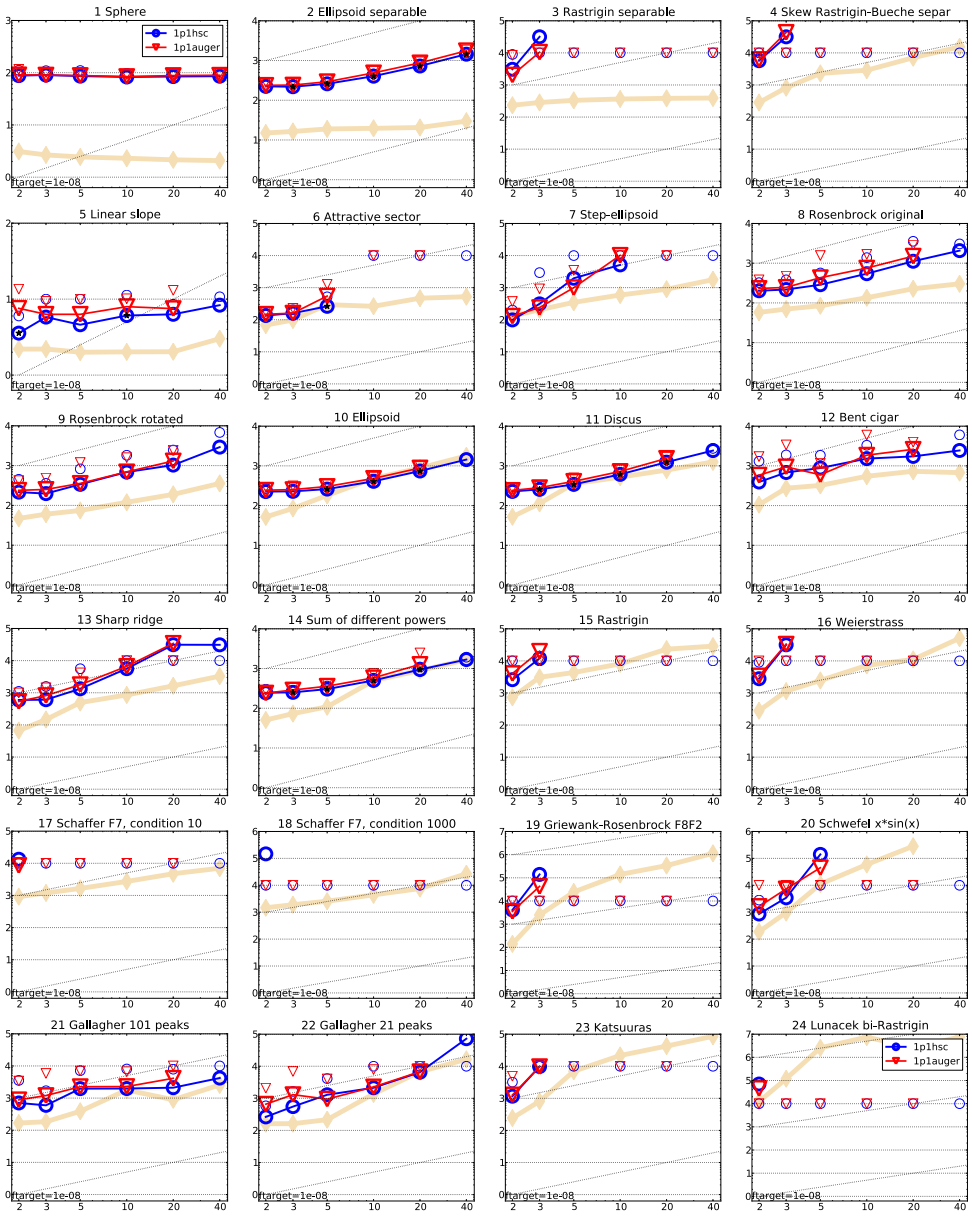

Figure 4.4: Expected running time (ERT in number of f-evaluations) divided by dimension for target function value 10^{-8} as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○:(1+1)-CMA-ES with Halfspace Sampling, ▽:(1+1)-CMA-ES.

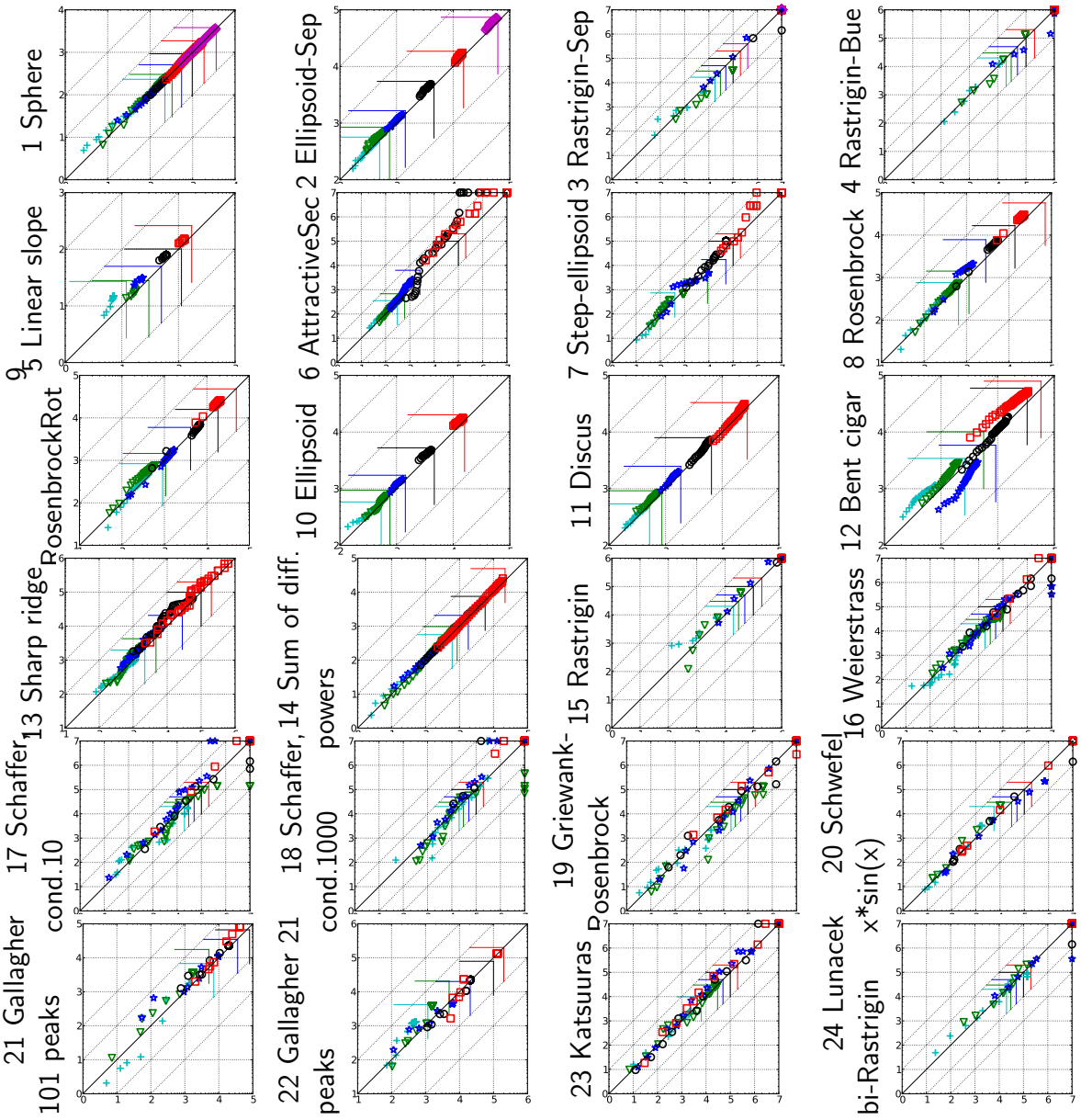

Figure 4.5: Expected running time (ERT in $\log_{10}$ of number of function evaluations) of (1+1)-CMA-ES with Halfspace Sampling (x-axis) versus (1+1)-CMA-ES (y-axis) for 46 target values$\Delta\ f \in [10^{-8}, 10]$ in each dimension on functions f_1–f_{24}. Markers on the upper or right edge indicate that the target value was never reached. Markers represent dimension: 2:+, 3:▽, 5:⋆, 10:○, 20:□, 40:◇.

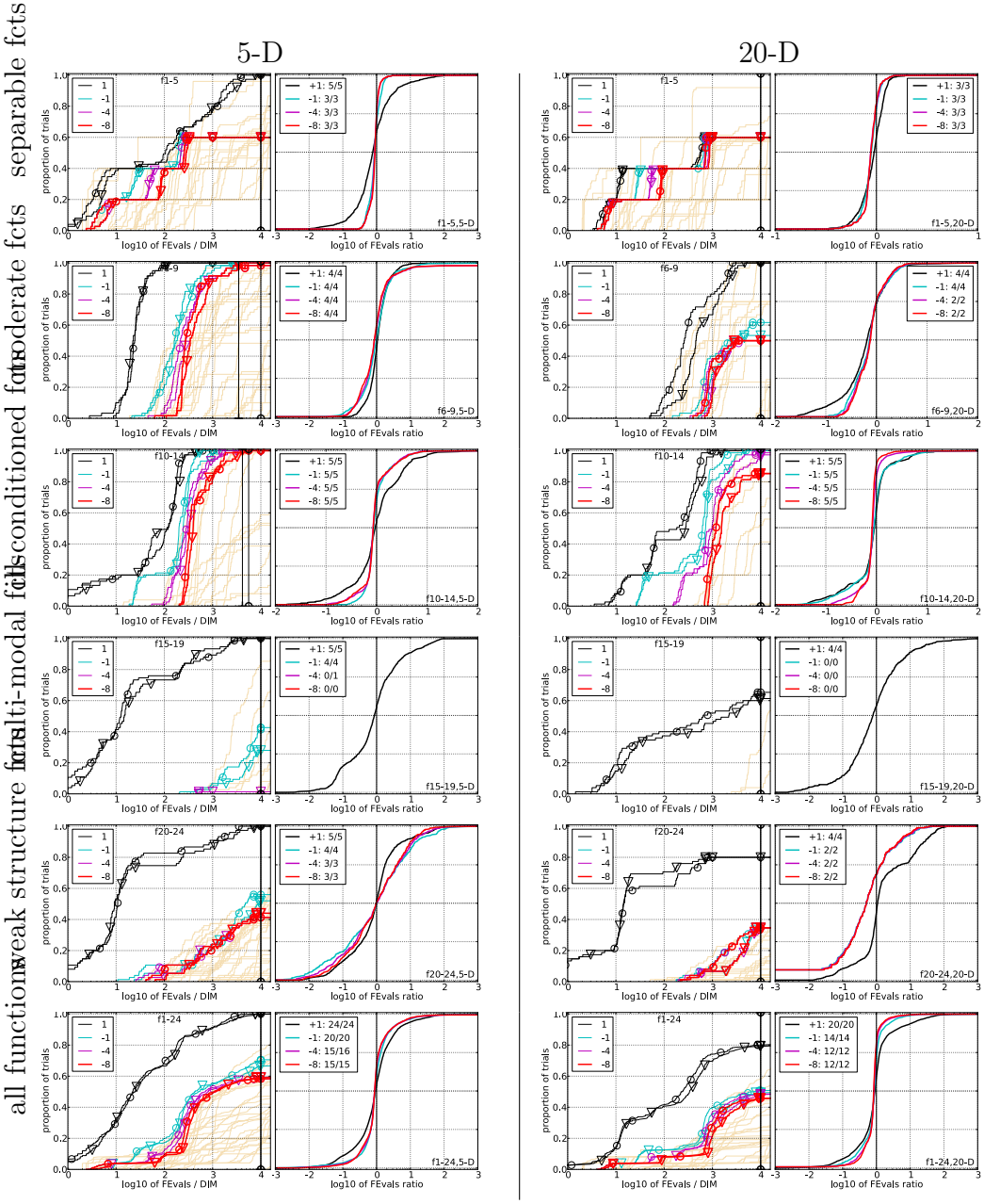

Figure 4.6: Empirical cumulative distributions (ECDF) of run lengths and speed-up ratios in 5-D (left) and 20-D (right). Left sub-columns: ECDF of the number of function evaluations divided by dimension D (FEvals/D) to reach a target value $f_{\text{opt}} + \Delta f$ with$\Delta\ f = 10^k$, where $k \in \{1, -1, -4, -8\}$ is given by the first value in the legend, for (1+1)-CMA-ES with Halfspace Sampling ($\circ$) and (1+1)-CMA-ES ($\triangledown$). Light beige lines show the ECDF of FEvals for target value$\Delta\ f = 10^{-8}$ of all algorithms benchmarked during BBOB-2009. Right sub-columns: ECDF of FEval ratios of (1+1)-CMA-ES with Halfspace Sampling divided by (1+1)-CMA-ES, all trial pairs for each function. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1. The legends indicate the number of functions that were solved in at least one trial ((1+1)-CMA-ES with Halfspace Sampling first).

5-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
f_1	11	12	12	12	12	15/15
1: hs	1.5(1)	10(3)	17(3)	24(3)	32(5)	15/15
2: 1p1	2.3(2)	10(3)	17(3)	25(3)	32(3)	15/15
f_2	83	88	90	92	94	15/15
1: hs	8.0(3)	11(1)	**12**(1)*	**13**(2)*	**13**(1)*	15/15
2: 1p1	9.4(4)	12(3)	14(1)	14(2)	15(2)	15/15
f_3	716	1637	1646	1650	1654	15/15
1: hs	8.4(7)	∞	∞	∞	∞*5.0e4*	0/15
2: 1p1	9.1(9)	∞	∞	∞	∞*5.0e4*	0/15
f_4	809	1688	1817	1886	1903	15/15
1: hs	8.2(8)	∞	∞	∞	∞*5.0e4*	0/15
2: 1p1	15(22)	∞	∞	∞	∞*5.0e4*	0/15
f_5	10	10	10	10	10	15/15
1: hs	1.7(1)	2.3(0.9)	2.3(0.9)	2.3(0.9)	2.3(0.9)	15/15
2: 1p1	2.3(1)	3.2(1)	3.2(1)	3.2(1)	3.2(1)	15/15
f_6	114	281	580	1038	1332	15/15
1: hs	1.3(0.6)	1.3(0.5)	1.0(0.3)	0.84(0.2)	0.89(0.3)	15/15
2: 1p1	1.4(0.9)	1.6(1)	1.6(1)	1.6(1)	1.8(1)	15/15
f_7	24	1171	1572	1572	1597	15/15
1: hs	4.6(2)	2.3(1)	6.2(7)	6.2(7)	6.1(7)	14/15
2: 1p1	3.5(3)	2.0(2)	3.1(4)	3.1(4)	3.1(4)	15/15
f_8	73	336	391	410	422	15/15
1: hs	2.3(1)	2.9(2)	3.0(2)	3.2(2)	3.3(2)	15/15
2: 1p1	2.1(2)	5.0(5)	4.9(4)	4.9(4)	5.0(4)	15/15
f_9	35	214	300	335	369	15/15
1: hs	4.2(1)	5.8(4)	4.8(3)	4.6(2)	4.5(2)	15/15
2: 1p1	4.2(1)	5.9(3)	5.0(2)	4.9(2)	4.7(2)	15/15
f_{10}	349	574	626	829	880	15/15
1: hs	2.2(0.6)	**1.7**(0.1)*	**1.7**(0.1)$^{\star 2}$	**1.4**(0.1)$^{\star 2}$	**1.4**(0.1)$^{\star 2}$	15/15
2: 1p1	2.5(0.9)	2.0(0.3)	2.1(0.3)	1.7(0.2)	1.7(0.2)	15/15
f_{11}	143	763	1177	1467	1673	15/15
1: hs	5.8(3)	**1.7**(0.3)$^{\star 2}$	**1.3**(0.1)$^{\star 3}$	**1.1**(0.1)$^{\star 3}$	**0.99**(0.1)$^{\star 3}$	15/15
2: 1p1	6.5(2)	2.1(0.3)	1.5(0.2)	1.3(0.1)	1.2(0.1)	15/15
f_{12}	108	371	461	1303	1494	15/15
1: hs	8.4(10)	7.0(6)	7.0(6)	2.9(2)	2.9(2)	15/15
2: 1p1	4.0(4)	3.5(2)	4.1(3)	1.9(1)	1.9(1)	15/15
f_{13}	132	250	1310	1752	2255	15/15
1: hs	3.2(3)	5.0(2)	1.5(1)	1.8(0.9)	2.0(1)	15/15
2: 1p1	4.4(4)	6.1(3)	1.9(1)	3.2(4)	3.1(3)	15/15
f_{14}	10	58	139	251	476	15/15
1: hs	1.2(2)	2.2(0.3)	2.8(0.5)	3.6(0.8)	**2.8**(0.2)$^{\star 2}$	15/15
2: 1p1	1.8(3)	2.3(0.5)	2.6(0.8)	4.0(1.0)	3.2(0.3)	15/15
f_{15}	511	19369	20073	20769	21359	14/15
1: hs	12(12)	∞	∞	∞	∞*5.0e4*	0/15
2: 1p1	10(13)	∞	∞	∞	∞*5.0e4*	0/15
f_{16}	120	2662	10449	11644	12095	15/15
1: hs	2.7(4)	22(25)	∞	∞	∞*5.0e4*	0/15
2: 1p1	2.5(5)	17(18)	31(38)	60(67)	∞*5.0e4*	0/15
f_{17}	5.2	899	3669	6351	7934	15/15
1: hs	3.0(4)	24(25)	∞	∞	∞*5.0e4*	0/15
2: 1p1	4.5(5)	108(114)	∞	∞	∞*5.0e4*	0/15
f_{18}	103	3968	9280	10905	12469	15/15
1: hs	5.1(13)	43(44)	∞	∞	∞*5.0e4*	0/15
2: 1p1	4.9(5)	84(101)	∞	∞	∞*5.0e4*	0/15
f_{19}	1	242	1.2e5	1.2e5	1.2e5	15/15
1: hs	22(20)	497(558)	∞	∞	∞*5.0e4*	0/15
2: 1p1	20(17)	971(1085)	∞	∞	∞*5.0e4*	0/15
f_{20}	16	38111	54470	54861	55313	14/15
1: hs	3.3(2)	18(21)	13(14)	13(14)	13(15)	1/15
2: 1p1	2.4(2)	5.9(6)	4.1(5)	4.1(5)	4.1(5)	3/15
f_{21}	41	1674	1705	1729	1757	14/15
1: hs	1.3(0.9)	5.8(8)	5.7(8)	5.7(8)	5.6(8)	15/15
2: 1p1	4.2(10)	6.6(9)	6.5(9)	6.5(9)	6.4(9)	15/15
f_{22}	71	938	1008	1040	1068	14/15
1: hs	1.6(0.7)	6.5(8)	6.2(7)	6.1(7)	6.0(7)	15/15
2: 1p1	2.8(7)	4.7(6)	4.5(6)	4.4(5)	4.4(5)	15/15
f_{23}	3.0	14249	31654	33030	34256	15/15
1: hs	5.3(7)	2.5(2)	∞	∞	∞*5.0e4*	0/15
2: 1p1	4.2(5)	7.5(7)	∞	∞	∞*5.0e4*	0/15
f_{24}	1622	6.4e6	9.6e6	1.3e7	1.3e7	3/15
1: hs	4.1(5)	∞	∞	∞	∞*5.0e4*	0/15
2: 1p1	6.7(8)	∞	∞	∞	∞*5.0e4*	0/15

20-D

Δf	1e+1	1e-1	1e-3	1e-5	1e-7	#succ
f_1	43	43	43	43	43	15/15
1: hs	5.6(0.7)	13(1)	20(1)	28(2)	36(3)	15/15
2: 1p1	5.4(0.8)	13(1)	21(2)	29(2)	37(2)	15/15
f_2	385	387	390	391	393	15/15
1: hs	28(6)	**33**(3)$^{\star 3}$	**35**(1)$^{\star 3}$	**36**(1)$^{\star 3}$	**37**(2)$^{\star 3}$	15/15
2: 1p1	30(5)	39(3)	41(3)	43(2)	44(2)	15/15
f_3	5066	7635	7643	7646	7651	15/15
1: hs	∞	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	∞	∞	∞	∞	∞*2.0e5*	0/15
f_4	4722	7666	7700	7758	1.4e5	9/15
1: hs	∞	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	∞	∞	∞	∞	∞*2.0e5*	0/15
f_5	41	41	41	41	41	15/15
1: hs	2.5(0.6)	3.1(0.8)	3.1(0.8)	3.1(0.8)	3.1(0.8)	15/15
2: 1p1	3.1(0.7)	3.7(0.7)	3.7(0.7)	3.7(0.7)	3.7(0.7)	15/15
f_6	1296	3413	5220	6728	8409	15/15
1: hs	3.5(2)	179(193)	∞	∞	∞*2.0e5*	0/15
2: 1p1	13(15)	835(982)	∞	∞	∞*2.0e5*	0/15
f_7	1351	9503	16524	16524	16969	15/15
1: hs	24(13)	96(104)	∞	∞	∞*2.0e5*	0/15
2: 1p1	30(17)	304(316)	∞	∞	∞*2.0e5*	0/15
f_8	2039	4040	4219	4371	4484	15/15
1: hs	2.6(1)	4.8(4)	4.9(4)	4.9(3)	4.9(3)	15/15
2: 1p1	3.7(1)	6.6(5)	6.6(4)	6.6(4)	6.6(4)	15/15
f_9	1716	3277	3455	3594	3727	15/15
1: hs	3.2(1)	5.4(3)	5.5(3)	5.5(3)	5.5(3)	15/15
2: 1p1	4.5(2)	7.0(4)	7.0(4)	7.0(4)	6.9(4)	15/15
f_{10}	7413	10735	14920	17073	17476	15/15
1: hs	**1.4**(0.2)*	**1.2**(0.1)$^{\star 2}$	**0.93**(0.1)$^{\star 3}$	**0.84**(0.1)$^{\star 3\downarrow 4}$	**0.84**(0.1)$^{\star 3\downarrow 4}$	15/15
2: 1p1	1.7(0.3)	1.4(0.2)	1.1(0.1)	1(0.1)	1(0.1)	15/15
f_{11}	1002	6278	9762	12285	14831	15/15
1: hs	6.9(2)	2.3(0.4)	2.1(0.3)	**1.9**(0.2)$^{\star 2}$	**1.7**(0.1)$^{\star 3}$	15/15
2: 1p1	7.0(1)	2.4(0.3)	2.4(0.4)	2.3(0.2)	2.0(0.1)	15/15
f_{12}	1042	2740	4140	12407	13827	15/15
1: hs	3.1(4)	5.1(5)	4.7(4)	2.1(1)	2.3(1)	15/15
2: 1p1	7.7(12)	10(9)	8.3(6)	3.4(2)	3.6(2)	15/15
f_{13}	652	2751	18749	24455	30201	15/15
1: hs	3.5(3)	8.2(6)	2.5(2)	3.9(4)	10(12)	4/15
2: 1p1	4.9(6)	10(14)	4.2(4)	6.3(6)	14(15)	4/15
f_{14}	75	304	932	1648	15661	15/15
1: hs	3.1(1)	2.2(0.3)	2.0(0.4)	4.6(0.8)	1.0(0.2)	15/15
2: 1p1	3.1(1)	2.3(0.5)	2.3(0.5)	5.6(0.9)	1.2(0.2)	15/15
f_{15}	30378	3.1e5	3.2e5	4.5e5	4.6e5	15/15
1: hs	∞	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	∞	∞	∞	∞	∞*2.0e5*	0/15
f_{16}	1384	77015	1.9e5	2.0e5	2.2e5	15/15
1: hs	36(44)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	34(37)	∞	∞	∞	∞*2.0e5*	0/15
f_{17}	63	4005	30677	56288	80472	15/15
1: hs	19(33)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	29(10)	∞	∞	∞	∞*2.0e5*	0/15
f_{18}	621	19561	67569	1.3e5	1.5e5	15/15
1: hs	969(1177)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	4824(4995)	∞	∞	∞	∞*2.0e5*	0/15
f_{19}	1	3.4e5	6.2e6	6.7e6	6.7e6	15/15
1: hs	558(522)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	1365(1312)	∞	∞	∞	∞*2.0e5*	0/15
f_{20}	82	3.1e6	5.5e6	5.6e6	5.6e6	14/15
1: hs	3.2(0.6)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	3.4(0.9)	∞	∞	∞	∞*2.0e5*	0/15
f_{21}	561	14103	14643	15567	17589	15/15
1: hs	3.6(6)	2.9(2)	2.8(2)	2.7(2)	2.4(2)	15/15
2: 1p1	3.6(7)	5.8(5)	5.6(5)	5.3(5)	4.7(4)	14/15
f_{22}	467	23491	24948	26847	1.3e5	12/15
1: hs	12(16)	5.5(6)	5.2(6)	4.8(6)	0.97(1)	11/15
2: 1p1	3.5(4)	5.6(6)	5.3(6)	5.0(5)	1(1)	12/15
f_{23}	3.2	67457	4.9e5	8.1e5	8.4e5	15/15
1: hs	9.2(15)	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	5.8(9)	∞	∞	∞	∞*2.0e5*	0/15
f_{24}	1.3e6	5.2e7	5.2e7	5.2e7	5.2e7	3/15
1: hs	∞	∞	∞	∞	∞*2.0e5*	0/15
2: 1p1	∞	∞	∞	∞	∞*2.0e5*	0/15

Table 4.2: ERT in number of function evaluations divided by the best ERT measured during BBOB-2009 given in the respective first row with the central 80% range divided by two in brackets for differentΔ f values. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. 1:hs is (1+1)-CMA-ES with Halfspace Sampling and 2:1p1 is (1+1)-CMA-ES. Bold entries are statistically significantly better compared to the other algorithm, with $p = 0.05$ or $p = 10^{-k}$ where $k \in \{2, 3, 4, \dots\}$ is the number following the $\star$ symbol, with Bonferroni correction of 48. A $\downarrow$ indicates the same tested against the best BBOB-2009.

while the negative halfspace contains worse candidate solutions. Therefore, in a $(1+1)$-ES with halfspace sampling, when an offspring lies in the negative halfspace, it will be discarded and its reflection with respect to the parent is used. We prove the linear convergence of a scale-invariant step size $(1+1)$-ES with optimal halfspaces and derive the convergence rates in finite and infinite dimensions. We prove that a $(1+1)$-ES with optimal halfspaces can converge two times faster than a $(1+1)$-ES. Lastly, we implemented halfspace sampling in the $(1+1)$-CMA-ES. Our preliminary results in BBOB 2013 show that $(1+1)$-CMA-ES with halfspace sampling does not appear to be statistically slower than a standard $(1+1)$-CMA-ES.

With halfspace sampling, there are a few areas which we are interested in producing future works. Firstly, we are interested in developing more robust methods of estimating the optimal halfspaces. Although the theorems have proven that a gain can be up to a factor 2, the basic assumption is on the use of optimal halfspaces at a given search point. It becomes a real challenge for ES to reach the theoretical gain closely when halfspace sampling is used. Secondly, we are particularly interested in developing strategies with the use of folded distributions. We have shown that given an objective function with convex sublevel sets, the positive halfspace with respect to a point $\mathbf{x} \in \mathbb{R}^n$ contains the better solution. With regards to folded distributions, we can take advantage of them by sampling in the positive halfspace only. We also plan to incorporate halfspace sampling into population-based evolution strategies with the use of recombinations and to investigate the behavior of halfspace sampling in ES for noisy functions.

□ **End of chapter.**

Chapter 5

Halfspace Sampling in Evolutionary Gradient Search

It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong.

Richard Feynman

5.1 Motivations

Sampling plays an important part in random search algorithms like evolution strategies (ES) [197]. In a family of $(1 \overset{+}{,} \lambda)$-ES, an offspring is created from a single parent $\mathbf{X}_k \in \mathbb{R}$ at iteration k. The offspring is generated by adding an *independent and identical distributed* (i.i.d) random vector $\boldsymbol{\mathcal{N}}_k$ to $\mathbf{X}_k$. The best solution out of these λ sample points (in case of plus selection the best out of λ offspring and the parent $\mathbf{X}_k$) is chosen to become the next parental candidate solution $\mathbf{X}_{k+1}$. The iteration continues until the termination condition is met.

To improve the performance of ES, it is natural to ask whether the replacement of independent random samples by dependent samples works or not. The studies [20, 52] demonstrates how the mirrored sampling can improve the performance of the $(1 \overset{+}{,} \lambda)$-ES. A new sampling method named "Halfspace sampling"

is introduced in previous chapter. In halfspace sampling, the search space is divided into two halfspaces in $\mathbb{R}^n$ such that there is always a hyperplane bounding the halfspaces, which passes through the parent $\mathbf{X}_k$. An ES with halfspace sampling has a tendency to sample candidate solutions in the positive halfspace in which the previous successful steps are located. In order to estimate the optimal positive halfspace that contains as many better solutions as possible, the normal vector $\mathbf{n}_k \in \mathbb{R}^n$ of the hyperplane is maintained during course of optimization and is updated by the exponential weighted sum of the previous successful and unsuccessful steps.

The work [17] also investigates how halfspace sampling can improve the performance of the (1+1)-ES. Specifically, it theoretically proved that on simple spherical functions, a scale-variant step size (1+1)-ES with halfspace sampling converges twice faster than the one without halfspace sampling. However, it is assumed that one knows of the optimal halfspaces with respect to a parent $\mathbf{X}_k$. In practice, the optimal halfspaces are unknown in the black-box optimization. To address this, we implement halfspace sampling in the context of evolutionary gradient search (EGS) [6, 193]. While the EGS uses random samples to estimate the true gradient vector with respect to a parent $\mathbf{X}_k$, we show that random samples can also be used to estimate the optimal halfspaces. We prove the linear convergence of scale-invariant step size EGS with and without halfspace sampling and theoretically investigate algorithms' convergence rates on the spherical functions. We also experimentally compare convergence rates to evaluate the impact of halfspace sampling to an EGS.

5.2 Halfspace Sampling in EGS

5.2.1 Basic EGS

In the basic iteration of an EGS [6, 193], the strategy creates λ offspring from the parent $\mathbf{X}_k$ where k denotes the iteration index. The ith offspring $\mathbf{X}_k + \sigma_k \boldsymbol{\mathcal{N}}_k^i$ is created by adding a vector $\boldsymbol{\mathcal{N}} \in \mathbb{R}^n$ drawn from a standard multivariate normal distribution to the parent $\mathbf{X}_k$, where $\sigma_k \in \mathbb{R}_+$ is the step size. The *mirrored* version of the offspring is then generated by reversing the direction of the random vector and adding it to the parent, *i.e.* $\mathbf{X}_k - \sigma_k \boldsymbol{\mathcal{N}}_k^i$, so-called "inverse mutation" [6]. To compute the progress vector $\mathbf{Z}^{(\text{prog})}$, all offspring in an iteration are collectively used to estimate the gradient by the weighted sum of random vectors $\left(\boldsymbol{\mathcal{N}}_k^i\right)_{1 \leq i \leq \lambda/2}$ with weights proportional to the fitness of offspring. The progress vector is then used to compute the next parent $\mathbf{X}_k + \sigma_k \mathbf{Z}^{(\text{prog})}$. Algorithm 9 shows the pseudo-code of a basic EGS.

5.2.2 EGS with Halfspace Sampling (EGS-HS)

First, we formally define halfspaces. Let $\mathbf{n}$ and $\mathbf{x}_0$ be two vectors in $\mathbb{R}^n$. The hyperplane H in $\mathbb{R}^n$ passing through $\mathbf{x}_0$ is defined as

$$H(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) = 0\right\} \tag{5.1}$$

and it has a normal vector $\mathbf{n}$. Corresponding to H, there is a positive closed halfspace $H_+(\mathbf{n}, \mathbf{x}_0)$ and a negative closed halfspace $H_-(\mathbf{n}, \mathbf{x}_0)$. Formally, they can be defined as:

$$H_+(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) \geq 0\right\} \tag{5.2}$$

$$H_-(\mathbf{n}, \mathbf{x}_0) = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) \leq 0\right\} \tag{5.3}$$

In halfspace sampling [17], the basic idea is to generate new candidate solutions in a halfspace where the previous successful samples are located. At iteration k, the positive halfspace

Algorithm 9: Pseudo Code of the EGS with/without Halfspace Sampling.

Given: $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{X}_1, \boldsymbol{\mathcal{N}}, \mathbf{n} \in \mathbb{R}^n$, $\sigma_1 > 0$
Initialise $\mathbf{X}_1$, σ_1, $k = 1$
repeat
  $f \leftarrow f(\mathbf{X}_k)$ /* fitness of parent */
  $i \leftarrow 1$ /* offspring counter */
  $\mathbf{n}^0 \leftarrow \mathbf{0}$ /* reset normal of positive halfspace */
  while $i \leq \lambda$ **do**
    $\mathbf{Z}_k^i \sim \boldsymbol{\mathcal{N}}(\mathbf{0}, \mathbf{I})$ /* a normal distribution vector */
    if Halfspace Sampling *and* $\mathbf{Z}_k^i \notin H_+(\mathbf{n}^{i-1}, \mathbf{0})$ **then**
      $\mathbf{Z}_k^i \leftarrow -\mathbf{Z}_k^i$ /* reverse the direction */
    else if Not Halfspace Sampling *and* $\mod(i, 2) = 0$ **then**
      $\mathbf{Z}_k^i \leftarrow -\mathbf{Z}_k^{i-1}$ /* use the mirrored sample */
    $\mathbf{Y}^i \leftarrow \mathbf{X}_k + \sigma_k \mathbf{Z}_k^i$ /* generate an offspring */
    $f_i \leftarrow f(\mathbf{Y}^i)$ /* fitness of offspring */
    if Halfspace Sampling **then**
      $\mathbf{n}^i \leftarrow \mathbf{n}^{i-1} + (f - f_i)\mathbf{Z}_k^i$ /* update the normal */
    $i \leftarrow i + 1$
  $\mathbf{Z}^{(\text{avg})} \leftarrow \sum_{i=1}^{\lambda} (f - f_i)\mathbf{Z}_k^i$ /* weight sum of all random vectors */
  $\mathbf{Z}^{(\text{prog})} \leftarrow \frac{\sqrt{n}\mathbf{Z}^{(\text{avg})}}{||\mathbf{Z}^{(\text{avg})}||}$ /* progress vector */
  $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k + \sigma_k \mathbf{Z}^{(\text{prog})}$ /* replace the parent */
  $k \leftarrow k + 1$ /* iteration counter */
until *termination condition is not met*

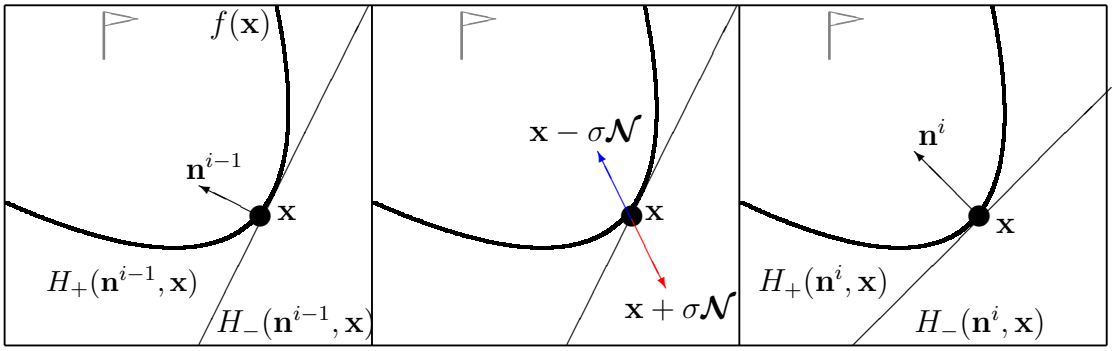

Figure 5.1: Graphical illustration of the ith offspring in the EGS-HS with halfspace sampling. **Left**: The flag and the curve represent the optimum of the function and the points having a function value $f(\mathbf{x})$ respectively. The straight line represents the hyperplane with normal $\mathbf{n}^{i-1}$ and it separates the space into the positive halfspace $H_+(\mathbf{n}^{i-1}, \mathbf{x})$ (the area above the line) and the negative halfspace $H_-(\mathbf{n}^{i-1}, \mathbf{x})$ (the area below the line). **Middle**: If the ith offspring (the red vector) lies in the negative halfspace, it will be discarded. Its reflection (the blue vector) with respect to $\mathbf{x}$, which lies in the positive halfspace, will be used. **Right**: The new normal vector $\mathbf{n}^i$ is replaced by a weighted sum of $\mathbf{n}^{i-1}$ and the resulting random vector used for the ith offspring.

$H_+(\mathbf{n}^{i-1}, \mathbf{X}_k)$ with respect to the parent $\mathbf{X}_k$ is formed by accumulating the information of the previous $i-1$ steps in the *current* iterations. Instead of sampling the ith offspring that lies in the whole search space $\mathbb{R}^n$, the EGS-HS with halfspace sampling generates the ith offspring that always lies in the positive halfspace $H_+(\mathbf{n}^{i-1}, \mathbf{X}_k)$: If the ith offspring $\mathbf{X}_k + \sigma_k \boldsymbol{\mathcal{N}}_k^i$ lies in the positive halfspace, no action is required. Only when the ith offspring does not lie in the positive halfspace $\mathbf{X}_k + \sigma_k \boldsymbol{\mathcal{N}}_k^i \notin H_+(\mathbf{n}^{i-1}, \mathbf{X}_k)$, the direction of the random vector is reversed so the ith offspring lies in the positive halfspace, i.e. $\mathbf{X}_k - \sigma_k \boldsymbol{\mathcal{N}}_k^i \in H_+(\mathbf{n}^{i-1}, \mathbf{X}_k)$. The vector $\mathbf{n}^i$ is computed by the weighted sum of $\mathbf{n}^{i-1}$ and the resulting random vector (either $\boldsymbol{\mathcal{N}}_k^i$ or $-\boldsymbol{\mathcal{N}}_k^i$). Algorithm 9 shows the pseudo-code of the EGS-HS with halfspace sampling. Figure 5.1 graphically illustrates how the ith offspring is generated in the EGS-HS.

5.3 Linear Convergence on Spherical Functions

In this section, we theoretically investigate the log linear convergence of the EGS with and without halfspace sampling. The same definition of the linear convergence rate from [117] is used. We study the case of the scale-invariant step size where the equation $\sigma_k = \sigma||\mathbf{X}_k||$ holds. We investigate the convergence rates on a spherical function having the optimum in zero $g(||\mathbf{x}||), g \in \mathcal{M}$ where $\mathcal{M}$ denotes the set of functions $g : \mathbb{R} \mapsto \mathbb{R}$ that are strictly increasing. For algorithms with scale-invariant step size, the linear convergence is proven by: there exists a $\mathrm{CR} \in \mathbb{R}$ such that for all $k, k_0 \in \mathbb{R}$ with $k > k_0$, $\frac{1}{\Lambda}\frac{1}{k-k_0}\mathrm{E}\left[-\ln\frac{||\mathbf{X}_k||}{||\mathbf{X}_{k_0}||}\right] = \mathrm{CR}$, where Λ is the number of function evaluations in each iteration. For details of proof, please refer to [117, 25].

5.3.1 Progress vector $\mathbf{Z}^{(\mathrm{prog})}$ and $\mathbf{Z}^{(\mathrm{prog})}_{\mathrm{hs}}$

Progress vectors $\mathbf{Z}^{(\mathrm{prog})}$

In the EGS with scale-invariant step size, at iteration k, $\lambda/2$ number of mirrored offspring $\mathbf{X}_k + \sigma||\mathbf{X}_k||(\boldsymbol{\mathcal{N}}^i_k)_{1\leq i\leq \lambda/2}$ and $\mathbf{X}_k - \sigma||\mathbf{X}_k||(\boldsymbol{\mathcal{N}}^i_k)_{1\leq i\leq \lambda/2}$ are sampled from the parent $\mathbf{X}_k$. These $\lambda/2$ offspring are used to compute the vectors $\mathbf{Z}^{(\mathrm{avg})}$ and the progress vector $\mathbf{Z}^{(\mathrm{prog})}$. According to Algorithm 9, the update equation for $||\mathbf{X}_k||$ can be written as

$$||\mathbf{X}_{k+1}|| = ||\mathbf{X}_k + \sigma||\mathbf{X}_k||\mathbf{Z}^{(\mathrm{prog})}|| \tag{5.4}$$

where $\mathbf{Z}^{(\text{prog})} = \sqrt{n}\mathbf{Z}^{(\text{avg})}/||\mathbf{Z}^{(\text{avg})}||$ is the progress vector of the EGS. We can write $\mathbf{Z}^{(\text{avg})}$ at iteration k as[1]:

$$\begin{aligned}
&\mathbf{Z}^{(\text{avg})} \\
&= \sum_{i=1}^{\lambda} \left[f(\mathbf{X}_k) - f(\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i) \right] \boldsymbol{\mathcal{N}}^i \\
&= \sum_{i=1}^{\lambda/2} \left[f(\mathbf{X}_k - \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i) - f(\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i) \right] \boldsymbol{\mathcal{N}}^i \qquad (5.5)
\end{aligned}$$

Expand the two terms inside summation by considering $f(\mathbf{X}_k \pm \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i) = \sum_{j=1}^{n} (X_k^j \pm \sigma||\mathbf{X}_k||\mathcal{N}^{i,j})^2$, where X_k^j and $\mathcal{N}^{i,j}$ denote the j-th component of the vectors $\mathbf{X}_k$ and $\boldsymbol{\mathcal{N}}^i$ respectively. Because of the isotropy of the distribution of $\boldsymbol{\mathcal{N}}$ and of the sphere function, we can write $\mathbf{X}_k = (||\mathbf{X}_k||, 0, \ldots, 0)$. Therefore we get

$$\mathbf{Z}^{(\text{avg})} = -4\sigma||\mathbf{X}_k||^2 \sum_{i=1}^{\lambda/2} [\boldsymbol{\mathcal{N}}^i]_1 \boldsymbol{\mathcal{N}}^i. \qquad (5.6)$$

where $[\boldsymbol{\mathcal{N}}^i]_1$ denotes the projection of $\boldsymbol{\mathcal{N}}^i$ onto $\mathbf{e}_1$. Substitute equation 5.6 into the progress vector $\mathbf{Z}^{(\text{prog})}$, we get:

$$\mathbf{Z}^{(\text{prog})} = -\frac{\sqrt{n}\sum_{i=1}^{\lambda/2}[\boldsymbol{\mathcal{N}}^i]_1 \boldsymbol{\mathcal{N}}^i}{\sqrt{\sum_{j=1}^{n}(\sum_{i=1}^{\lambda/2}[\boldsymbol{\mathcal{N}}^i]_1 \mathcal{N}^{i,j})^2}} \qquad (5.7)$$

Progress vector $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$

In the EGS-HS with scale-invariant step size, λ number of offspring are sampled from the parent $\mathbf{X}_k$

$$\begin{aligned}
&\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i 1_{\{\boldsymbol{\mathcal{N}}^i \in H_+(\mathbf{n}^{i-1}, \mathbf{0})\}} \\
&\quad - \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i 1_{\{\boldsymbol{\mathcal{N}}^i \notin H_+(\mathbf{n}^{i-1}, \mathbf{0})\}} \qquad (5.8)
\end{aligned}$$

[1] We omit the dependence in k for the sampled vectors for the sake of readability.

for $1 \leq i \leq \lambda$ where the vector $\mathbf{n}^i$ is the normal of the positive halfspace after the generation of the ith offspring. The events $1_{\{\boldsymbol{\mathcal{N}}^i \in H_+(\mathbf{n}^{i-1},\mathbf{0})\}}$ and $1_{\{\boldsymbol{\mathcal{N}}^i \notin H_+(\mathbf{n}^{i-1},\mathbf{0})\}}$ represent the event of a standard multivariate normal distributed vector lying in the positive closed halfspace and negative open halfspace respectively. The update equation for $\mathbf{X}_k$ is written as

$$||\mathbf{X}_{k+1}|| = ||\mathbf{X}_k + \sigma||\mathbf{X}_k||\mathbf{Z}_{\text{hs}}^{(\text{prog})}|| \tag{5.9}$$

where $\mathbf{Z}_{\text{hs}}^{(\text{prog})} = \sqrt{n}\mathbf{Z}_{\text{hs}}^{(\text{avg})}/||\mathbf{Z}_{\text{hs}}^{(\text{avg})}||$ is the progress vector of the EGS-HS. Given $\mathbf{n}^0 = \mathbf{0}$, we can write the vector $\mathbf{n}^p$ after the generation of the pth offspring as:

$$\begin{aligned}\mathbf{n}^p = &\left[f\left(\mathbf{X}_k\right) - f(\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^1\right]\boldsymbol{\mathcal{N}}^1 \\ &+ \sum_{i=2}^{p}\Big[f\left(\mathbf{X}_k\right) - f(\mathbf{X}_k + \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i)1_{\{\boldsymbol{\mathcal{N}}^i \in H_+(\mathbf{n}^{i-1},\mathbf{0})\}} \\ &\quad - f(\mathbf{X}_k - \sigma||\mathbf{X}_k||\boldsymbol{\mathcal{N}}^i)1_{\{\boldsymbol{\mathcal{N}}^i \notin H_+(\mathbf{n}^{i-1},\mathbf{0})\}}\Big]\boldsymbol{\mathcal{N}}^i\end{aligned} \tag{5.10}$$

The first term denotes the normal after the first offspring is generated. Now define $V_i = \boldsymbol{\mathcal{N}}^i \cdot \mathbf{n}^{i-1}$, $W_i^+ = 2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma||\boldsymbol{\mathcal{N}}^i||^2$ and $W_i^- = -2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma||\boldsymbol{\mathcal{N}}^i||^2$. Then the events $1_{\{\boldsymbol{\mathcal{N}}^i \in H_+(\mathbf{n}^{i-1},\mathbf{0})\}}$ and $1_{\{\boldsymbol{\mathcal{N}}^i \notin H_+(\mathbf{n}^{i-1},\mathbf{0})\}}$ can be replaced by $1_{\{V_i \geq 0\}}$ and $1_{\{V_i < 0\}}$ respectively. We can further simplify equation 5.10 into:

$$\begin{aligned}\mathbf{n}^p = &\sigma||\mathbf{X}_k||^2 \times \\ &\left(\sum_{i=2}^{p}\left[W_i^- 1_{\{V_i<0\}} - W_i^+ 1_{\{V_i \geq 0\}}\right]\boldsymbol{\mathcal{N}}^i - W_1^+\boldsymbol{\mathcal{N}}^1\right)\end{aligned} \tag{5.11}$$

Because the vector $\mathbf{Z}_{\text{hs}}^{(\text{avg})}$ is the normal vector $\mathbf{n}^\lambda$ after λ offspring are generated, the progress vector $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$ can be written as:

$$\mathbf{Z}_{\text{hs}}^{(\text{prog})} = \frac{\sqrt{n}\mathbf{n}^\lambda}{||\mathbf{n}^\lambda||} = \frac{\sqrt{n}\left(\sum_{i=2}^{\lambda} W_i\boldsymbol{\mathcal{N}}^i - W_1^+\boldsymbol{\mathcal{N}}^1\right)}{\sqrt{\sum_{j=1}^{n}\left(\sum_{i=2}^{\lambda} W_i\mathcal{N}^{i,j} - W_1^+\mathcal{N}^{1,j}\right)^2}} \tag{5.12}$$

where $W_i = W_i^- 1_{\{V_i<0\}} - W_i^+ 1_{\{V_i \geq 0\}}$.

5.3.2 Convergence Rates in Finite Dimensions

Before deriving the main results, we establish a technical lemma.

Lemma 11. *The followings equation holds:*

$$\frac{||\mathbf{X}_k + \sigma||\mathbf{X}_k||\mathbf{Z}^{(\text{prog})}||^2}{||\mathbf{X}_k||^2} = 1 + \sigma(2[\mathbf{Z}^{(\text{prog})}]_1 + \sigma||\mathbf{Z}^{(\text{prog})}||^2) \tag{5.13}$$

$$\frac{||\mathbf{X}_k + \sigma||\mathbf{X}_k||\mathbf{Z}^{(\text{prog})}_{\text{hs}}||^2}{||\mathbf{X}_k||^2} = 1 + \sigma(2[\mathbf{Z}^{(\text{prog})}_{\text{hs}}]_1 + \sigma||\mathbf{Z}^{(\text{prog})}_{\text{hs}}||^2) \tag{5.14}$$

Proof. Define $\mathbf{Y}_k = \mathbf{X}_k/||\mathbf{X}_k||$. We can write LHS of equation 5.13 as $||\mathbf{Y}_k + \sigma\mathbf{Z}^{(\text{prog})}||^2$. Because of the isotropy of the distribution of $\mathbf{Z}^{(\text{prog})}$ and of the sphere function, we can further write it further as $||\mathbf{e} + \sigma\mathbf{Z}^{(\text{prog})}||^2$. Expand the terms and the equation holds. Equation 5.14 can be derived similarly. □

We are now ready to prove the linear convergence of the EGS and the EGS-HS and express their convergence rates in terms of expectations of the progress vectors $\mathbf{Z}^{(\text{prog})}$ and $\mathbf{Z}^{(\text{prog})}_{\text{hs}}$.

Theorem 6. *The EGS having λ offspring in each iteration and the scale-invariant step size ($\sigma_k = \sigma||\mathbf{X}_k||$) converge linearly on the class of spherical functions $g(||\mathbf{x}||), g \in \mathcal{M}$, and*

$$\text{CR}_{\text{EGS}}(\sigma) = \frac{1}{2\lambda}\text{E}\left[\ln(1 + \sigma(2[\mathbf{Z}^{(\text{prog})}]_1 + \sigma||\mathbf{Z}^{(\text{prog})}||^2)\right] \tag{5.15}$$

where $\mathbf{Z}^{(\text{prog})}$ is defined as in equation 5.7.

Proof. We start with the equation 5.4, square it, normalize with respect to $\mathbf{X}_k$, take the logarithm and divide it by $\frac{1}{2}$. The term

$\frac{1}{2}\frac{||\mathbf{X}_{k+1}||^2}{||\mathbf{X}_k||^2}$ is obtained. By Lemma 11, we can write

$$\begin{aligned}
&\frac{1}{2}\frac{||\mathbf{X}_{k+1}||^2}{||\mathbf{X}_k||^2} \\
&= \frac{1}{2}\ln\left(\frac{||\mathbf{X}_k + \sigma||\mathbf{X}_k||\mathbf{Z}^{(\text{prog})}||^2}{||\mathbf{X}_k||^2}\right) \\
&= \frac{1}{2}\ln\left(1 + \sigma(2[\mathbf{Z}^{(\text{prog})}]_1 + \sigma||\mathbf{Z}^{(\text{prog})}||^2)\right) \qquad (5.16)
\end{aligned}$$

Since the RHS of equation 5.16 is always less than ∞, we can apply the Law of Large Numbers (LLN) for this random variable. We then get $\frac{1}{k\lambda}\ln\frac{||\mathbf{X}_k||}{||\mathbf{X}_1||} = \frac{1}{2k\lambda}\sum_{i=1}^{k}\ln\frac{||\mathbf{X}_{i+1}||^2}{||\mathbf{X}_i||^2}$ and thus obtain equation 5.15. □

We can also prove similarly the linear convergence of EGS-HS and derive its convergence rate

Theorem 7. *The EGS-HS having λ offspring in each iteration and the scale-invariant step size ($\sigma_k = \sigma||\mathbf{X}_k||$) converge linearly on the class of spherical functions $g(||\mathbf{x}||), g \in \mathcal{M}$, and*

$$\begin{aligned}
&\text{CR}_{\text{EGS-HS}}(\sigma) \\
&= \frac{1}{2(\lambda+1)}\text{E}\left[\ln(1 + \sigma(2[\mathbf{Z}_{\text{hs}}^{(\text{prog})}]_1 + \sigma||\mathbf{Z}_{\text{hs}}^{(\text{prog})}||^2)\right] \qquad (5.17)
\end{aligned}$$

where $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$ is defined as in equation 5.12.

Proof. The proof follows similarly the same steps in Theorem 6 by considering equations 5.9 and 5.14. Notice that unlike EGS, the fitness of parents in EGS-HS are required to compute $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$ and hence $\lambda + 1$ function evaluations are needed in each iteration. □

5.3.3 Asymptotic Convergence Rates

We have proved the linear convergence rates of the EGS and the EGS-HS in finite dimensions. We now investigate the convergence rates when the dimension goes to infinity. We first

establish a technical lemma to derive the expectation for the projections of progress vectors $\mathbf{Z}^{(\text{prog})}$ and $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$ on $\mathbf{e}_1$.

Lemma 12. *The following two equations hold:*

$$\mathrm{E}\left[[\mathbf{Z}^{(\text{prog})}]_1\right] \overset{n\to\infty}{=} -\mathrm{E}\left[\sqrt{\chi^2_{\frac{\lambda}{2}}}\right] \tag{5.18}$$

$$\mathrm{E}\left[[\mathbf{Z}_{\text{hs}}^{(\text{prog})}]_1\right] \overset{n\to\infty}{=} -\mathrm{E}\left[\frac{2\chi^2_\lambda + \sigma\mathcal{N}}{\sqrt{4\chi^2_\lambda + 2\sigma\mathcal{N} + \sigma^2}}\right] \tag{5.19}$$

where χ^2_k denotes a chi-square distribution with k degree of freedom.

Proof. For Equation 5.18, we start with the definition of $\mathbf{Z}^{(\text{prog})}$. Its projection onto $\mathbf{e}_1$ equals

$$[\mathbf{Z}^{(\text{prog})}]_1 = \sqrt{n}\frac{[\mathbf{Z}^{(\text{avg})}]_1}{||\mathbf{Z}^{(\text{avg})}||}. \tag{5.20}$$

After normalizing the vector $[\mathbf{Z}^{(\text{avg})}]_1$ by $\sigma||\mathbf{X}_k||^2$, we can write:

$$\frac{[\mathbf{Z}^{(\text{avg})}]_1}{\sigma||\mathbf{X}_k||^2} = \left[-4\sum_{i=1}^{\lambda/2}[\boldsymbol{\mathcal{N}}^i]_1\boldsymbol{\mathcal{N}}^i\right]_1 = -4\sum_{i=1}^{\lambda/2}(\mathcal{N}^i)^2 \tag{5.21}$$

Now consider the squared length of the vector $\mathbf{Z}^{(\text{avg})}$ normalized

by $\sigma||\mathbf{X}_k||^2$ and divide it by n. We get

$$\begin{aligned}
\frac{||\frac{\mathbf{Z}^{(\text{avg})}}{\sigma||\mathbf{X}_k||^2}||^2}{n} &= \frac{16}{n}\sum_{j=1}^{n}\left(\sum_{i=1}^{\lambda/2}[\boldsymbol{\mathcal{N}}^i]_1\mathcal{N}^{i,j}\right)^2 \\
&= 16\left(\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{\lambda/2}([\boldsymbol{\mathcal{N}}^i]_1\mathcal{N}^{i,j})^2\right. \\
&\qquad \left. + \underbrace{\frac{1}{n}\sum_{j=1}^{n}\sum_{i\neq k}[\boldsymbol{\mathcal{N}}^i]_1[\boldsymbol{\mathcal{N}}^k]_1\mathcal{N}^{i,j}\mathcal{N}^{k,j}}_{\to 0}\right) \\
&= 16\left(\sum_{i=1}^{\lambda/2}([\boldsymbol{\mathcal{N}}^i]_1)^2 \ \underbrace{\frac{1}{n}\sum_{j=1}^{n}(\mathcal{N}^{i,j})^2}_{\to 1}\right) \\
&= 16\left(\sum_{i=1}^{\lambda/2}(\mathcal{N}^i)^2\right) \qquad (5.22)
\end{aligned}$$

Substitute equation 5.21 and the square root of equation 5.22 into equation 5.20, the RHS of equation 5.18 is obtained. The random variable is χ^2-distributed with $\lambda/2$ degree of freedom and the result holds.

For Equation 5.19, we start with the definition of $\mathbf{Z}_{\text{hs}}^{(\text{prog})}$. Its projection onto $\mathbf{e}_1$ equals

$$[\mathbf{Z}_{\text{hs}}^{(\text{prog})}]_1 = \sqrt{n}\frac{[\mathbf{Z}_{\text{hs}}^{(\text{avg})}]_1}{||\mathbf{Z}_{\text{hs}}^{(\text{avg})}||}. \qquad (5.23)$$

The vector $[\mathbf{Z}_{\text{hs}}^{(\text{avg})}]_1$ is basically $[\mathbf{n}^\lambda]_1$ therefore by equation 5.11,

it can be written as

$$
\begin{aligned}
&[\mathbf{Z}_{\text{hs}}^{(\text{avg})}]_1 \\
&= \left[\sigma||\mathbf{X}_k||^2 \left(\sum_{i=2}^{\lambda} \left[W_i^- 1_{\{V_i<0\}} - W_i^+ 1_{\{V_i\geq 0\}}\right] \boldsymbol{\mathcal{N}}^i - W_1^+ \boldsymbol{\mathcal{N}}^1\right)\right]_1 \\
&= \sigma||\mathbf{X}_k||^2 \left(\sum_{i=2}^{\lambda} \left[W_i^- 1_{\{V_i<0\}} - W_i^+ 1_{\{V_i\geq 0\}}\right] [\boldsymbol{\mathcal{N}}^i]_1 - W_1^+ [\boldsymbol{\mathcal{N}}^1]_1\right)
\end{aligned} \tag{5.24}
$$

Since $\lim_{n\to\infty} \frac{1}{n}||\boldsymbol{\mathcal{N}}||^2 = 1$, we can derive two limits $\lim_{n\to\infty} \frac{W_i^+}{n} = 2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma$ and $\lim_{n\to\infty} \frac{W_i^-}{n} = -2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma$. Normalize the RHS of equation 5.24 by $n\sigma||\mathbf{X}_k||^2$ and then substitute these two limits into it to get

$$
\begin{aligned}
&\lim_{n\to\infty} \left(\frac{[\mathbf{Z}_{\text{hs}}^{(\text{avg})}]_1}{n\sigma||\mathbf{X}_k||^2}\right) \\
&= \sum_{i=2}^{\lambda} \left((-2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma)1_{\{V_i<0\}} - (2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma)1_{\{V_i\geq 0\}}\right) [\boldsymbol{\mathcal{N}}^i]_1 \\
&\quad - (2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma)[\boldsymbol{\mathcal{N}}^1]_1
\end{aligned} \tag{5.25}
$$

Moreover, given that $\boldsymbol{\mathcal{N}}$ is independent, the probability of $\boldsymbol{\mathcal{N}}$ lying either in the positive halfspace or the negative halfspaces delimited by $\mathbf{n}^i, \forall i \in \{1, \ldots, \lambda\}$ is the same. The sum of probabilities must be equal to 1. We can write the event $1_{\{V_i\geq 0\}}$ as

$1_{\{V_i<0\}}$ and vice versa. Therefore we can further simplify

$$\begin{aligned}
&\lim_{n\to\infty}\left(\frac{[\mathbf{Z}_{\text{hs}}^{(\text{avg})}]_1}{n\sigma||\mathbf{X}_k||^2}\right)\\
&= -2\sum_{i=2}^{\lambda}[\boldsymbol{\mathcal{N}}^i]_1[\boldsymbol{\mathcal{N}}^i]_1 - (2[\boldsymbol{\mathcal{N}}^i]_1+\sigma)[\boldsymbol{\mathcal{N}}^1]_1\\
&= -2\sum_{i=1}^{\lambda}(\mathcal{N}^i)^2 - \sigma\mathcal{N}^1 \qquad (5.26)
\end{aligned}$$

Now consider the squared length of the vector $\mathbf{Z}_{\text{hs}}^{(\text{avg})}$ normalized by $\sigma||\mathbf{X}_k||^2$ and divide it by n, we get:

$$\begin{aligned}
&\lim_{n\to\infty}\left(\frac{||\frac{\mathbf{Z}_{\text{hs}}^{(\text{avg})}}{\sigma||\mathbf{X}_k||^2}||^2}{n}\right)\\
&= \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=2}^{\lambda}\left[W_i^-1_{\{V_i<0\}} - W_i^+1_{\{V_i\geq 0\}}\right]\mathcal{N}^{i,j} - W_1^+\mathcal{N}^{1,j}\right)^2\\
&= \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=2}^{\lambda}W_i\mathcal{N}^{i,j} - W_1^+\mathcal{N}^{1,j}\right)^2 \qquad (5.27)\\
& \qquad (5.28)
\end{aligned}$$

Further,

$$\begin{aligned}
&\frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=2}^{\lambda} W_i\mathcal{N}^{i,j} - W_1^+\mathcal{N}^{1,j}\right)^2 \\
&= \frac{1}{n}\sum_{j=1}^{n}\sum_{i=2}^{\lambda}(W_i\mathcal{N}^{i,j})^2 - \underbrace{\frac{1}{n}\sum_{j=1}^{n}\sum_{i\neq k} W_iW_k\mathcal{N}^{i,j}\mathcal{N}^{k,j}}_{\to 0} \\
&\quad \underbrace{-\frac{2}{n}\sum_{j=1}^{n}\sum_{i=2}^{\lambda} W_iW_1^+\mathcal{N}^{i,j}\mathcal{N}^{1,j}}_{\to 0} + \frac{1}{n}\sum_{j=1}^{n}(W_i^+\mathcal{N}^{i,j})^2 \\
&= \sum_{i=2}^{\lambda}(W_i)^2\underbrace{\frac{1}{n}\sum_{j=1}^{n}(\mathcal{N}^{i,j})^2}_{\to 1} + \sum_{i=2}^{\lambda}(W_i^+)^2\underbrace{\frac{1}{n}\sum_{j=1}^{n}(\mathcal{N}^{i,j})^2}_{\to 1} \\
&= 4\sum_{i=2}^{\lambda}(\mathcal{N}^i)^2 + (2\mathcal{N}^1 + \sigma)^2 \\
&= 4\sum_{i=1}^{\lambda}(\mathcal{N}^i)^2 + 2\sigma\mathcal{N}^1 + \sigma^2 \qquad (5.29)
\end{aligned}$$

Substitute equation 5.26 and the square root of equation 5.29 into equation 5.19, the RHS of equation 5.19 is obtained. The random variable is χ^2-distributed with λ degree of freedom, hence the result holds.

□

Theorem 8. *Let χ_k^2 be a chi-square distribution with k degree of freedom. For $\sigma > 0$, the convergence rate of the EGS with λ offspring and scale-invariant step size on spherical functions satisfies at the limit*

$$\lim_{n\to\infty} n \times \mathrm{CR}_{\mathrm{EGS}}\left(\frac{\sigma}{n}\right) = \frac{1}{\lambda}\left(\frac{\sigma^2}{2} - \sigma\mathrm{E}\left[\sqrt{\chi^2_{\frac{\lambda}{2}}}\right]\right). \qquad (5.30)$$

Proof. We start by investigating the limit of the random variable of

$$\mathrm{CR}_{\mathrm{EGS}}\left(\frac{\sigma}{n}\right) = \frac{1}{2\lambda}\mathrm{E}\left[\ln(1+\frac{\sigma}{n}(2[\mathbf{Z}^{(\mathrm{prog})}]_1 + \frac{\sigma}{n}||\mathbf{Z}^{(\mathrm{prog})}||^2)\right] \tag{5.31}$$

By considering $\lim_{n\to\infty}\frac{1}{n}||\mathbf{Z}^{(\mathrm{prog})}||^2 = 1$ and the fact that

$$\ln(1+h) = h + \mathcal{O}(h^2)$$

the following equation holds almost surely

$$\begin{aligned}\lim_{n\to\infty} n \times \frac{1}{2\lambda}\ln(1+\frac{\sigma}{n}(2[\mathbf{Z}^{(\mathrm{prog})}]_1 + \frac{\sigma}{n}||\mathbf{Z}^{(\mathrm{prog})}||^2)) \\ \xrightarrow[n\to\infty]{} \frac{\sigma}{2\lambda}(2[\mathbf{Z}^{(\mathrm{prog})}]_1 + \sigma)\end{aligned} \tag{5.32}$$

Assuming the uniform integrability for LHS of equation 5.32, we can further deduce that

$$\lim_{n\to\infty} n \times \mathrm{CR}_{\mathrm{EGS}}\left(\frac{\sigma}{n}\right) = \mathrm{E}\left[\frac{\sigma}{2\lambda}(2[\mathbf{Z}^{(\mathrm{prog})}]_1 + \sigma)\right]. \tag{5.33}$$

By equation 5.18, substitute the expectation for the projection of $\mathbf{Z}^{(\mathrm{prog})}$ on e and the result holds. □

Similarly we can derive the asymptotic convergence rates of the EGS-HS.

Theorem 9. *Let χ^2_k be a chi-square distribution with k degree of freedom. For $\sigma > 0$, the convergence rate of the EGS-HS with λ offspring and scale-invariant step size on spherical functions satisfies at the limit*

$$\begin{aligned}&\lim_{n\to\infty} n \times \mathrm{CR}_{\mathrm{EGS-HS}}\left(\frac{\sigma}{n}\right) \\ &= \frac{1}{\lambda+1}\left(\frac{\sigma^2}{2} - \sigma\mathrm{E}\left[\frac{2\chi^2_\lambda + \sigma\mathcal{N}}{\sqrt{4\chi^2_\lambda + 2\sigma\mathcal{N} + \sigma^2}}\right]\right)\end{aligned} \tag{5.34}$$

5.4 Numerical Simulations

To compare the linear convergence rates of the EGS with and without halfspace sampling, we simulate the convergence rates by means of the Monte-Carlo method. For every convergence rate expression, we simulated each expectation of a random variable 10^6 times and averaged the results to obtain an estimate of the expectation. The step size σ is chosen such that it is in the range of $0.01 \leq \sigma \cdot d \leq 10$ and has steps of 0.01 in $\sigma \cdot d$.

In Figure 5.2, the two graphs in the top row show the convergence rates versus σ in the dimensions from 3 to 160. Overall, the step sizes for the best measured convergence rates for the EGS-HS are larger than those of the EGS in the respective dimensions. For instance, when n is 80 and λ is 2, the minimum $\mathrm{CR}_{\mathrm{EGS}}(\sigma)$ and $\mathrm{CR}_{\mathrm{EGS-HS}}(\sigma)$ are -0.1623 and -0.2295. The observed step sizes for EGS and EGS-HS are 0.81 and 1.03 respectively.

Figure 5.2 also shows the asymptotic convergence rates when λ are 2, 5 and 50. Increasing λ improves the convergence rates for both strategies. It also increases the step sizes for the best measured convergence rates. For instance, the observed best convergence rates in EGS for $\lambda = \{2, 10, 50\}$ are -0.159, -0.2265 and -0.2451 respectively. The observed step sizes are 0.8, 2.13 and 4.95 respectively. When halfspace sampling is used, the convergence rates are improved to -0.2263, -0.3668 and -0.4107 respectively. The observed step sizes increase to 1.03, 2.47 and 5.63. Computing the ratio of convergences rates, EGS is improved by 42%, 62% and 68%.

Lastly in Figure 5.3, we plot two graphs to show the best observed convergence rates against the dimensions n and the number of offspring λ. In all cases, the convergence rates of EGS-HS is faster than those of EGS regardless of dimensions n and the number of offspring λ.

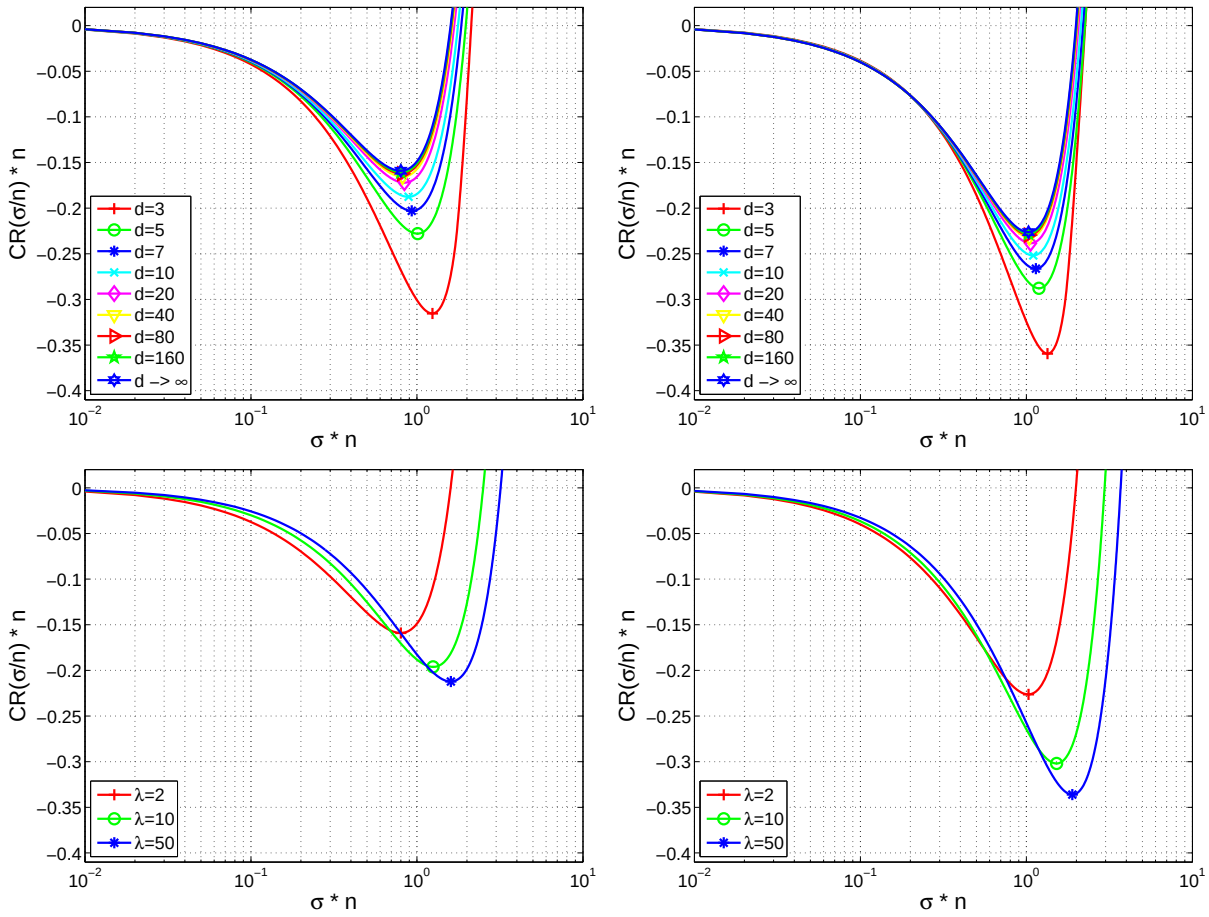

Figure 5.2: **Top**: Convergence rates $\mathrm{CR}_{\mathrm{EGS}}(\sigma)$ (left) and $\mathrm{CR}_{\mathrm{EGS-HS}}(\sigma)$ (right) for different dimensions n when the number of offspring λ is 2. **Bottom**: Theoretical limits of convergence rates $\mathrm{CR}_{\mathrm{EGS}}(\sigma)$ (left) and $\mathrm{CR}_{\mathrm{EGS-HS}}(\sigma)$ (right) for different λ when dimension n goes to infinity.

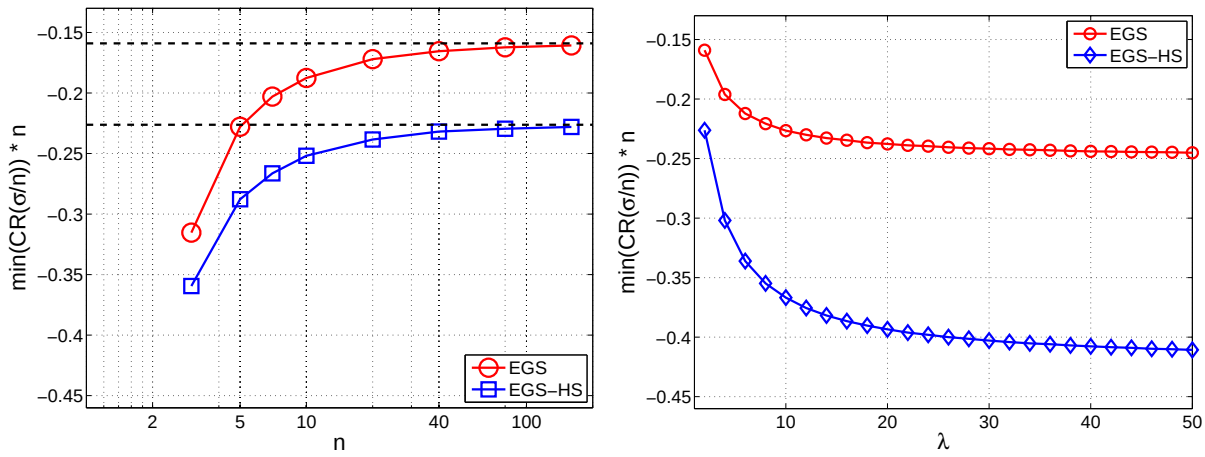

Figure 5.3: **Left**: The best observed convergence rates plotted against the dimensions when λ is 2. The dashed lines represent the theoretical limits when $n \to \infty$. **Right**: The best observed values of theoretical limits of convergence rates plotted against λ when $n \to \infty$.

5.5 Conclusion and Future Perspective

In this chapter, we have investigated in theory the log-linear convergence of the EGS with halfspace sampling. When halfspace sampling is used in the EGS, the whole search space is divided into two halfspaces with respect to a parent. A random sample is only used when it lies in the positive halfspace, otherwise its reflection with respect to the parent is used. All resulting random vectors in an iteration are used to estimate the optimal halfspaces in order to improve its performance in finding better solutions. We have proven the log-linear convergence of the scale-invariant step size EGS with and without halfspace sampling and we have expressed the convergence rates in terms of expectations of random variables. By means of the Monte-Carlo simulation, we have numerically computed the convergence rates and compared the lower bounds in finite and infinite dimensions, regardless of the number of offspring. The EGS with halfspace sampling always converges faster than the EGS without halfspace sampling. An improvement of 42% to 68% is observed

asymptotically. This gives rise to a promising future for halfspace sampling in the EGS in practice.

□ **End of chapter.**

Chapter 6

Eigenspace Sampling in Evolution Strategies

Research is creating new knowledge.

Neil Armstrong

6.1 Motivations

Evolution strategy (ES) is one of the popular evolutionary algorithms (EAs) used to solve many black-box optimization problems. More specifically, ES usually optimizes the real-valued objective functions $f : \mathbb{R}^n \to \mathbb{R}$ in the continuous domain. Candidate solutions, so-called individuals, are n-dimensional vectors of real numbers. The ES iteratively evolves the populations of individuals through variations and selection. The qualities of the individuals is determined by the fitness of the individuals, computed by the underlying objective functions. Selection chooses the best individuals in terms of their fitness and the selected individuals become the parents in the future generations. Variations occur through mutations and recombinations where mutations entail multivariate normal distributed random vectors to a recombinant generated by recombining the parents in the population. Multiple works have demonstrated the success of ESs in many domains including optimization, machine

learning and the real world applications.

The dimension of the search space n plays an important role in optimization. In particular, the volume of the continuous search space increases *exponentially* with n making it very difficult for search algorithms to optimize problems which are in high dimensions. In addition, many real parameter optimization problems have dependencies between the parameters, and learning these dependencies is demanded. Covariance matrix adaptation evolution strategy (CMA-ES) is one of the ES variants which is designed to learn the inter-dependencies between all the parameters by updating the covariance matrix for the sample distribution. The basic idea is to obtain the information about the successful search steps, and to use the information to update the covariance matrix of the mutation distribution in a derandomised mechanism. The covariance matrix is updated such that variances in the directions of the search space that have previously been successful are increased while those in other directions are decreased. Even for a small population, the accumulation of information over a number of successful steps can reliably adapt the covariance matrix. An experimental study [106] shows that the derandomised approach can outperform other variants of ES.

Originally designed for small population sizes, the CMA-ES efficiently minimizes unimodal functions and it is superior on ill-conditioned and non-separable problems to other evolutionary and estimation of distribution algorithms [121]. In [104], the CMA-ES is further extended by the so-called rank-μ-update. The rank-μ-update exploits the information contained in large populations more effectively without affecting the performance for small populations. Recent studies [102] showed a good performance of the CMA-ES combining large populations and rank-μ-update on the unimodal and multimodal functions without any additional tuning on parameter tuning.

The CMA-ES needs a large number of function evaluations to adapt the covariance matrix. The work [26] proposes to use the least squares approach to approximate the Hessian matrix[1] of the objective function using information obtained from a quadratic number of function evaluations. Inverting the Hessian can produce a matrix that can be used as the mutation covariance matrix. In some functions, inverting an approximation to the Hessian matrix produces good mutation covariance matrices very quickly. However, while it is beneficial in terms of the number of objective function evaluations required, the method requires an additional computational costs of order n^6 to solve the least squares problem.

This chapter proposes a modification to improve the adaptation of the covariance matrix in the standard CMA-ES. The update rule in the original algorithm is that the information stored in the covariance matrix decays at a constant rate, and the information from the successful steps of the algorithm is used to increase variances in directions of the search space that have proven beneficial in the past. Future offspring candidate solutions are therefore generated preferably in directions that have proven to be worth exploring. We propose to make a modification to the original CMA-ES such that the CMA-ES always biases to evaluate the directions with high variances while it randomly evaluates the directions with low variances. This is achieved by grouping the eigenvalues of the covariance matrix where the dominant eigenspaces are always evaluated and the minor eigenspaces are randomly evaluated. We experimentally study our proposed modification and investigate when our modification can show substantial benefits to the CMA-ES.

This chapter also introduces a new sampling method. Instead of sampling all the mirrored directions along the principal axes,

[1] A Hessian matrix is a square matrix of second-order partial derivatives of the objective function.

determined by the covariance matrix, we propose to cluster the eigenvalues of the covariance matrix of a CMA-ES and sample search points on a mirrored eigenspace spanned by eigenvectors that have the same repeated or clustered eigenvalues in the Hessian matrices of the objective functions. We apply this sampling method to a $(1, \lambda)$-CMA-ES and compare its performance with that of a $(1, \lambda^s_m)$-CMA-ES that uses the mirrored sampling and sequential selection method. Our simulations demonstrate promising results: the mirrored eigenspace sampling method is particularly pronounced on convex quadratic functions with eigenvalue spectra that are dominated by a large number of relatively large values.

6.2 Eigenspace of Search Space

Consider a convex quadratic objective function $f_{\mathbf{H}} : \mathbf{x} \mapsto \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}$, where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a positive definite matrix. If we use $\mathcal{N}(\mathbf{m}, \mathbf{C})$ to sample the mutation distribution, there is a close relationship between the Hessian matrix $\mathbf{H}$ and the covariance matrix $\mathbf{C}$: setting $\mathbf{C} = \mathbf{H}^{-1}$ on $f_{\mathbf{H}}$ is equivalent to optimizating the isotropic function $f_{\mathrm{Sphere}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{x}$. That implies that on the convex quadratic functions, setting the covariance matrix of the sample distribution to the inverse Hessian matrix is equivalent to rescaling an ellipsoid function to a sphere function. Therefore we assume the optimal covariance matrix is equal to the inverse Hessian matrix. Consequently, the objective of covariance matrix adaptation is to approximate the inverse Hessian matrix. An accurate and efficient covariance matrix adaptation is crucial to the performance of the CMA-ES, particularly when the underlying objective functions are ill-conditioned functions. By "ill-conditioned", we mean a large value of the condition number in the Hessian matrix of the objective function. Formally,

we can define the condition number as

$$\text{cond}(\mathbf{A}) = \frac{\xi_n}{\xi_1}$$

where ξ_i is the i-th eigenvalue of the positive definite matrix $\mathbf{A}$, such that $\xi_1 \leq \xi_2 \leq \cdots \leq \xi_{n-1} \leq \xi_n$. Given the convex-quadratic function $f_{\mathbf{H}}$, the condition number of the optimal covariance matrix is the same as the inverse Hessian matrix $\mathbf{H}^{-1}$.

Now consider if we have another convex-quadratic objective function $f_{\mathbf{H}}$ whose the number of distinct eigenvalues in the $n \times n$ Hessian matrix $\mathbf{H}$ is smaller than n. In other words, there exists a k number of *repeated eigenvalues* in the Hessian matrix. The optimal covariance matrix for this $f_{\mathbf{H}}$ also has repeated eigenvalues. The search space spanned by the same eigenvalues and the corresponding eigenvectors forms an eigenspace which is a scaled hypersphere. The convex-quadratic objective function $f_{\mathbf{H}}$ having k repeated eigenvalues in its Hessian matrix consists of k hyperspheres with different scales. When a covariance matrix adaptation attempts to estimate the covariance matrix during optimization, learning the intra-dependencies of parameters in each of these eigenspaces becomes less prominent because the same eigenvalues indicate that the dependencies between each dimension of an eigenspace become constant. Learning the inter-dependencies of the k eigenspaces with k distinct repeated eigenvalues becomes more demanding. Particularly, the largest repeated eigenvalue in the estimated covariance matrix which represents the eigenspace having the largest variances in the directions of the search space is more promising than the other $k-1$ eigenspaces. When the CMA-ES adapts the covariance matrix, it is intuitive to evaluate the dominant eigenspace with the largest repeated eigenvalue than the first-dominated eigenspace with the second largest repeated eigenvalue. Evaluating the first dominated eigenspace is more encouraging than evaluating the second-dominated eigenspace with the third largest repeated

eigenvalue. This makes the minor eigenspace (with the smallest repeated eigenvalue) become the most discouraging subspace. When the covariance matrix adaptation attempts to estimate the optimal covariance matrix during optimization, a dominance relationship is built among the eigenspaces

The concept of a dominance relationship among the eigenspaces can be extended to another kind of convex-quadratic function $f_{\mathbf{H}}$ where there exists an l number of *clustered eigenvalues* in the $n \times n$ Hessian matrix $\mathbf{H}$. Formally, there exists l number of clusters $\mathcal{C}^i$ in the eigenvalues of the Hessian matrix such that in each cluster $\mathcal{C}^i \equiv \left\{\xi_j : \forall j = 1, \ldots, |\mathcal{C}^i|\right\}$, the eigenspectrum of eigenvalues within each cluster has the condition of

$$\text{for } \epsilon \in \mathbb{R} \geq 0, \frac{\max\left(\mathcal{C}^i\right)}{\min\left(\mathcal{C}^i\right)} \leq (1+\epsilon)$$

while the spectrum between the clusters has the condition of

$$\text{for } \epsilon \in \mathbb{R} \geq 0, \forall i = 1, \ldots, l-1, \frac{\min\left(\mathcal{C}_i\right)}{\max\left(\mathcal{C}^{i+1}\right)} \geq (1+\epsilon)$$

where $|\mathcal{C}^i|$ is the number of eigenvalues in the cluster $\mathcal{C}^i$, $\max\left(\mathcal{C}^i\right)$ and $\min\left(\mathcal{C}^i\right)$ represent the maximum and the minimum eigenvalues in the i-th cluster $\mathcal{C}^i$ respectively. Hence the dominant cluster $\mathcal{C}^1$ consists of the set of eigenvalues having the first $|\mathcal{C}^1|$ largest eigenvalues of the Hessian matrix while the minor cluster $\mathcal{C}^l$ is the set of eigenvalues having the $|\mathcal{C}^l|$ smallest eigenvalues of the matrix. Notice that if we consider the eigenvalues in the Hessian matrix as one cluster, the condition number of the matrix is equivalent to its eigenspectrum for any given $\epsilon \in \mathbb{R} \geq 0$. By clustering the eigenvalues of the estimated covariance matrix during covariance matrix adaptation, one can have the preference on the dominant eigenspace spanned by the dominant eigenvalue cluster $\mathcal{C}^1$. The minor eigenspace consists of the set of eigenvalues which has the smallest variances in search directions and

should be less promising. Unlike the repeated eigenvalues, the space spanned by the clustered eigenvalues in a single cluster forms a hyper-ellipsoid in the search space when ϵ is greater than 0.

6.3 Implemention in (μ,λ)-CMA-ES

6.3.1 Algorithms

We outline our modification to the standard CMA-ES and name the resulting algorithm "ϵ-CMA-ES". The following steps replace the step 3 of the CMA-ES described in previous section.

a) Sort n number of square roots of eigenvalues d^i in the matrix $\mathbf{D}_k$ such that $d_k^1 \leq d_k^2 \leq \cdots < d_k^{n-1} \leq d_k^n$

b) Group the square roots of eigenvalues into n_e eigenvalue clusters such that each cluster $\mathcal{C}_k^i \equiv \left\{ d_k^j : \forall j = 1, \ldots, |\mathcal{C}_k^i| \right\}$ has the condition of

$$\text{for } \epsilon \in \mathbb{R} \geq 0, \frac{\max\left(\mathcal{C}_k^i\right)}{\min\left(\mathcal{C}_k^i\right)} \leq (1+\epsilon)$$

while the spectrum between the clusters has the condition of

$$\text{for } \epsilon \in \mathbb{R} \geq 0, \forall i = 1, \ldots, l-1, \frac{\min\left(\mathcal{C}_k^i\right)}{\max\left(\mathcal{C}_k^{i+1}\right)} \geq (1+\epsilon)$$

The notation $\mathcal{S}_i$ is used to represent the eigenspace spanned by the eigenvalues in the clusters $\mathcal{C}_k^i$. Notice that at any time during the optimization, the number of eigenvalue clusters is always bound by $1 \leq n_e \leq n$.

c) If the number of the eigenvalue clusters is between 1 and n, generate the mutation vectors $\mathbf{z}_{k+1}^i$ for the offspring i such that there is a probability of p_m for it to mutate

in each direction belonging to the minor eigenspace $\mathcal{S}_{n_e}$ while it *always* mutates in the directions belonging to other eigenspaces $\mathcal{S}_1, \ldots \mathcal{S}_{n_e-1}$. Given the mutation vectors $\mathbf{z}^i = \left[z^{i,1}, \ldots, z^{i,n}\right]$, if the j-th direction does not belong to the eigenspace $\mathcal{S}_{n_e}$, then $z^{i,j}$ is $\mathcal{N}(0,1)$. If the j-th direction belongs to the eigenspace $\mathcal{S}_{n_e}$, then mutate it in the j-th direction with a probability p_m.

c) If the number of the eigenvalue clusters is either 1 or n, generate the mutation vectors $\mathbf{z}^i_{k+1}$ for the offspring i in the same way as in the standard CMA-ES, i.e. $\mathbf{z}^i = [\mathcal{N}(0,1), \ldots, \mathcal{N}(0,1)]$.

Applying the above modification will change the way of generating the mutation vectors only. If the number of eigenvalue clusters is between the 1 and n, the ϵ-CMA-ES will randomly evaluate the minor eigenspace with a probability of p_m in each direction of the minor eigenspace. Setting $p_m = 1$ will restore the algorithm back to the standard CMA-ES since all directions in the eigenspaces have been evaluated.

6.3.2 Experimental Study

6.3.3 Setup

In our experiments, we studied the performance of the ϵ-CMA-ES and compared its performance with that of the standard CMA-ES. Table 6.1 lists the test functions in the comparisons. The test functions are commonly used in the related work [106, 104]. All the functions are scalable in the problem dimensions and have a minimum function value of 0. The global minimum is located at $\mathbf{x} = \mathbf{0}$, except for the Rosenbrock function, which has an optimum at $x_i = 1$ for all i. The number of function evaluations required to reach an objective function value of f_{stop} is used as a performance measure, and the respective values of f_{stop} are also given in the table. To avoid an easy exploitation

of the symmetry, we use non-symmetrical initialization intervals which are also shown in Table 6.1.

Table 6.1: Test Functions to be minimized, the stopping criteria and the initialization region

Name	Function	f_{stop}	Initialization Region
Sphere	$f_{\text{sphere}}(\mathbf{y}) = \sum_{i=1}^{n} y_i^2$	10^{-10}	$[1,5]^n$
Ellipsoid	$f_{\text{ellipsoid}}(\mathbf{y}) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} y_i^2$	10^{-10}	$[1,5]^n$
Cigar	$f_{\text{cigar}}(\mathbf{y}) = y_1^2 + \sum_{i=2}^{n} \alpha \cdot y_i^2$	10^{-10}	$[1,5]^n$
Tablet	$f_{\text{tablet}}(\mathbf{y}) = \alpha \cdot y_1^2 + \sum_{i=2}^{n} y_i^2$	10^{-10}	$[1,5]^n$
Cigtab	$f_{\text{cigtab}}(\mathbf{y}) = \alpha \cdot y_1^2 + \sum_{i=2}^{n-1} \alpha^{\frac{1}{2}} \cdot y_i^2 + y_n^2$	10^{-10}	$[1,5]^n$
Twoaxes	$f_{\text{twoaxes}}(\mathbf{y}) = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \alpha \cdot y_i^2 + \sum_{i=\lfloor \frac{n}{2}+1 \rfloor}^{n} y_i^2$	10^{-10}	$[1,5]^n$
Rosenbrock	$f_{\text{rosenbrock}} = \sum_{i=1}^{n-1} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	10^{-10}	$[1,5]^n$

The performance of the CMA-ESs are tested for problem dimensions $n \in \{2, 3, 5, 10, 20, 40, 80\}$. The parameter ϵ is set to 0.2 and 4. The probability p_m is set to $\frac{1}{2}$. All runs are performed with the default parameters in the CMA-ES. Except for the default population size λ, we performed additional experiments for $\lambda = 4n$. 50 runs are conducted for each setting. The starting point $\mathbf{x}_0$ is sampled uniformly with the initialization intervals given in Table 6.1. The initial step size of σ_0 is set to half of the initialization intervals. Additionally, the run is stopped after 10^7 function evaluations. The scaling coefficient α in the definitions of $f_{\text{ellipsoid}}$, f_{cigar}, f_{tablet}, f_{cigtab} and f_{twoaxes} is 10^6 unless when noted otherwise. Lastly, except the Rosenbrock function, all experiments are performed on the objective variables $\mathbf{y}$ obtained by multiplying $\mathbf{x}$ with a random orthonormal base, making all functions except f_{sphere} non-separable.

6.3.4 Results

The average numbers of function evaluations to reach the f_{stop} versus the problem dimensions in each test function are reported in the Figure 6.4 and Figure 6.1. Typical single runs of the standard CMA-ES and the ϵ-CMA-ES on the tested functions with problem dimension $n = 10$ are shown in Figure 6.5 and Figure 6.2. Each graph shows the objective function values $f(\mathbf{x})$, and the squared global step size σ^2 and the sorted principal axis length of the mutation distribution (*i.e.* the d^i of the matrix $\mathbf{D}_k$) plotted against the number of function evaluations. Notice that each d^i is scaled for a better readability.

We now analyze the test functions one by one. Firstly, on f_{sphere} function, we can observe that the standard CMA-ES is almost identical to the ϵ-CMA-ES for $\epsilon \in \{0.2, 4\}$. When the standard CMA-ES adapts the covariance matrix, the eigenspectrum of square roots of eigenvalues d^i in the matrix $\mathbf{D}$ varies

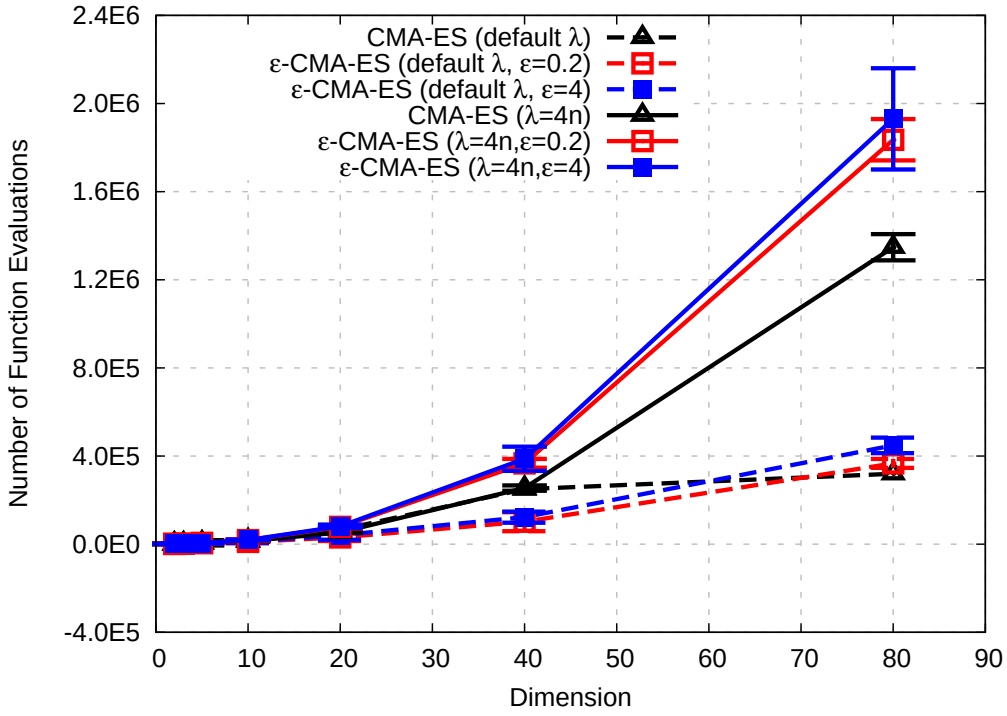

Figure 6.1: The average numbers of function evaluations to reach f_{stop} in all the successful runs over the 50 trials, versus the problem dimensions n for the standard CMA-ES (the black lines), our ϵ-CMA-ES (the red lines and the blue lines represent the performance when ϵ are 0.2 and 4 respectively), on Rosenbrock function. Dotted lines and solid lines represent the CMA-ES using default $\lambda = 4 + \lfloor 3 \cdot ln(n) \rfloor$ and $\lambda = 4n$ respectively. Notice that each d^i is scaled up for a better readability.

very slightly because the underlying Hessian matrix of the objective function is $\mathbf{H} = \mathbf{I}$. The condition number of this matrix is always 1 making the ideal number of eigenvalue cluster also 1. Setting $\epsilon \in \{0.2, 4\}$ makes the estimated number of eigenvalue cluster n_e always 1, except for the higher dimension when n is 80 and ϵ is 0.2. The fluctuation of the estimated square roots of eigenvalues d^i becomes greater making a larger value of estimated number of eigenvalue cluster.

The eigenvalues of the $f_{\text{ellipsoid}}$ function are evenly distributed over the eigenspectrum of the Hessian matrix. Therefore, it is considered that there is no apparent eigenvalue clusters in the

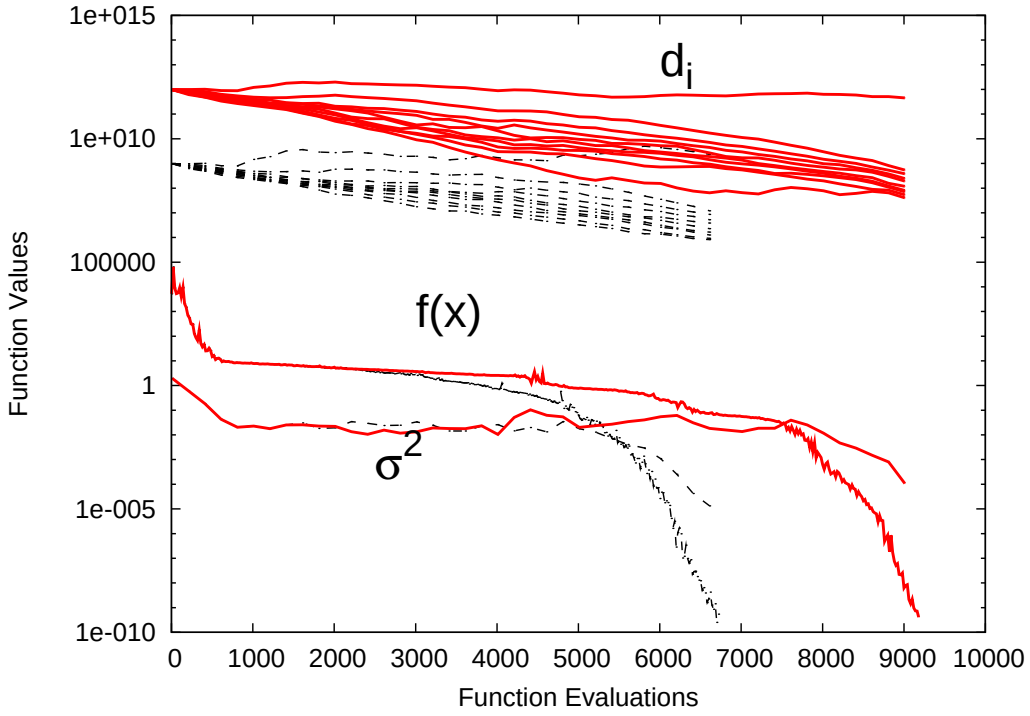

Figure 6.2: A single run of the ϵ-CMA-ES (the thick red solid lines) and the standard CMA-ES (the thin black dotted lines) on Rosenbrock functions when ϵ is 4. The lines shown are the function values $f(\mathbf{x})$, the global step size σ and the square roots of eigenvalues of the covariance matrix of mutation distribution d_i when the problem dimension n is 10.

$f_{\text{ellipsoid}}$ function and the optimal number of eigenvalue cluster n_e is 1. If we set a small ϵ, *e.g.* 0.2, the minor eigenspace is usually spanned by the smallest square roots of eigenvalues d_1. This can cause a slight improvement when the minor eigenspace is in 1-dimension or in 2-dimension. There is around 10% of decrease in the number function evaluations when n is 10 and 20. However, when ϵ is 5, the minor eigenspace may be a subspace in a higher dimension. Since the ϵ-CMA-ES will randomly evaluate the directions in the minor eigenspace, it will degrade the progress of the covariance matrix adaptation. More function evaluations are required in such cases. An example of these cases can be seen when the standard CMA-ES outperforms the

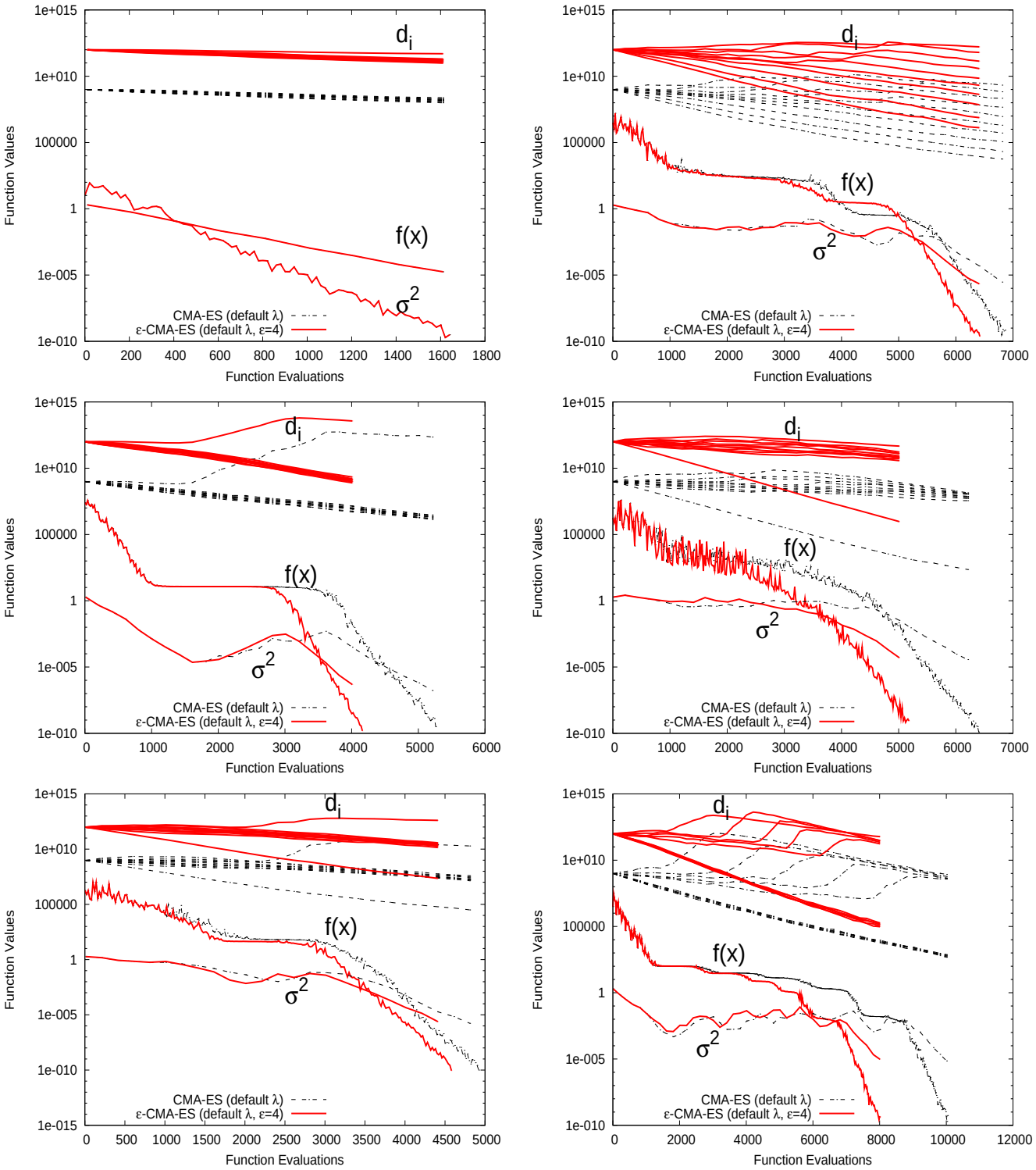

Figure 6.3: A single run of the ϵ-CMA-ES (the thick red solid lines) and the standard CMA-ES (the thin black dotted lines) on various functions when ϵ is 4 and the problem dimension n is 10. From top to bottom, left to right, the functions are Sphere, Ellipsoid, Cigar, Tablet, Cigtab, and Twoaxes functions. Notice that each d^i is scaled up for a better readability.

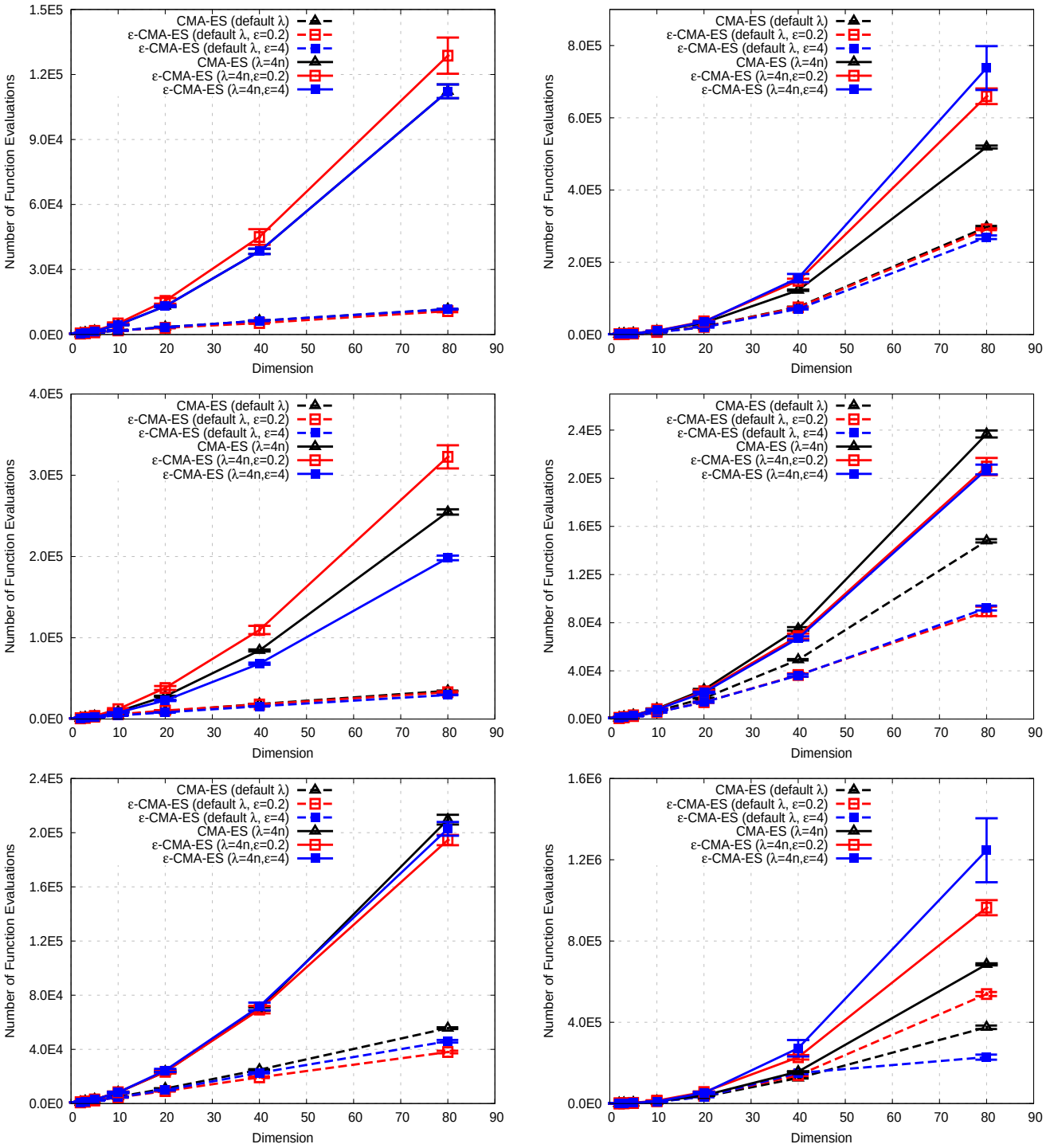

Figure 6.4: The average numbers of function evaluations to reach f_{stop} in all the successful runs over the 50 trials, versus the problem dimensions n for the standard CMA-ES (the black lines), our ϵ-CMA-ES (the red lines and the blue lines represent the performance when ϵ are 0.2 and 4 respectively). From top to bottom, left to right, the functions are Sphere, Ellipsoid, Cigar, Tablet, Cigtab, and Twoaxes functions. Dotted lines and sold lines represent the CMA-ES using default $\lambda = 4 + \lfloor 3 \cdot ln(n) \rfloor$ and $\lambda = 4n$ respectively.

ϵ-CMA-ES for $n \in \{40, 80\}$.

The f_{cigar}, f_{tablet} and f_{twoaxes} functions are similar in that there are two apparent eigenvalue clusters and their minor eigenspaces are spanned by the subspace in $n-1$ dimension, 1 dimension and $\frac{n}{2}$ dimension respectively. It can seen that all the ϵ-CMA-ESs outperform the standard CMA-ES except for f_{twoaxes} when n is 80 and λ is $4n$. The best ϵ-CMA-ES requires about 23%, 38% and 40% fewer function evaluations to reach f_{stop}. Overall, the performance advantage of the ϵ-CMA-ES can be observed across the range of search space dimensionality when it is used to optimize the test functions which have two eigenvalue clusters in their Hessian matrix of the objective functions. The Hessian matrix of the f_{cigtab} has three eigenvalue clusters. The ϵ-CMA-ES also outperforms the standard CMA-ES in f_{cigtab} function. A reduction of around 8% in the number of function evaluations is observed. Finally, it can seen from the figures that the performance of the ϵ-CMA-ES does not outperform the standard CMA-ES on the $f_{\text{rosenbrock}}$ function which is not a convex-quadratic objective function. This is the only test function in this chapter where ϵ-CMA-ES requires more function evaluations across the whole range of search space dimensionality. We believe that the poor performance is due to the evenly distributed eigenspectrum of eigenvalues in the Rosenbrock function which is similar to that in the Ellipsoid function $f_{\text{ellipsoid}}$.

6.4 Implemention in mirrored variant of $(1, \lambda)$-CMA-ES

6.4.1 Mirroring sampling and Sequential selection

Recently, mirroring sampling [53, 21] has been proposed to replace the independent sampling in evolution strategies by de-

pendent ones in order to increase the probability of successful sampling possibly resulting in an increase in the convergence speed of the algorithms. Instead of sampling all λ offspring i.i.d., the mirrored variant of evolution strategies samples only $\lceil \lambda/2 \rceil$ i.i.d. offspring as $\mathbf{x}_k^i = \mathbf{m}_k + \sigma_k \times \boldsymbol{\mathcal{N}}_i(\mathbf{0}, \mathbf{C}_k)$ for $1 \leq i \leq \lceil \lambda/2 \rceil$, where $\boldsymbol{\mathcal{N}}_i(\mathbf{0}, \mathbf{C}_k)$ is the realization of the mutation step for i-th offspring. Up to $\lfloor \lambda/2 \rfloor$ and further, the offspring depend on the already drawn samples as $\mathbf{x}_k^i = \mathbf{m}_k - \sigma_k \times \boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k)$ for $\lceil \lambda/2 \rceil + 1 \leq i \leq \lambda$. They are thus mirrored or symmetric with respect to the parent $\mathbf{m}_k$. Consequently, a mirrored sample is used if and only if the iteration index g is even. We introduce a new notation. The number of the the independent offspring per iteration is denoted by λ_{iid} and the number of the mirrored offspring per iteration is denoted by λ_m, where $\lambda = \lambda_{iid} + \lambda_m$ solutions are evaluated in each iteration. As a result, if we set $\lambda_m = 0$, it results in the standard $(1, \lambda)$-CMA-ES. We denote the new algorithm as the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES.

Evaluating a sampled solution and its mirrored counterpart can result in unnecessary function evaluations. On unimodal objective functions with convex sub-level sets, the mirrored solution $\mathbf{m}_k - \sigma_k \times \boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k)$ for $\lceil \lambda/2 \rceil + 1 \leq i \leq \lambda$ must be worse than the parent $\mathbf{m}_k$, if $\mathbf{m}_k + \sigma_k \times \boldsymbol{\mathcal{N}}_i(\mathbf{0}, \mathbf{C}_k)$ for $1 \leq i \leq \lceil \lambda/2 \rceil$ was better than $\mathbf{m}_k$. Sequential selection [21], originally introduced to save these unnecessary function evaluations, can be independent of mirrored sampling: the offspring are evaluated one by one, compared with their parent, and the iteration is concluded immediately if one offspring is better than its parent. If all the λ offspring are worse than the parent, the original selection scheme is applied. When sequential selection is implemented in the $(1, \lambda)$-CMA-ES, the offspring are generated with mirrored sampling. Evaluations are carried out in a sequential manner. After evaluating $\mathbf{x}_k^1$, it is compared to $\mathbf{m}_k$ and if $f(\mathbf{x}_k^1) \leq f(\mathbf{m}_k)$, the sequence of evaluations

is stopped and the offspring will replace the parent immediately for the next iteration, i.e. $\mathbf{m}_{k+1} = \mathbf{x}_k^1$. Sequential selection will proceed until all $\lambda_{iid} + \lambda_m$ offspring are generated and evaluated. In case all offspring are worse than $\mathbf{m}_k$, the best individual will become the parent in the next iteration, i.e. $\mathbf{m}_{k+1} = \text{argmin}\left\{f(\mathbf{x}_k^1), \ldots, f(\mathbf{x}_k^\lambda)\right\}$ according to the comma selection. The number of offspring evaluated is a random variable ranging from 1 to $\lambda_{iid} + \lambda_m$ that can reduce the number of offspring adaptively as long as an improvement is achieved.

6.4.2 Algorithms

As the concepts of mirrored sampling and eigenvalue clustering are independent, they can be applied simultaneously. We consider a new variant of the $(1, \lambda)$-CMA-ES that differs in the choice of mirrored offspring in the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES. Formally, in each iteration step, we first cluster the eigenvalue in the covariance matrix as follow:

a) Sort n number of eigenvalues in the matrix $\mathbf{D}_k$ such that $d_k^1 \leq d_k^2 \leq \cdots \leq d_k^{n-1} \leq d_k^n$

b) Cluster the eigenvalues into n_e eigenvalue clusters such that each cluster $\mathcal{C}_k^i \equiv \left\{d_k^j : \forall j = 1, \ldots, |\mathcal{C}_k^i|\right\}$ has the condition of:

$$\text{for } \epsilon \in \mathbb{R} \geq 0, \frac{\max\left(\mathcal{C}_k^i\right)}{\min\left(\mathcal{C}_k^i\right)} \leq (1 + \epsilon)$$

while the spectrum between two neighbouring clusters has the condition of
for $\epsilon \in \mathbb{R} \geq 0$,

$$\forall i = 1, \ldots, l, \frac{\min\left(\mathcal{C}_k^i\right)}{\max\left(\mathcal{C}_k^{i-1}\right)} > (1 + \epsilon)$$

We use the notation $\mathcal{S}_i$ to represent the eigenspace spanned by eigenvectors associated with the eigenvalues in the clusters $\mathcal{C}_k^i$. Notice that at any time during the optimization, the number of eigenvalue clusters is always bound by $1 \leq n_e \leq n$.

Instead of sampling all the eigenspace (which is what a traditional mirroring offspring does), we sample the search points on an eigenspace $\mathcal{S}_u$ where $u \sim \mathcal{U}(1, n_e)$ is an integer uniformly generated from 1 to n_e. Formally, the original equations of offspring generation in Algorithm 3 are changed to:
for $\lceil \lambda/2 \rceil + 1 \leq i \leq \lambda$,

$$\mathbf{x}_i \sim \mathbf{m}_k - \sigma_k \times \boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k^{S_u})$$

where $\mathbf{C}_{S_u} = \mathbf{B}\mathbf{D}_{S_u}\mathbf{B}^T$ and

$$\mathbf{D}_{S_u} = \begin{bmatrix} 0 & 0 & \dots. & 0 \\ 0 & d_k^1 & \dots. & 0 \\ \vdots & 0 & \ddots & 0 \\ \vdots & \vdots & d_k^{|\mathcal{C}_k^i|} & 0 \\ 0 & 0 & \dots. & 0 \end{bmatrix}$$

where the set of eigenvalues $d_k^1, \dots, d_k^{|\mathcal{C}_k^i|} \in \mathcal{C}_k^i$ are the eigenvalues in the kth-eigencluster $\mathcal{C}_k^i$. We now introduce another notation. The number of offspring sampled by the above eigencluster per iteration is denoted by λ_g, where in each iteration $\lambda = \lambda_{iid} + \lambda_g$ solutions are evaluated. Similar to the mirrored variant in the previous subsection, when $0 \leq \lambda_g \leq \lambda_{iid}$ it results in the standard $(1, \lambda)$-CMA-ES in case $\lambda_m = 0$. We denote this new algorithm as the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES. In short, the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES executes the same update equations in Algorithm 3 except only when it generates the λ_g number of offspring. It first samples a candidate solution $\mathbf{m}_k + \sigma_k \times \boldsymbol{\mathcal{N}}_i(\mathbf{0}, \mathbf{C}_k)$. If this solution is better than the parent $\mathbf{m}_k$, it will replace the parent and

CMA-ES will go to the next iteration. If this solution is worse than the parent $\mathbf{m}_k$, the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES will cluster the eigenvalues in the covariance matrix. It then generates a random integer $u \sim \mathcal{U}(1, n_e)$ to select an eigenspace uniformly and samples a new candidate solution $\mathbf{x}_i \sim \mathbf{m}_k - \sigma_k \times \boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k^{S_u})$. Notice that this candidate solution is similar to the mirrored offspring except that the realization of $\boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k^{S_u})$ has the same dimensionality of the eigenspace $\mathcal{S}_u$. If this solution is better than the parent $\mathbf{m}_k$, replace the parent. Otherwise, it will generate a new offspring with a new realization of $\boldsymbol{\mathcal{N}}_{i+1}(\mathbf{0}, \mathbf{C}_k)$. Sequential selection will proceed until all λ_{iid} offspring are generated.

Lastly we consider one variant of the $(1, \lambda)$-CMA-ES that does not use any eigenvalue clustering to sample an eigenspace. This variant always samples a direction along a principal axis of the covariance matrix that is selected uniformly. Formally, the update equations of offspring generation becomes:
for $\lceil \lambda/2 \rceil + 1 \leq i \leq \lambda$,

$$\mathbf{x}_i \sim \mathbf{m}_k - \sigma_k \times \boldsymbol{\mathcal{N}}_{i-\lceil \lambda/2 \rceil}(\mathbf{0}, \mathbf{C}_k^i)$$

where $\mathbf{C}_k = \mathbf{B}\mathbf{D}_k\mathbf{B}^T$ and

$$\mathbf{D}_k = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & d_k^u & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

where $u \sim \mathcal{U}(1, n)$ is an integer uniformly generated from 1 to n and d_k^u is the k-th eigenvalue associated with the k-principal axis of the covariance matrix $\mathbf{C}_k$. We refer to this variant as the $(1, \lambda_{iid} + \lambda_u)$-CMA-ES.

6.4.3 Parameters

The update of evolution path of $\mathbf{p}^\sigma$ and $\mathbf{p}^c$ in a $(1, \lambda)$-CMA-ES is an exponential smoothing. The information stored in the covariance matrix decays at a constant rate. The information from the successful steps of the algorithm is used to increase variances in the directions of the search space that have proven beneficial in the past. The updates of evolution paths assume that the search points are fully sampling in all the directions along the principal axes. Since the two strategies $(1, \lambda_{iid} + \lambda_g)$-CMA-ES and $(1, \lambda_{iid} + \lambda_u)$-CMA-ES sample on a mirrored eigenspace and a 1-dimensional space along a principal axis, this will result in a bias towards small step sizes. Most importantly, the eigenspaces that are not sampled by these two variants will shrink more rapidly. The eigenvalues of these eigenspace will decrease much faster than those in the sampled eigenspace. Hence we consider to changing the learning parameters c_σ, c_c of the two evolution paths $\mathbf{p}^\sigma$ and $\mathbf{p}^c$, and those learning parameters c_1, c_μ for covariance matrix update from scalars to n-dimensional vectors. When a CMA-ES samples search points on an eigenspace S_u, only those dimensions in the sampled eigenspace are updated by the learning parameters. Other eigenspace that are not sampled will not be updated. Formally, the update equations in Algorithm 3 are changed to :

$$\mathbf{p}^\sigma_{k+1} = (\mathbf{1} - \vec{c}_\sigma) \cdot \mathbf{p}^\sigma_k + \sqrt{\vec{c}_\sigma(2 - \vec{c}_\sigma)\mu_w} \left(\mathbf{C}_k{}^{-\frac{1}{2}} \frac{\mathbf{m}_{k+1} - \mathbf{m}_k}{\sigma_k} \right)$$

$$\mathbf{p}^c_{k+1} = (\mathbf{1} - \vec{c}_c) \cdot \mathbf{p}^c_k + h_\sigma \sqrt{\vec{c}_c(2 - \vec{c}_c)\mu_w} \frac{\mathbf{m}_{k+1} - \mathbf{m}_k}{\sigma_k}$$

$$\begin{aligned} \mathbf{C}_{k+1} =& (\mathbf{1} - \vec{c}_1 - \vec{c}_\mu)\mathbf{C}_k(\mathbf{1} - \vec{c}_1 - \vec{c}_\mu)^T \\ &+ (\vec{c}_1 \cdot \mathbf{p}^c_{k+1})(\mathbf{p}^c_{k+1} \cdot \vec{c}_1)^T + \vec{c}_\mu \mathbf{C}^\mu \vec{c}_\mu{}^T \end{aligned}$$

where $\vec{c}_\sigma = c_\sigma \vec{a}$, $\vec{c}_1 = c_1 \vec{a}$, $\vec{c}_c = c_c \vec{a}$, $\vec{c}_\mu = c_\mu \vec{a}$ for $\vec{a} = (\mathbf{e}_1^{\mathbf{S}_k} + \ldots + \mathbf{e}_{|\mathbf{S}_k|}^{\mathbf{S}_k})$ and $\mathbf{e}_i^{\mathbf{S}_k}$ is the i-orthonormal basis spanned in the kth-eigenspace.

6.4.4 Experimental Study

In order to evaluate the eigenspace sampling in CMA-ES, we compared it with corresponding algorithms that do not use eigenspace sampling. The following subsection describes the results observed on a set of well-understood and convex-quadratic functions that are frequently used to evaluate the performance of real-valued evolutionary algorithms.

6.4.5 Performance on the Convex-Quadratic Functions

The six functions in the Table 6.1 are all convex-quadratic and have been employed extensively in the literature [106, 104]. All the functions are scalable in the problem dimensions and have a minimum function value of 0. The global minimum is located at $\mathbf{x} = \mathbf{0}$. In all cases, the scaling factor α is set to 10^8, resulting in mostly ill-conditioned problems. This scaling factor basically determines the eigenvalue spectra of the Hessian matrices of the functions. The larger the scaling factor, the larger the eigenvalue spectra of the Hessian matrices. Intuitively, CMA-ES requires more function evaluations to learn the appropriate covariance matrix for highly conditioned problems. We believe that the eigenvalue distributions of the Hessian matrices will affect the performance of a CMA-ES, particularly what the optimal updates on the covariance matrices should be when the eigenvalue distribution has an arbitrary number of eigenclusters which have different eigengaps between eigenclusters and in each eigencluster. Notice that while all functions are separable, this is not a limitation as CMA-ES is invariant with regard to the transformations of the coordinate system. Applying random rotations

would result in non-separable functions without affecting the performance.

All runs are initialized with the parental candidate solution $\mathbf{m}_0$ drawn from a n-dimensional uniform distribution over the search space. The initial global step size σ is set to 0.5, the covariance matrix is set to the identity matrix, and the search path is set to the zero vector. The eigenvalue clustering parameter ϵ is set to 1. All runs are terminated when (1) a candidate solution with an objective function value of $f_{stop} = 10^{-10}$ or better has been generated, or (2) the number of function evaluations exceeds $10^3 \cdot n^2$.

The behavior of CMA-ES on convex-quadratic functions is well studied. The optimal covariance matrix is equal to the inverse Hessian matrix and it should be constant throughout the search space. A CMA-ES basically learns a covariance matrix close to the optimal one and then proceeds as fast as they can to optimize the sphere function. Given the small f_{stop} in the simulations, it is largely the amount of time required to learn an approximation to the inverse Hessian that determines the number of function evaluations required to satisfy the termination condition for any dimensional problems.

For each function, we simulate 4 variants of the $(1, \lambda)$-CMA-ES:

1. The mirrored variant of the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES using the mirrored eigenspace sampling method. Both λ_{iid} and λ_g are set to 1.

2. The mirrored variant of the $(1, \lambda_{iid}+\lambda_u)$-CMA-ES using the mirrored sampling along a principal axis. Both λ_{iid} and λ_g are set to 1.

3. The mirrored variant of the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES in [21, 53]. Both λ_{iid} and λ_m are set to 1.

4. The original $(1, \lambda_{iid})$-CMA-ES in [106, 104]. λ_{iid} is set to 2.

Figure 6.5 illustrates the behavior of the $(1, 2)$-CMA-ES in the typical runs on the cigar function with $n = 10$. Shown are the objective function values of the search points, the global step size σ, and the eigenvalues of the covariance matrix $\mathbf{C}$ that is multiplied with 10^4 for clarity. The cigar function is characterized by one eigenvalue of its Hessian matrix being significantly smaller than the remaining eigenvalues. In this case, the optimal covariance matrix therefore has one eigenvalue significantly larger than the others. The axis scales in Figure 6.5 suggest that a covariance matrix close to the optimal one is achieved toward the end of the runs as there is a dominant eigenvalue while the remaining nine minor eigenvalues are smaller by a factor of about 10^8 (about the same value of α). Comparing the two subfigures, using the eigenspace sampling in CMA-ES allows a faster reduction of the magnitude of multiple eigenvalues than the mechanism in the original strategy does. Using the eigenspace sampling in this particular case, the improvement resulting from is almost one third. Due to the nature of the eigenvalue spectrum, the cigar function is a function for which using the eigenspace sampling can be expected to be beneficial.

Figure 6.6 shows the median numbers of function evaluations required to reach f_{stop} for all convex-quadratic test functions. The dimensions of the search space are from $n =$ to $n = 40$. First, we analyze the performance on the sphere function. Comparing all mirrored variants to the original CMA-ES, the median speed-up is from 42% to 56% from $n = 3$ to $n = 40$. The medians of the $(1, 2_g)$-CMA-ES and the $(1, 2_m)$-CMA-ES are almost identical. The $(1, 2_u)$-CMA-ES has a smaller median number. Overall, the median speed up of the mirrored variants is expected. Since the Hessian matrix of the sphere function is an identify matrix, the optimal number of eigenclusters is therefore 1. The eigenvalue spectrum d^i in the matrix $\mathbf{D}$ can vary slightly

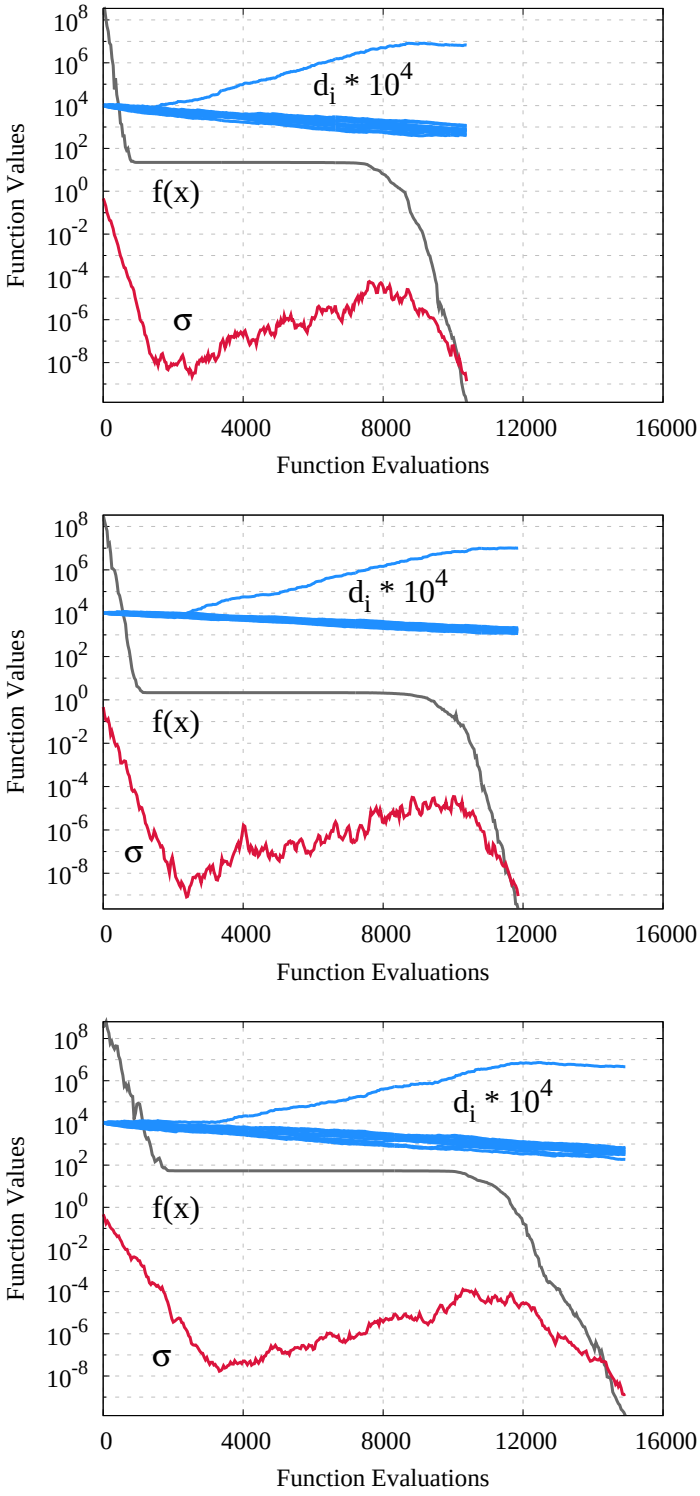

Figure 6.5: Typical runs of the $(1, 2_g)$-CMA-ES using the mirrored eigenspace sampling method (top), the $(1, 2_u)$-CMA-ES using the mirrored sampling along a principal axis. (middle), and the standard $(1, 2)$-CMA-ES (bottom) on the cigar functions. The lines shown are the function values $f(\mathbf{x})$, the global step size σ and the eigenvalues of the covariance matrix of mutation distribution d_i when the problem dimension n is 10.

in the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES. In most cases, a realization of a sample on an eigenspace is basically a sample on a n-dimensional space. Sampling on the n-dimensional eigenspace is in fact identical to mirrored sampling in the $(1, 2_m)$-CMA-ES. This explains why the median numbers of the $(1, 2_g)$-CMA-ES and the $(1, 2_m)$-CMA-ES are close. On the other hand, sampling in the $(1, 2_u)$-CMA-ES is always on the 1-dimensional space. As the sphere function is invariant, sampling on a 1-dimensional space along a principal axis would always result in a better solution than the parental candidate solution.

On ellipsoid function, the median speed up of the $(1, 2_g)$-CMA-ES is from 18% to 51% for $n = 5$ to $n = 40$. The worst case was when $n = 2$ with a loss in performance about 69%. The $(1, 2_u)$-CMA-ES is not perform as the $(1, 2_g)$-CMA-ES does. In moderate values of n, the median speed up is about 25%. The performance starts to deteriorate when the problem dimension is large. The $(1, 2_m)$-CMA-ES performs consistently from $n = 5$ to $n = 40$. Its median speed up is up to 60%. In fact, the eigenvalue spectrum of the Hessian matrix in the ellipsoid function is evenly distributed. As there is no dominant eigenvalue, it is therefore right to deduce that there are no eigenclusters. Hence, the optimal number of eigenclusters n_e is 1. As the $(1, \lambda_{iid} + \lambda_u)$-CMA-ES samples on 1-dimensional space, the learning of covariance matrix becomes relatively slow, compared with the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES and the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES that both sample the kth-dimensional eigenspace and all the n-dimensional search space respectively. It is thus suggested to have a large eigenvalue clustering factor ϵ which can result in a decrease in the number of eigenclusters formed. Setting ϵ to ∞ will force the $(1, \lambda_{iid} + \lambda_g)$-CMA-ES to always form a single eigencluster, making it actually behave like the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES, i.e. sampling all directions of the principal axes.

The cigar, tablet and twoaxes functions are similar in that

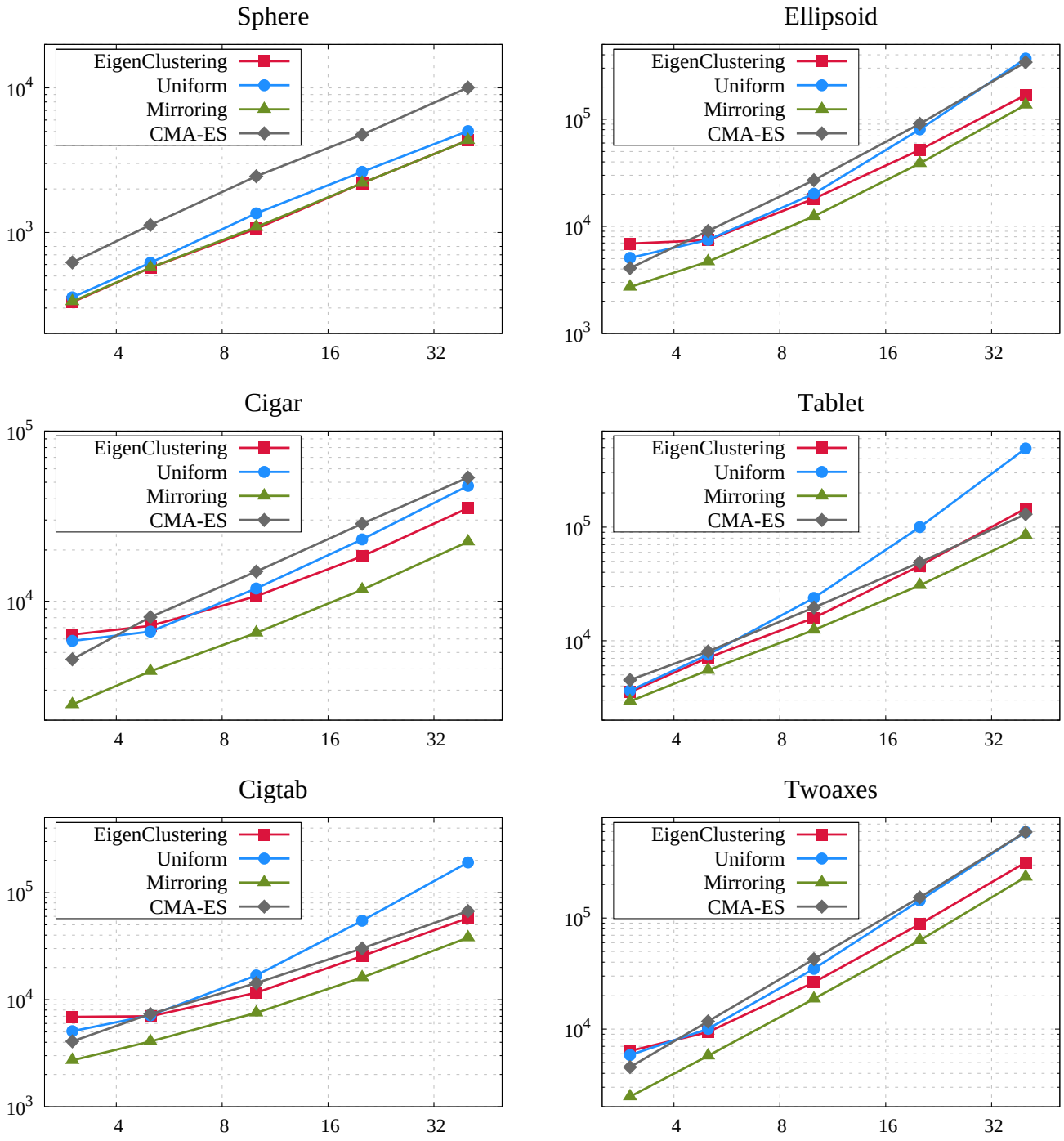

Figure 6.6: Log-log plot of the median number of function evaluations (over the successful trials out of 21) required to reach the target fitness value of 10^{-10} for 6 different benchmark functions on dimensions 3 to 40. For each graph, the y-axis represents the median numbers of function evaluations and the x-axis represents the problem dimensions. The legends "Eigenclustering", "Uniform", "Mirroring" and "CMA-ES" represent the median numbers of the $(1, 2_g)$-CMA-ES, $(1, 2_u)$-CMA-ES, $(1, 2_m)$-CMA-ES and the original $(1, 2)$-CMA-ES respectively.

there are two apparent eigenclusters in their Hessian matrices, but they are different in their eigenvalue spectra. In the cigar function, the eigenvalue spectrum is a relatively small eigenvalue dominated by all the remaining large eigenvalues. The median speed up of the $(1, 2_g)$-CMA-ES is from -40% up to 40% for $n = 3$ to $n = 40$. It is relatively higher than that the $(1, 2_u)$-CMA-ES. The best speed up is about 20%, lower than that of the $(1, 2_m)$-CMA-ES, which a median gain of 59%. Similarly, we observe the same behavior among all mirrored variants on the twoaxes function. The $(1, 2_m)$-CMA-ES achieved the best median speed up, followed by the $(1, 2_g)$-CMA-ES and the $(1, 2_u)$-CMA-ES respectively. On the other hand, observations are different in the tablet function. On average, the median speed up in the $(1, 2_m)$-CMA-ES is about 35%. Both the $(1, 2_g)$-CMA-ES and the $(1, 2_m)$-CMA-ES appear to underperform with increasing n. In no instance do the savings in the median number of function evaluations exceed 22%. We believe that the advantage resulting from the use of eigenvalue clustering to sample on an eigenspace is related to the ratio of small eigenvalues to large eigenvalues. We plan to investigate the correlations between these two. The $(1, 2_u)$-CMA-ES is relatively outperformed by the other two mirrored variants as sampling on a 1-dimensional space spanned by one eigenvector is not beneficial.

The last convex quadratic function is the cigtab function, which has the characteristics of both cigar and tablet functions. Its eigenvalue spectrum for the Hessian matrix has a relatively large eigenvalue and a relatively small eigenvalue, all the $n - 2$ remaining eigenvalues are between the two. The $(1, 2_m)$-CMA-ES has the best median speed up at 47%. The median speed up of the $(1, 2_g)$-CMA-ES is up to 19%. The $(1, 2_u)$-CMA-ES underperforms relative to the original $(1, 2)$-CMA-ES. The worst results occur happens when $n = 40$ with the loss in performance down to -184%.

6.5 Future Perspective

In the first part of this chapter, we proposed a modification to the standard covariance matrix adaptation evolution strategies (CMA-ES) algorithm called ϵ-CMA-ES. The objective of the modification is to improve the performance of the standard CMA-ES by reducing the number of function evaluations needed for covariance matrix adaptation. The covariance matrix adaptation can be improved if the ϵ-CMA-ES can identify the minor eigenspace in the Hessian matrix of the underlying objective functions, which have repeated eigenvalues or clustered eigenvalues. While the ϵ-CMA-ES always evaluates all the directions of the dominant eigenspace and the dominated eigenspaces, the ϵ-CMA-ES will randomly evaluate each direction in the minor eigenspace. In this thesis, the ϵ-CMA-ES has been investigated on a set of common unimodal benchmark problems, including ill-conditioned functions and functions that have a few repeated eigenvalues in their Hessian matrices. The advantages of the ϵ-CMA-ES are most pronounced on objective functions with a minor eigenspace that is dominated by one or two eigenspaces, such as the Cigar function f_{cigar}, Tablet function f_{tablet} and Twoaxes function f_{twoaxes}. In these functions, when the ϵ-CMA-ES randomly evaluates the minor eigenspace, the variances of the mutation distribution in the directions in the minor eigenspace are reduced much faster. However, limited benefits were observed for objective functions with eigenspectra which are more evenly distributed, such as the Ellipsoid function f_{cigar} and the Rosenbrock function $f_{\text{rosenbrock}}$. When the problem dimension is 80, the performance advantage of the ϵ-CMA-ES ranges from none (on the Ellipsoid and Rosenbrock functions) to more than 40% (for default default $\lambda = 4 + \lfloor 3 \cdot ln(n) \rfloor$ on Twoaxes function).

In future works, we plan to study how and when the ϵ-CMA-ES should start to randomly evaluate the dominated eigenspace

as the current ϵ-CMA-ES only randomly evaluates the minor eigenspace which has the smallest set of eigenvalues. We expect that the ϵ-CMA-ES can be further improved if the dominated eigenspaces are randomly evaluated with "good" timing during the course of optimization. Finally, a subject of future investigation is the performance of the ϵ-CMA-ES on multimodal test functions.

In the second part of this chapter, we introduced a new sampling method the $(1, \lambda)$-CMA-ES. The sampling method samples search points on the mirrored eigenspace after the eigenspace associated with the covariance matrix are identified. Whether using mirrored eigenspace sampling results in a performance advantage depends on the nature of the objective functions. A significant improvement can be observed on convex quadratic functions which have a large number of eigenvalues in their Hessian matrices that are significantly larger than all the remaining ones. Using the mirrored eigenspace sampling method can enable the $(1, \lambda)$-CMA-ES to quickly reduce the variance of mutations vectors in the eigenspace. We also compared its performance with that of the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES. While it was slightly worse than the $(1, \lambda_{iid} + \lambda_m)$-CMA-ES, the performance is still encouraging when we compare it with the original $(1, \lambda)$-CMA-ES. Overall the best median speed was up to 56% on the sphere function, and the worst median speedup was a loss of 40%, happening only when $n = 3$.

In future works, we plan to investigate the correlation between eigenspace sampling and the eigenvalue distributions of the Hessian matrices of underlying optimization functions. We believe that the CMA-ES can exploit the information of eigenvalue distributions and it can further reduce the number of steps to learn the covariance matrix to an optimal one. Investigating the eigenspace sampling in the traditional ES and CMA-ES with weight recombinations is also one of our interests. Finally, a pos-

sible subject of future work is applying eigenspace sampling to the elitist version of the $(1 + 1)$-CMA-ES.

□ **End of chapter.**

Part III

Dynamic Environments

Chapter 7

CMA-ES in Dynamic Environments

I do not know whether I was a men dreaming I was a butterfly, or whether I am now a butterfly dreaming I am a man.

Zhuangzi

7.1 Motivations

In recent years, there has been a fair amount of research works that have contributed to the state-of-the-art covariance matrix adaptation evolution strategies (CMA-ES) [106, 104, 189, 115] that are used to solve many black-box optimization problems. CMA-ES usually optimizes the real-valued objective functions $f : \mathbb{R}^n \to \mathbb{R}$ in the continuous domain. On ill-conditioned problems, covariance matrix adaptation can accelerate the rate of convergence of evolution strategies by orders of magnitude. For example, a successful covariance matrix adaptation can enable strategies to generate candidate solutions predominantly in the direction of narrow valleys. The CMA-ES is able to learn the appropriate covariance matrix from successful steps that the algorithm has taken. The covariance matrix is updated such that variances in directions of the search space that have previously

been successful are increased while those in other directions are decreased. Even for a small population, the accumulation of information over a number of successful steps can reliably adapt the covariance matrix.

However, the problem classes that have been considered in most of these works are of a static nature. In contrast, many problems in engineering, computational and biological domains are dynamic in that the objective functions are not constant, but vary with time. Examples of dynamic optimization problems arise in the context of online job scheduling, where new jobs arrive in the course of optimization. A complete list of survey and works on the evolutionary algorithms for dynamic optimization has been reviewed by [118, 46].

There are a few works that focus on evolution strategies in dynamic optimization. The early work [220] empirically studied the family of evolution strategies in dynamic rotating problems. It investigated the performance when evolution strategies employed different forms of mutation step size adaptation. The experimental results show that a simple mutation step size adaptation achieves the best results compared to other complicated adaptation mechanisms, including covariance matrix adaptation. It also suggested the use of small populations in evolution strategies because using large populations implies a higher degree of dynamism, which is undesirable in dynamic optimization. Another work [201] studied evolution strategies for the number of mutation step sizes required when the optima moves in one or all n coordinates with different severity. The results showed that adapting all n mutation step sizes achieves a better performance than adapting a single mutation step size. The work [29] compares different variants of mutative self-adaptation and shows that the lognormal self-adaptation used in evolution strategies performs better than the variants of self-adaptation commonly used in evolutionary programming. Obviously all these works

demonstrate the difficulty of understanding the behavior of evolution strategies, their operators and their parameters in dynamic environments.

7.2 Experimental Validation on CMA-ES

7.2.1 Setup

In our setup, we use the same set of problems in [129]. A total of six dynamic problems F_1 to F_6 are tested[1] . All six problems are multi-modal, scalable, rotated and have a large number of local optima. Unless stated otherwise, a change will occur only after $10^2 \cdot n$ number of functions evaluations are used. Fifty independent runs are executed per problem and per change. All problems have the global optimum within the given bounds and there is no need to perform searches outside of the given bounds for these problems. All algorithms will be terminated when the number of changes reaches 60. To evaluate the performance of the algorithms for maximization problems, we record the relative function error value $E^{last}(t) = \frac{f(\mathbf{x}_{best}(t))}{f(\mathbf{x}^*(t))}$ after each change. The vector $\mathbf{x}_{best}(t)$ is the best solutions found by the algorithm at time t and the vector $\mathbf{x}^*(t)$ is the location of the global optimum at time t. In our experiments, all three variants of CMA-ES and the (1+1)-ES with the one-fifth success rule are compared on six benchmark problems. All strategy parameter of evolution strategies are set to the default values in their original works [184, 106, 104, 189, 115]. No parameters tuning has been conducted.

7.2.2 Results

The experimental results are shown in Figure 7.1. The graphs show the relative function error values against the change types.

[1]For details of functions please reference to [129]

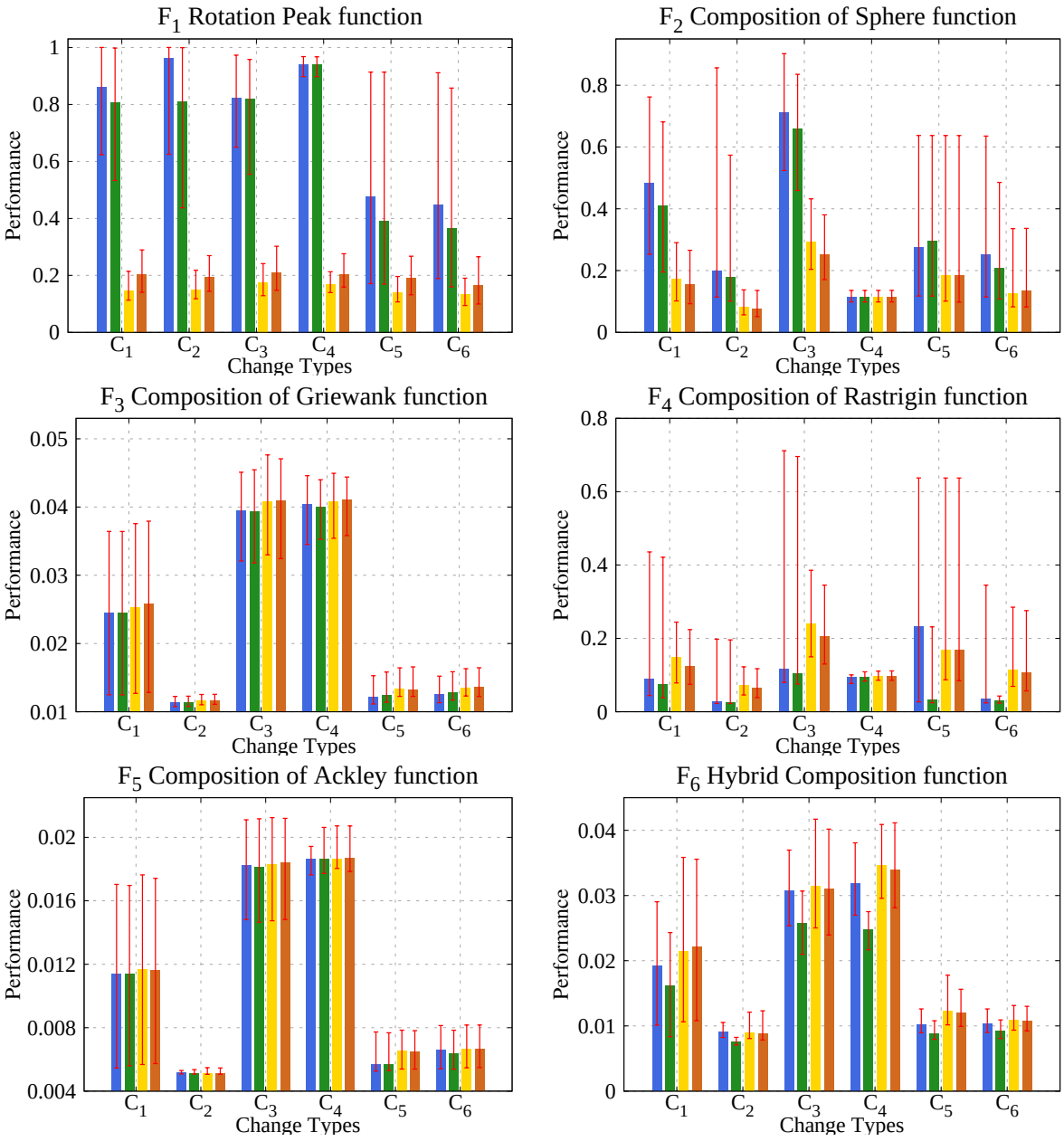

Figure 7.1: The median performance of the CMA-ES variants and the (1+1)-ES with the one-fifth success rule on F_1 to F_6 over the trials of 50. The dynamic change types are C_1 Small step change, C_2 Large step change, C_3 Random step change, C_4 Chaotic change, C_5 Recurrent change and C_6 Recurrent change step with noise. For each change type, from left to right, the bars represent the (1+1)-ES with the one-fifth success rule (blue), (1+1)-CMA-ES (green), the standard CMA-ES (yellow) and the sep-CMA-ES (orange).

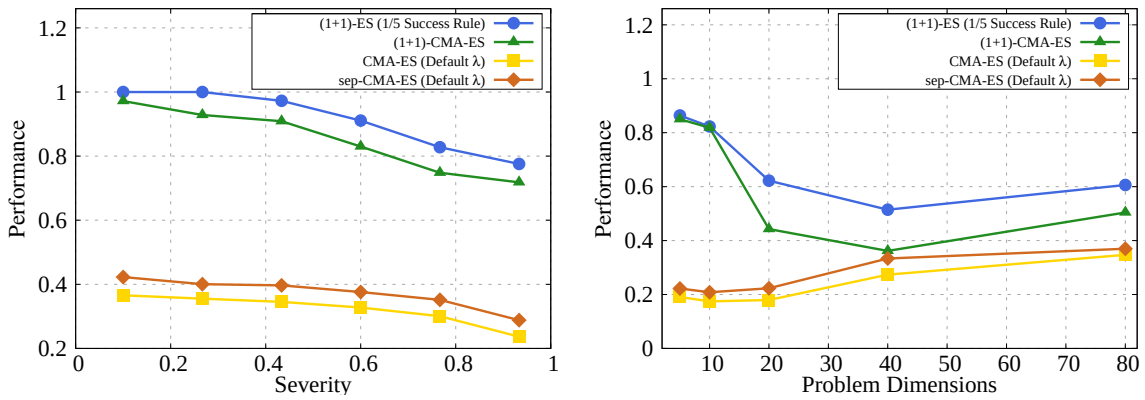

Figure 7.2: The median performance of the CMA-ES variants and the (1+1)-ES with the one-fifth success rule on F_1 when the change type is C_3 random step change. The left graph shows the performance against the severity of the dynamic changes while the right graph shows the performance against the problem dimensions.

The whiskers in the graphs mark the 10th and 90th percentiles. We first take a look at the first problem F_1 Rotation Peak Problem. The performances of the (1+1)-ES and the (1+1)-CMA-ES generally outperform the standard CMA-ES and the sep-CMA-ES. The results are consistent for all 6 types of dynamic changes. An elitist evolution strategy using a small population size leads to the best results compared to all other strategies that are non-elitist and are in large population sizes. Both the (1+1)-ES and the (1+1)-CMA-ES are statistically indistinguishable for all 6 types of dynamic changes. Comparing the standard CMA-ES with the sep-CMA-ES, their performances are also statistically equivalent. None of our statistical tests is able to show any significance in these two strategies. Obviously the simple adaptation technique like the one-fifth success rule can adapt quickly to the dynamically changing environments. The more complicated mechanisms that produce very good results in static optimization, are not adapting very well for dynamic optimization. If we look at the graph for F_2, the results are consistent with those in F_1. The (1+1)-ES and the (1+1)-CMA-ES achieve the

best performance. From functions F_3 to F_6, the performance of all strategies becomes worser since the fitness landscapes are more rugged than the functions F_1 and F_2. However, the performance differences between the elitist and non-elitist versions become smaller. In some of the cases in functions F_3 and F_5, all four variants are statistically equivalent. In functions F_4 and F_6, there are a few cases where the CMA-ES and the sep-CMA-ES outperform the (1+1) variants. Overall all strategies in most of the cases are indistinguishable in functions F_4 and F_6.

We next investigate how robust these strategies are to dynamic changes with different severity and to the problem dimensions. Figure 7.2 shows the median numbers of relative function errors against the severity when the underlying function is F_1 and the change type is C_3. The severity $\phi_{severity}$ is normalized such that severity is equal to $\frac{\phi_{severity}}{|\phi|}$ where $|\phi|$ is the range of the system control parameters. When we increase the severity, the performance generally becomes worse. The elitist (1+1)-ES and the (1+1)-CMA-ES are generally better than the non-elitist strategies in dynamic changes with different severity. Lastly we investigate the performance of the strategies when the dimensions are increased. The right graph in Figure 7.2 shows the median numbers against the problem dimensions. The performance of elitist strategies becomes worse when the problem dimensions are scaled up. We believe this is due to the small population sizes of these point-based (1+1) strategies. In contrast to the elitist strategies, the non-elitist version of the CMA-ES that are population-based improves gradually in higher dimensions. The performance gap between the elitist and non-elitist CMA-ES is getting smaller. Obviously when we increase the problem dimension, the dynamic problems become more challenging and it is necessary to use the population-based strategies in order to achieve a reasonable performance.

7.3 Future Perspective

In this chapter, we investigate the state-of-the-art CMA-ES variants for dynamic optimization and they include the elitist (1+1)-CMA-ES, the standard (μ,λ)-CMA-ES and the sep-(μ,λ)-CMA-ES. We first briefly review the CMA-ES variants in the context of static optimization and then we discuss the latest dynamic optimization benchmark problems that are used in our simulations. In one out of the six dynamic functions, the elitist (1+1)-ES with the one-fifth rule and the (1+1)-CMA-ES achieve the best performance. In most of our simulations, these two elitist strategies are statistically equivalent. The non-elitist strategies, including the standard (μ,λ)-CMA-ES and the sep-CMA-ES, are outperformed by the elitist variants. The results are consistent for dynamic changes with different severity. However, the performance of the elitist strategies, which are pointed-based search algorithms, becomes worse for higher dimensional problems. Using the population-based strategies like the standard (μ,λ)-CMA-ES and the sep-CMA-ES can achieve the equivalent performance as what the elitist (1+1) variants can do.

In the future work, it would be interesting to introduce additional diversity into the CMA-ES variants. Concentrating the search near the the current optima in a dynamic environment could make the strategies miss the important changes in different regions of the search space. Adding predictions mechanism and diversity control methods can be a promising way for CMA-ES to optimize the dynamic functions.

□ **End of chapter.**

Chapter 8

CCEA in Dynamic Environments

I've failed over and over and over again in my life and that is why I succeed.

Michael Jordan

8.1 Motivations

Coevolutionary algorithms are one of the popular evolutionary algorithms (EAs) and they are fundamentally different from the traditional EAs. Fitness evaluations in the coevolutionary algorithms always depend on the outcomes of interactions between individuals. Traditionally, the coevolutionary algorithms can be classified into the competitive coevolution [190] and the cooperative coevolution [173]. In a competitive coevolutionary algorithm, the individuals compete against others. The increase in the fitness of an individual will decrease the fitness of another individual. In a cooperative coevolutionary algorithm (CCEA), the fitness demonstrates how well the individuals perform when they collaborate. Higher fitness is given to the individuals when they perform well. Lower fitness is given when they perform poorly. Intuitively, one may consider the CCEAs more superior to the traditional EAs because the CCEAs decompose the search

space when they search the optima for optimization problems. Each population of a CCEA only requires searching the projection of a n-dimensional problem and it is therefore natural to consider that the CCEAs perform better. However, the work [222] reports that each CCEA population easily searches for components of candidate solutions that are robust to the other components. Most importantly, the combinations of all these components are not always globally optimal. Therefore there are many works in the literature that aim at understanding the behaviour of the CCEAs [171] and improving the performance of the CCEAs for static optimization [161].

While most of these works use the CCEAs to optimize those problems with the objective functions that remain constant during the course of optimization, their behaviour in the dynamic environments [46, 145, 233] has not yet been explored. A recent empirical investigation [13] has studied the CCEAs in using the hypermutations [58] and the random immigrant (RI) scheme [59, 89] to optimize two dynamic problems that are produced by the problem generator from [143]. These two dynamic problems are called *one moving peak problem* and *two moving peaks problem*. They basically imitate the typical dynamic environments that exists in many real world problems. Interestingly, the experimental results show that using a cooperative coevolutionary approach achieves a better performance. In particular, using the random immigrants in a CCEA has prominent advantages over using them in other EAs.

Following the work [13], Au and Leung extend their works [14] by investigating the behaviour of a CCEA in the multimodal environments where the locations, the coverage and the heights of the moving peaks are changing during the course of optimization. Specifically, the work studies a CCEA in using different combinations of the collaboration methods in the orig-

inal individuals[1] and the RI individuals. It investigates how the choices for the collaboration methods can greatly influence the performance of a CCEA. Using the best collaboration method in the original individuals shows a better performance in the moderate-changing and the slow-changing environments. Using the random collaboration method in the original individuals is more promising in the fast-changing environments. The best choice of the collaboration method in the RI individuals, however, depend on the collaboration methods in the original individuals. When the original individuals use the best collaboration method, the collaboration methods used by RI individuals are no longer significant. When the random collaboration method is used in the original individuals, the choices for the collaboration methods in the RI individuals become significant. By choosing the appropriate collaboration methods in the original individuals and the RI individuals, a CCEA using the RI scheme can consistently outperform an EA using the RI scheme.

We further generalize the works [13, 14] by studying the performance of a CCEA on a new set of the dynamic benchmark problems. The new dynamic benchmark problems were introduced by [129, 128] and were used in 2009 IEEE Congress on Evolutionary Computation (CEC) competition to evaluate the state-of-the-art algorithms for dynamic optimization. Our motivation is to benchmark the performance of a CCEA and to provide a more thorough analysis on the performance of the CCEAs on a wide variety of the problem classes. To our best knowledge, the understanding of a CCEA in the literature is very minimal and there is a need to understand how it behaves in the dynamic environments. We follow the work in [14] in using the RI scheme. The motivation of using the RI individuals is to provide an additional diversity for the CCEAs and

[1] In [14], the original individuals referred to the parents and the offspring of an evolutionary algorithm.

this is particularly useful in the changing environments. Since the RI individuals are randomly generated in the search space and they have no dependency on the individuals in the current generation, using the RI individuals can increase the probability of searching other optima that are not yet explored. However, there is also a drawback in using the RI individuals. The diversity brought by them will also weakened the selection pressure and is undesirable for local convergence. In order to balance out the weaken selection pressure, we introduce the CCEA with a new type of individuals. It is called the *elitist* individuals[2]. We refer to elitists as individuals having the highest fitness as well as their offspring. The elitist individuals are generated from the current best individuals of the populations and they can increase the convergence rate to the local optima. The balance between the exploration and the exploitation on the search space can be achieved by combining the use of the RI individuals and the use of elitist individuals. In the experimental setup, we will demonstrate that either the use of the RI individuals or the elitist individuals separately cannot improve a CCEA. Only when both the RI individuals and the elitist individuals are used, a CCEA can be improved to give its best performance.

We also investigate the effect of various CCEA settings and study how these settings can influence the performance. The first setting is related to the choices for the collaboration methods. It is interesting to re-evaluate whether the best collaboration method is still promising when the new type of the elitist individuals is used in the new dynamic benchmark problems. The second setting is about the selection schemas used by the CCEAs. We use the well-known plus-comma selections in the CCEAs and investigate if the different selection pressure in the plus-comma selections can influence the performance. Another

[2]In the literature, elitists are generally refer to the individuals having the highest fitness and they are unmutated from generation to generation

interesting setting is to investigate the use of the mutative σ-self adaptation in a CCEA. Self-adaptation for the mutative step sizes is common in the standard ES. Therefore it is interesting to understand if using the self-adaptation in a CCEA has any advantage over using the fixed mutative step sizes in a CCEA. Lastly we will evaluate the CCEA in the high dimensional dynamic problems and compare its scalability with that of the standard ES.

8.2 Algorithms Under study

In this section, we describe how a CCEA is used in dynamic optimization. We use the well-known $(\mu \overset{+}{,} \lambda)$-selection scheme from evolution strategies (ES), apply them in a CCEA and investigate its performance in our simulations. Fundamentally, an ES individual differs from a CCEA individual. In a $(\mu \overset{+}{,} \lambda)$-ES, an individual, a, is a 3-tuple $a = [\mathbf{x}_a, \boldsymbol{\sigma}_a, f_a(\mathbf{x}_a)]$ comprising its candidate solution vector $\mathbf{x}_a \in \mathbb{R}^n$, the mutation step size $\boldsymbol{\sigma}_a \in \mathbb{R}^n_+$ and the fitness computed by the objective function being optimized $f : \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \mapsto f(\mathbf{x})$. In a $(\mu \overset{+}{,} \lambda)$-CCEA, a CCEA individual, b, is a 4-tuple $b = [x_b, \sigma_b, \mathbf{c}_b, f_b(\mathbf{c}_b)]$ comprising a component of the candidate solution[3] $x_b \in \mathbb{R}$, the mutation step size $\sigma_b \in \mathbb{R}_+$, the collaboration vector $\mathbf{c}_b \in \mathbb{R}^n$ and the fitness computed by the objective function that takes the collaboration vector $\mathbf{c}_b$ as the input argument. The major difference is that an ES individual directly uses the candidate solution $\mathbf{x}_a$ to evaluate its fitness while a CCEA individual has to first combine the component of its candidate solution x_b with the components of candidate solutions in other individuals to form a collaboration vector $\mathbf{c}_b$ and then uses this vector $\mathbf{c}_b$ to evaluate the fitness of the CCEA individual b.

[3] We can define a CCEA individual such that the number of dimensions for a component of candidate solution is greater than 1, i.e. $\mathbf{x}_b \in \mathbb{R}^d$ for $d > 1$.

To address dynamic optimization, we introduce two types of individuals in the $(\mu \dot{+} \lambda)$-ES and $(\mu \dot{+} \lambda)$-CCEA. They are the *random immigrants (RI)* individuals and the *elitist* individuals. The common way to generate the RI individuals is to randomly generate them in the search space. They are independent from the current populations of the algorithm. The motivation for the RI individuals is to provide the additional diversity for the algorithm. Having these RI individuals can definitely increase the diversity so the algorithms can explore the search space that is not explored by the current populations. In this way, the algorithm can quickly adapt to the changing environments. In the literature, there is an early analysis on the random immigrant approaches [239] and the study examines different mechanisms of generating the immigrants that will influence the performance. The immigrant schemes are classified into two categories: the direct immigrant scheme and the indirect immigrant scheme. In the direct immigrant scheme, the immigrants are generated directly from the current populations. One of the examples is the elitism-based immigrants scheme [231]. In the indirect immigrant scheme, a model is first built and the immigrants are generated based on this model. An example is the memory-based immigrants [230]. The last type of the immigrant scheme is a hybrid that combines the direct immigrants and the indirect immigrants.

Adding diversity by using the RI individuals can provide additional exploration for the algorithm. However, this also weakens the selection pressure and is undesirable for the local convergence to the optima. In order to balance the exploration and the exploitation, we introduce another new type of individuals "the *elitist* individuals". While the RI individuals increase the diversity to explore the search space, the elitist individuals are generated by mutating the *best* individuals in the populations. This can exploit the search space and can also increase the lo-

cal convergence rate to the optima. Both the RI individuals and the elitist individuals are used together in order to provide a balance between the exploration and the exploitation on the search space. In the next section, we will show that the use of the RI or elitist individuals cannot improve the CCEA. We name the resulting algorithm as $[\mu \dagger (\lambda + \kappa + \iota)]$-CCEA where κ and ι are the numbers of the RI individuals and the elitist individuals respectively.

Algorithm 10 describes the details of the $[\mu \dagger (\lambda + \kappa + \iota)]$-CCEA. The CCEA first initializes n number of CCEA populations $\mathcal{Q}_1^i, \forall i \in 1, \ldots, n$. Each CCEA population $\mathcal{Q}_k^i$ consists of μ number of parents that are first evaluated by the random collaboration method. We are not able to use the best collaboration method at the beginning, because we need the "best individuals" and none of the individuals is evaluated when the algorithm starts. Algorithm 6 shows how a CCEA individual selects its collaborators randomly. In step 7 of the Algorithm 6, the CCEA individual b selects the collaborators from other populations by using a uniform distribution $\mathcal{U}(1, |\mathcal{Q}_k^i|)$ where $|\mathcal{Q}_k^i|$ represents the population size of $\mathcal{Q}_k^i$ and $\mathcal{U}(1, |\mathcal{Q}_k^i|)$ uniformly returns the random integer between 1 and $|\mathcal{Q}_k^i|$. Every individual of the same population has the same probability of becoming the component of the collaboration vector $\mathbf{c}$ for the CCEA individual b. Note that in RandomCollaborate(), the fitness of individuals is not considered. Another collaboration method is the *best collaboration method* and it is shown in Algorithm 7. The major difference between RandomCollaborate() and BestCollaborate() is in step 7 where the individuals with the maximum fitness are selected. Both of the procedures return the collaboration vector $\mathbf{c} \in \mathbb{R}^n$ that is used to evaluate the fitness of the CCEA individual b. Finally the collaboration methods are *sequential*, meaning that the CCEA always uses the updated individuals in the populations $\mathcal{Q}_k^i, \ldots \mathcal{Q}_k^n$.

After selecting the collaborators in step 5, the CCEA enters the main loop. The parents are re-evaluated because the objective function may change in every generation and the fitness of the parent evaluated in the last generation can be irrelevant in the current generation. Step 10 is calling either RandomCollaborate(...) or BestCollaborate(...). After reevaluating the parents, it then assigns the parents into the population for the next generation $\mathcal{Q}^i_{k+1}$ if the plus selection is used.

The next step is to generate λ number of offsprings sequentially. An offspring is first cloned from its parent. If the mutative σ-self adaptation is used, its mutation step size is updated. After the mutation step size is updated, the offspring mutates. This is done in step 22 where $\sigma_{q'}$ is the updated mutation step size and $\mathcal{N}(0,1)$ is a random scalar drawn from a normal distribution. The collaboration vector $\mathbf{c}_{q'}$ is formed by calling either one of the collaboration procedures. Then, the offspring is added to the population $\mathcal{Q}^i_{k+1}$.

After the offsprings are generated, κ number of the RI individuals are generated. In step 28, the RI individuals are randomly generated in the search space. The notation $\mathcal{U}(l_i, u_i)$ represents a random scalar drawn from the uniform distribution in the ith dimension with a lower bound l_i and an upper bound u_i. Similarly, the RI individuals collaborate with other individuals, by calling either RandomCollaborate(...) or BestCollaborate(...). Note that there is a difference in the input arguments when we call these two procedures in step 23 and step 29. In step 23 we use the parental population $\mathcal{Q}^i_k$ while in step 29 we use the latest population $\mathcal{Q}^i_{k+1}$ which consists of both the parents and offspring for the plus selection or just the offspring for the comma selection. After the evaluations, the RI individuals are added to the CCEA populations $\mathcal{Q}^i_{k+1}$ in step 31.

From step 32 to step 41, the elitist individuals are gener-

ated sequentially. Each elitist individual is first copied from the best individual $b_{1:|\mathcal{Q}^i_{k+1}|}$ and is then mutated by a Gaussian distributed random scalar. Similar to the generation of the offspring, the mutation step sizes can be updated by the mutative σ-self adaptation. The elitist individuals collaborate with other individuals to evaluate the fitness and they are added to the CCEA populations $\mathcal{Q}^i_{k+1}$.

After all the individuals are generated, the selection takes place in step 43. According to the fitness, $(\mu+\lambda+\kappa+\iota)$ number and $(\lambda+\kappa+\iota)$ number of individuals are ordered when the plus selection and the comma selection are used respectively. The best μ individuals in each population $\mathcal{Q}^i_k$ are selected for the next generation. Finally we go back to step 3 and continue until the termination condition is met.

There can be many combinations of how the parents, the offsprings, the RI individuals and the elitist individuals collaborate. Table 8.1 summarizes these combinations. There are two variants of the CCEA. In the $[\mu \overset{+}{,} (\lambda+\kappa+\iota)]$-bCCEA and the $[\mu \overset{+}{,} (\lambda+\kappa+\iota)]$-rCCEA, all individuals use the best collaboration method and the random collaboration method respectively.

Algorithm 11 describes the $[\mu \overset{+}{,} (\lambda+\kappa+\iota)]$-ES that is used in our simulations. The objective is to investigate if using the cooperative coevolutionary approach for dynamic optimization demonstrates any advantages. The ES first initializes the parental population $\mathcal{P}_1$ that consists of μ number of the parents. Each parent p_i in $\mathcal{P}$ is evaluated from step 4 to step 5. The off - spring are generated from step 10 to step 17. A parent is first uniformly selected in step 11 and is used to clone a new offspring that is mutated in step 15. If the mutative σ-self adaptation is used, the mutation step sizes are updated before the offsprings mutate. The new offspring is evaluated in step 16 and is added to the ES population $\mathcal{P}^{k+1}$.

The RI individuals are created in a similar way as those in

Algorithm 10: Pseudo Code of $[\mu \dagger (\lambda + \kappa + \iota)]$ Cooperative Coevolutionary Algorithm (CCEA)

Given: $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma > 0$
Initialise $\mathcal{Q}_1^i, \forall i \in \{1, \ldots, n\}$,$k = 1$
for $i = 1$ *to* n **do**
    for $j = 1$ *to* μ **do**
        $\mathbf{c}_{q_{ij}} \leftarrow$ `RandomCollaborate`$(q_{ij}, \mathcal{Q}_k^1, \ldots, \mathcal{Q}_k^n)$
        $f_{q_{ij}} \leftarrow f\left(\mathbf{c}_{q_{ij}}, k\right)$
repeat
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* μ **do**
            $\mathbf{c}_{q_{ij}} \leftarrow$ `Collaborate`$(q_{ij}, \mathcal{Q}_k^1, \ldots, \mathcal{Q}_k^n)$
            $f_{q_{ij}} \leftarrow f\left(\mathbf{c}_{q_{ij}}, k\right)$
    if *Plus-selection* **then**
        $\mathcal{Q}_{k+1} \leftarrow \mathcal{Q}_k$
    else
        $\mathcal{Q}_{k+1} \leftarrow \emptyset$
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* λ **do**
            $k \leftarrow \mathcal{U}(1, \mu)$
            $q' \leftarrow q_{ij} \in \mathcal{Q}_k^i$
            if *Self-adaptive* **then**
                $\sigma_{q'} \leftarrow \sigma_{q_{ij}} \exp\left(\frac{\mathcal{N}_i(0,1)}{\sqrt{2n}} + \frac{\mathcal{N}_j(0,1)}{\sqrt{2\sqrt{n}}}\right)$
            $x_{q'} \leftarrow x_{q_{ij}} + \sigma_{q'} \cdot \mathcal{N}(0, 1)$
            $\mathbf{c}_{q'} \leftarrow$ `Collobrate(...)`
            $f_{q'} \leftarrow f\left(\mathbf{c}_{q'}, k\right)$
            $\mathcal{Q}_{k+1}^i \leftarrow \mathcal{Q}_{k+1}^i \cup \{q'\}$
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* κ **do**
            $x_{q'} \leftarrow \mathcal{U}(l_i, u_i)$
            $\mathbf{c}_{q'} \leftarrow$ `Collobrate(...)`
            $f_{q'} \leftarrow f\left(\mathbf{c}_{q'}, k\right)$
            $\mathcal{Q}_{k+1}^i \leftarrow \mathcal{Q}_{k+1}^i \cup \{q'\}$
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* ι **do**
            $l \leftarrow 1 : |\mathcal{Q}_i^{k+1}|$
            $q' \leftarrow q_l \in \mathcal{Q}_k^i$
            if *Self-adaptive* **then**
                $\sigma_{q'} \leftarrow \sigma_{q_{ij}} \exp\left(\frac{\mathcal{N}_i(0,1)}{\sqrt{2n}} + \frac{\mathcal{N}_j(0,1)}{\sqrt{2\sqrt{n}}}\right)$
            $x_{q'} \leftarrow x_{q_{ij}} + \sigma_{q'} \cdot \mathcal{N}(0, 1)$
            $\mathbf{c}_{q'} \leftarrow$ `Collobrate(...)`
            $f_{q'} \leftarrow f\left(\mathbf{c}_{q'}, k\right)$
            $\mathcal{Q}_{k+1}^i \leftarrow \mathcal{Q}_{k+1}^i \cup \{q'\}$
    for $i = 1$ *to* n **do**
        $\mathcal{Q}_{k+1}^i \leftarrow \left\{b_{j:|\mathcal{Q}_{k+1}^i|} | 1 \leq j \leq \mu\right\}$
    $k \leftarrow k + 1$
until *termination condition is met*

Table 8.1: Different Combinations of Collaboration Methods in Algorithm 10

COLLABORATE(...) Methods	Algorithms	
	bCCEA	rCCEA
Parents (in line 10)	BESTCOLLABORATE()	RANDOMCOLLABORATE()
Offsprings (in line 23)	BESTCOLLABORATE()	RANDOMCOLLABORATE()
RI Individuals (in line 29)	BESTCOLLABORATE()	RANDOMCOLLABORATE()
Elitist Individuals (in line 39)	BESTCOLLABORATE()	RANDOMCOLLABORATE()

the CCEA. The major difference is that the candidate solutions in the RI individuals are assigned with the vector $\boldsymbol{\mathcal{U}}(\mathbf{l}, \mathbf{u})$ that is a nth-dimensional random vector with the independent components drawn from the uniform distribution with a lower bound vector $\mathbf{l}$ and an upper bound vector $\mathbf{u}$. All RI individuals and the elitist individuals are added to the current population after evaluations. In step 30, the algorithm selects the new parental population $\mathcal{P}_{k+1}$. The notation $p_{i:|\mathcal{P}_{k+1}|}$ represents the i-th best individuals in the population $\mathcal{P}_{k+1}$. Notice that an ES is allowed to have the self-adaptation on its mutation steps sizes in order to have a fair comparison when both ES and CCEA use the mutative σ-self adaptation schemas.

8.3 Experimental Study

8.3.1 Benchmark

In the literature, there are many dynamic problem generators, including the moving peak benchmark (MPB) proposed by Branke [45], and the DF1 generator proposed by Morrison and De Jong [143]. Both of them consist of the multi-dimensional landscapes where the heights, the widths and the positions of the peaks can be changed during the course of optimization. They generate a predefined number of peaks in a fixed problem dimensions. We use the generalized dynamic benchmark generator (GDBG) [129, 128] to evaluate the performance of the $[\mu \dot{+} (\lambda + \kappa + \iota)]$-

Algorithm 11: Pseudo Code of $[\mu \stackrel{+}{,} (\lambda + \kappa + \iota)]$ Evolution Strategies

Given: $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{x}_1, \boldsymbol{\mathcal{N}} \in \mathbb{R}^n$, $\sigma > 0$
Initialise $\mathcal{P}_1$ and $k = 1$
repeat
  for $i = 1$ *to* μ **do**
    $f_{p_i} \leftarrow f\left(\mathbf{x}_{p_i}, k\right)$
  if *Plus-selection* **then**
    $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k$
  else
    $\mathcal{P}_{k+1} \leftarrow \emptyset$
  for $i = 1$ *to* λ **do**
    $k \leftarrow \mathcal{U}(1, \mu)$
    $p' \leftarrow p_k \in \mathcal{P}_k$
    if *Self-adaptive* **then**
      $\boldsymbol{\sigma}_{p'} \leftarrow \boldsymbol{\sigma}_{p_k} \exp\left(\frac{\boldsymbol{\mathcal{N}}_i(0,1)}{\sqrt{2n}} + \frac{\boldsymbol{\mathcal{N}}(0,1)}{\sqrt{2\sqrt{n}}}\right)$
    $\mathbf{x}_{p'} \leftarrow \mathbf{x}_{p_k} + \boldsymbol{\sigma}_{p'} \cdot \boldsymbol{\mathcal{N}}(0,1)$
    $f_{p'} \leftarrow f\left(\mathbf{x}_{p'}, k\right)$
    $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_{k+1} \cup \{p'\}$
  for $j = 1$ *to* κ **do**
    $\mathbf{x}_{p'} \leftarrow \boldsymbol{\mathcal{U}}(\mathbf{l}, \mathbf{u})$
    $f_{p'} \leftarrow f\left(\mathbf{x}_{p'}, k\right)$
    $\mathcal{P}_{k+1} \leftarrow \mathcal{P}_{k+1} \cup \{a'\}$
  for $j = 1$ *to* λ **do**
    $l \leftarrow 1 : |\mathcal{P}_{k+1}|$
    $p' \leftarrow p_l$
    if *Self-adaptive* **then**
      $\boldsymbol{\sigma}_{p'} \leftarrow \boldsymbol{\sigma}_{p_l} \exp\left(\frac{\boldsymbol{\mathcal{N}}_i(0,1)}{\sqrt{2n}} + \frac{\boldsymbol{\mathcal{N}}(0,1)}{\sqrt{2\sqrt{n}}}\right)$
    $\mathbf{x}_{p'} \leftarrow \mathbf{x}_{p_l} + \boldsymbol{\sigma}_{p'} \cdot \boldsymbol{\mathcal{N}}(0,1)$
    $f_{p'} \leftarrow f\left(\mathbf{x}_{p'}, k\right)$
    $\mathcal{P}^{k+1} \leftarrow \mathcal{P}^{k+1} \cup \{p'\}$
  $\mathcal{P}_{k+1} \leftarrow \left\{p_{i:|\mathcal{P}_{k+1}|} | 1 \leq i \leq \mu\right\}$
  $k \leftarrow k + 1$
until *termination condition is met*

CCEA and the $[\mu \dagger (\lambda + \kappa + \iota)]$-ES. This benchmark is a generalized benchmark generator that constructs the dynamic environments in the continuous space. The benchmark problem was used in 2009 IEEE Congress on Evolutionary Computation (CEC) competition to evaluate the state-of-the-art algorithms for dynamic optimization. It differs from the MPB and the DF1 benchmarks in that it uses the rotation method instead of shifting the positions of the peaks.

8.3.2 Setup

We use the same set of problems in [129]. Table 2.1 summarizes the parameters used in the rotation DBG and the composition DBG. A total of six dynamic problems F_1 to F_6 are tested. All six problems are multi-modal, scalable, rotated and have a large number of local optima. Unless stated otherwise, the problem dimension n is set to 10 and a change will occur only after $100 \cdot n$ functions evaluations are used. Fifty independent runs are executed per problem and per change.

An uniform random initialization in the search space will be used. All algorithms need to detect the non-dimensional change by itself instead of being informed when a non-dimensional change occurs. For all the dimensional changes, all algorithms will be informed when a dimensional change occurs. All algorithms will be terminated when the environment changes 10 times. To evaluate the performance of the algorithms, when each change occurs, we record the ratios of $f(\mathbf{x}^*(t))/f(\mathbf{x}_{best}(t))$ and $f(\mathbf{x}_{best}(t))/f(\mathbf{x}^*(t))$ for minimisation and maximisation respectively. The vector $\mathbf{x}_{best}(t)$ is the best solutions found by the algorithm at time t and the vector $\mathbf{x}^*(t)$ is the location of the global optimum at time t.

In all the simulations, we compare the two variants of CCEAs including the bCCEA and rCCEA and the ES on all six bench-

Table 8.2: Comparisons on the median performance of ES and CCEAs using different population sizes. [1]

Algorithms	C_1				
	$\mu = 1$, $\lambda = 1$	$\mu = 10$, $\lambda = 40$	$\mu = 10, \lambda = 20$, $\kappa = 10, \iota = 10$	p-value [2]	p-value [3]
ES	0.098	0.149	**0.326**	1.3E-74	6.9E-1
bCCEA	0.155	0.384	**0.461**	1.0E-07	3.2E-58
rCCEA	0.150	0.151	0.155	2.4E-01	3.6E-01
Algorithms	C_2				
ES	0.099	0.162	**0.344**	3.3E-81	5.4E-113
bCCEA	0.148	0.384	**0.470**	1.3E-07	1.4E-72
rCCEA	0.157	0.151	**0.156**	1.6E-02	5.0E-01
Algorithms	C_3				
ES	0.110	0.196	**0.369**	6.7E-69	2.2E-119
bCCEA	0.201	0.403	**0.503**	4.6E-10	9.9E-44
rCCEA	0.196	0.171	0.171	3.8E-01	1.7E-04
Algorithms	C_4				
ES	0.105	0.165	**0.393**	2.4E-127	5.7E-141
bCCEA	0.171	0.365	**0.502**	1.2E-29	1.1E-73
rCCEA	0.171	0.157	**0.169**	8.7E-09	2.6E-01
Algorithms	C_5				
ES	0.082	0.152	0.147	9.7E-01	9.5E-57
bCCEA	0.165	0.354	0.351	8.2E-01	9.6E-20
rCCEA	0.185	**0.145**	0.125	1.4E-13	3.2E-28
Algorithms	C_6				
ES	0.084	0.145	**0.184**	1.4E-11	1.7E-57
bCCEA	0.163	0.310	0.337	1.4E-01	3.9E-12
rCCEA	0.164	**0.134**	0.118	3.0E-05	2.3E-13

[1] Plus selection and self-adaptation were used in all the algorithms.

[2] A two-tailed Wilcoxon rank sum test was conducted between population sizes ($\mu = 10, \lambda = 40$) and the population sizes ($\mu = 10, \lambda = 20, \kappa = 10, \iota = 10$). If the two results are statistically different, the better one was highlighted in bold.

[3] Another two-tailed Wilcoxon rank sum test was conducted between population sizes ($\mu = 1, \lambda = 1$) and the population sizes ($\mu = 10, \lambda = 20, \kappa = 10, \iota = 10$). If the two results are statistically different, the better one was highlighted in gray.

Table 8.3: Comparison of the median performance of ES and CCEAs using the mutative σ-self adaptation.

Algorithms	C_1							
	Plus-selection			Comma-selection			p-value [4]	
	AD	NAD	p-value [3]	AD	NAD	p-value [3]	AD	NAD
ES	**0.326**	0.260	1.7E-24	0.201	0.201	1.0E+00	4.0E-80	6.9E-53
bCCEA	0.461	0.447	3.2E-01	0.405	0.436	1.4E-01	7.7E-05	1.7E-01
rCCEA	0.155	0.155	9.9E-01	0.140	**0.148**	1.4E-02	6.7E-06	3.0E-02
Algorithms	C_2							
ES	**0.344**	0.288	7.8E-21	0.220	0.220	1.0E+00	1.1E-75	2.1E-58
bCCEA	0.470	0.469	7.2E-01	0.411	**0.433**	4.4E-02	2.5E-05	1.0E-01
rCCEA	0.156	0.156	9.2E-01	0.149	0.153	6.0E-02	9.9E-04	1.0E-01
Algorithms	C_3							
ES	**0.369**	0.295	6.3E-24	0.218	**0.230**	2.7E-03	9.3E-81	2.0E-45
bCCEA	0.503	0.515	2.0E-01	0.428	**0.500**	2.8E-04	2.1E-05	2.2E-01
rCCEA	0.171	0.170	7.0E-01	0.160	0.164	6.9E-02	3.3E-05	8.4E-02
Algorithms	C_4							
ES	**0.393**	0.310	1.5E-46	0.233	0.236	4.8E-01	8.9E-120	5.8E-104
bCCEA	0.502	0.513	6.3E-01	0.459	0.462	6.0E-01	3.5E-06	9.0E-04
rCCEA	0.169	0.168	5.5E-01	0.162	**0.169**	2.2E-03	1.7E-04	9.9E-01
Algorithms	C_5							
ES	0.147	**0.153**	2.7E-02	0.114	0.114	1.0E+00	5.1E-25	5.4E-34
bCCEA	0.351	0.367	8.8E-01	0.324	0.331	7.3E-02	2.2E-02	3.1E-01
rCCEA	0.125	0.127	4.6E-01	0.093	**0.106**	4.6E-16	1.1E-53	9.5E-23
Algorithms	C_6							
ES	**0.184**	0.167	3.1E-09	0.126	0.126	1.0E+00	1.3E-45	7.3E-49
bCCEA	0.337	0.352	5.7E-01	0.323	0.330	1.9E-01	6.9E-02	2.3E-01
rCCEA	0.118	0.127	2.8E-01	0.095	**0.103**	8.4E-05	3.5E-29	6.4E-20

[1] AD means that the mutative σ-self adaptation was used while NAD means that the mutative σ-self adaptation was not in use.

[2] The population sizes μ, λ, κ and ι are 10, 20, 10 and 10 respectively.

[3] A two-tailed Wilcoxon rank sum test was conducted between numbers for AD and NAD. If the two results are statistically different, the better one was highlighted in bold.

[4] Another two-tailed Wilcoxon rank sum test was conducted between numbers for the plus selection and the comma selection. If the two results were statistically different, the better one was highlighted in gray.

Table 8.4: Comparison of the median performance of ES and CCEAs on six dynamic functions.

ChgType	F_1				F_2			
	ES	bCCEA	rCCEA	p-value [3]	ES	bCCEA	rCCEA	p-value [3]
C_1	0.326	**0.461**	0.155	3.1E-28	0.056	0.060	0.052	1.4E-01
C_2	0.344	**0.470**	0.156	6.6E-23	0.026	**0.026**	0.024	1.3E-07
C_3	0.369	**0.503**	0.171	3.1E-26	0.092	**0.094**	0.081	9.2E-03
C_4	0.393	**0.502**	0.169	4.4E-30	0.102	**0.110**	0.100	1.2E-05
C_5	0.147	**0.351**	0.125	4.2E-60	0.028	**0.045**	0.027	3.5E-49
C_6	0.184	**0.337**	0.118	1.0E-28	0.029	**0.043**	0.027	1.4E-55
C_7	0.253	**0.397**	0.149	2.9E-49	0.074	**0.078**	0.073	2.5E-02
ChgType	F_3				F_4			
	ES	bCCEA	rCCEA	p-value [3]	ES	bCCEA	rCCEA	p-value [3]
C_1	0.026	**0.029**	0.025	1.2E-05	0.048	0.050	0.046	2.4E-01
C_2	0.012	**0.014**	0.011	2.4E-62	0.022	**0.023**	0.020	1.8E-06
C_3	0.042	**0.048**	0.039	5.2E-14	0.078	0.080	0.071	8.5E-02
C_4	0.044	**0.049**	0.042	8.2E-32	0.086	**0.091**	0.084	2.4E-05
C_5	0.013	**0.016**	0.012	1.3E-54	0.024	**0.034**	0.023	3.7E-53
C_6	0.013	**0.016**	0.013	2.5E-44	0.025	**0.033**	0.023	2.0E-39
C_7	0.039	**0.045**	0.039	3.2E-14	0.069	**0.072**	0.067	3.6E-02
ChgType	F_5				F_6			
	ES	bCCEA	rCCEA	p-value [3]	ES	bCCEA	rCCEA	p-value [3]
C_1	0.016	**0.018**	0.013	6.5E-05	0.024	0.025	0.021	8.1E-02
C_2	0.007	**0.009**	0.006	4.5E-21	0.011	**0.011**	0.009	6.7E-10
C_3	0.024	**0.029**	0.020	5.8E-17	0.038	**0.039**	0.033	1.3E-03
C_4	0.023	**0.025**	0.020	1.7E-06	0.041	0.040	0.035	9.5E-01
C_5	0.008	**0.016**	0.007	1.5E-124	0.012	**0.019**	0.012	3.0E-74
C_6	0.008	**0.015**	0.007	6.7E-117	0.012	**0.018**	0.012	1.6E-73
C_7	0.021	**0.023**	0.019	4.0E-07	0.038	**0.040**	0.034	1.3E-03

[1] The plus selection and the mutative σ-self adaptation were used in all the algorithms.
[2] The population sizes μ, λ, κ and ι are 10, 20, 10 and 10 respectively.
[3] A two-tailed Wilcoxon rank sum test was conducted between for numbers for ES and bCCEA. If two results are statistically different, the better one is highlighted in bold.

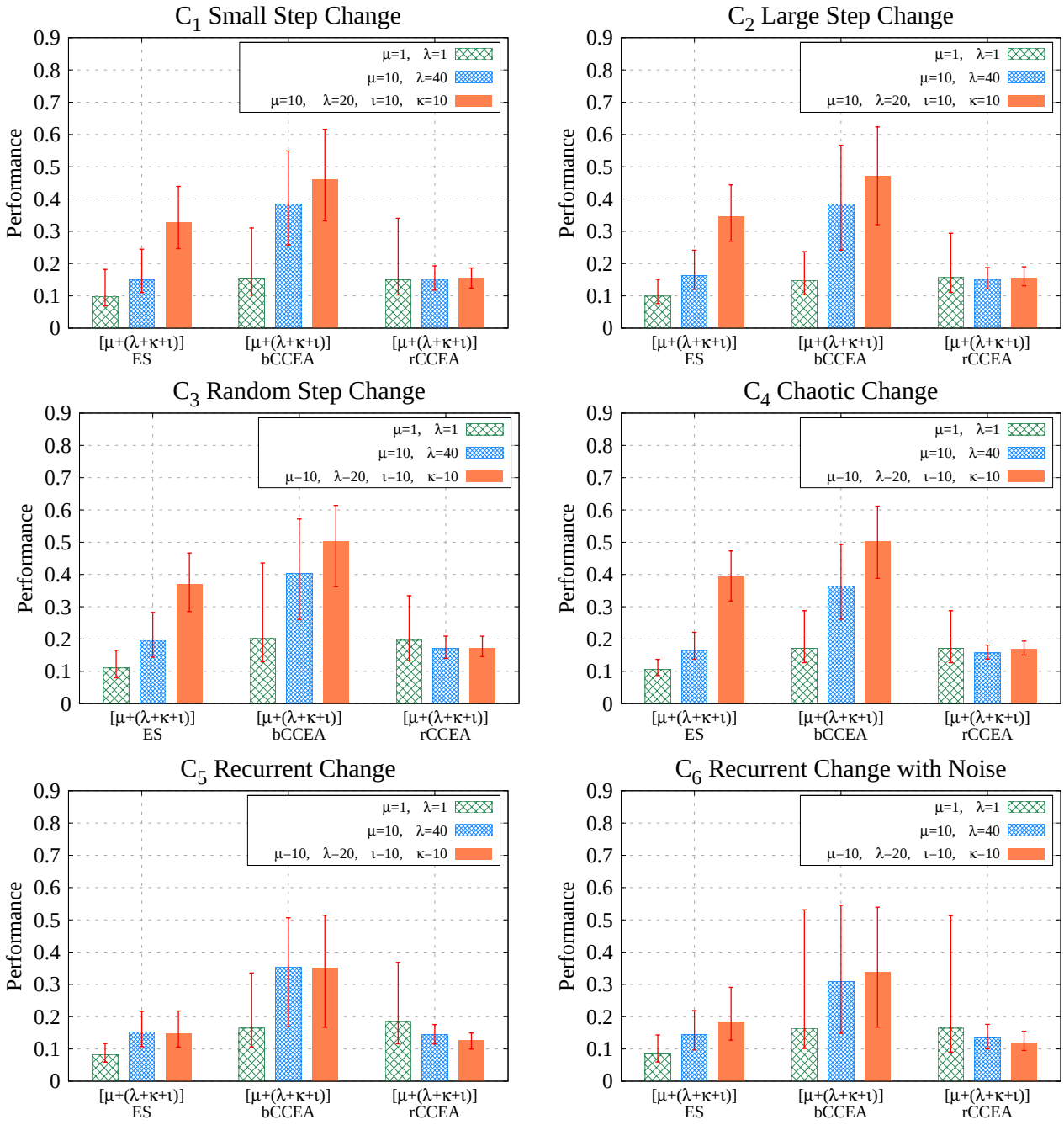

Figure 8.1: Graphs showing the median performance of the ES and the CCEAs on F_1 Rotation Problem when the change frequency τ is $100 \cdot n \cdot \text{FES}$ under the six types of dynamic changes. In all the algorithms, the plus selection and the mutative σ-self adaptation are used. The whiskers mark the 25th and 75th percentiles.

mark problems. The comparison between the use of the best collaboration method and the random collaboration method is significant because the use of the RI individuals and the elitist individuals may influence the best choices for the collaboration methods. The experiments consist of four major sets of experiments and each of them has a different objective. In the first set of experiments, we investigate the effects of using the RI individuals and the elitist individuals. We demonstrate how these two new type of individuals can improve the performance of a CCEA and the standard ES. This is done in our simulations by setting their population sizes as below:

- $\mu = 1, \lambda = 1$;
- $\mu = 10, \lambda = 40$;
- $\mu = 10, \lambda = 20, \kappa = 10, \iota = 10$.

The first and second sets are the baseline algorithms for comparisons. The first setting is to use one parent and one offspring and this is a classical (1+1)-ES that is a point-based search algorithm. Investigating the performance of the point-based search by using the cooperative coevolutionary approach is of interest to us, particularly in the domain of dynamic optimization. The second setting attempts to investigate the population-based algorithms by using 40 offsprings which are four times the problem dimension. The third setting addresses the use of the RI individuals and the elitist individuals by using 10 RI individuals and 10 elitist individuals.[4] By comparing the performance when different population sizes are used, we can understand if the two new types of individuals can improve a CCEA or an ES for dynamic optimization.

[4] The settings were based on the results of the sensitivity analysis in the final set of experiments where the optimal performance was achieved when $\kappa = 10, \iota = 10$. For comparisons with the second settings, we also maintained the total number of offsprings to 40 and hence $\lambda = 20$.

In the second set of experiments, we examine the use of the mutative σ-self adaptation for the mutation step sizes in a CCEA. Using the self-adaptation for the mutation step sizes is common in the context of ES. It is interesting to understand whether the self-adaptation in the context of the CCEA is still applicable and if it can improve a CCEA for dynamic optimization. In addition, we also report the performance when a CCEA uses the plus-selection and the comma selection. The objective is to investigate if the plus-selection or the comma selection is preferable in dynamic environment.

In the third set of experiments, we report the results of the six dynamic problems. This can give us an overview of how CCEAs perform in different dynamic problems. The objective is to understand how robust the algorithms can be when they are used to optimize the dynamic problems with rugged landscapes.

In the final set of experiments, we investigate the sensitivity of the CCEA to the algorithms parameters. These parameters include the offspring population sizes, the RI population sizes and the elitist population sizes. Another objective in this set of experiments is to understand how scalable the CCEAs can be for dynamic problems. This is done by evaluating the CCEA in the high dimensional dynamic problems and by comparing its scalability with that in the standard ES.

8.4 Results and Discussion

8.4.1 Performance when RI and elitist individuals are used

The experimental results of CCEAs that use the RI individuals and the elitist individuals are shown in Figure 8.1. The median performance is reported on the F_1 rotation problem under the six types of dynamic changes. In all the algorithms, the

plus selection and the mutative σ-self adaptation are used. The whiskers mark the 25th and 75th percentiles. We also use the simple $(1+1)$-CCEA and $(\mu+\lambda)$-CCEA to compare with our proposed $[\mu \overset{+}{,} (\lambda+\kappa+\iota)]$-CCEA.

Comparing the bCCEA and the rCCEA, the bCCEA outperforms the rCCEA. The results are consistent across the different types of dynamic changes from C_1 to C_6. This is also aligned with the results from the work [14]. Apparently using the best collaboration method in a CCEA is more promising than using the random collaboration method. If we compare three variants of the CCEAs with different population sizes, the results show that a CCEA using the RI individuals and the elitist individuals generally outperform a CCEA not using these two types of individuals. Table 8.2 shows the statistical comparisons on the algorithms using the population sizes $(\mu = 10, \lambda = 40)$ and the population sizes $(\mu = 10, \lambda = 20, \kappa = 10, \iota = 10)$. For most of the dynamic changes, the $[10+(20+10+10)]$-bCCEA is statistically better than the $(10+40)$-bCCEA. Only when the changes are the recurrent change C_5 or the recurrent change with noise C_6, the median performance of the $[10+(20+10+10)]$-bCCEA and the $(10+40)$-bCCEA are statistically equivalent. Similar results are also observed for the standard ES : the $[10+(20+10+10)]$-ES generally outperforms the $(10+40)$-ES unless when the dynamic change is the recurrent change C_5. Notice that there is a slight difference in the results of the rCCEA. Although using the RI individuals and the elitist individuals can improve a rCCEA for dynamic changes C_2 and C_4, there is no prominent advantage for the rCCEA in other types of the dynamic changes. The performance of the $[10+(20+10+10)]$-rCCEA in C_5 and C_6 is even worse than that of the $(10+40)$-rCCEA. We believe the results are due to the nature of the elitist individuals. Since the elitist individuals are generated from the best individuals in the populations, using the random collaboration method cannot always

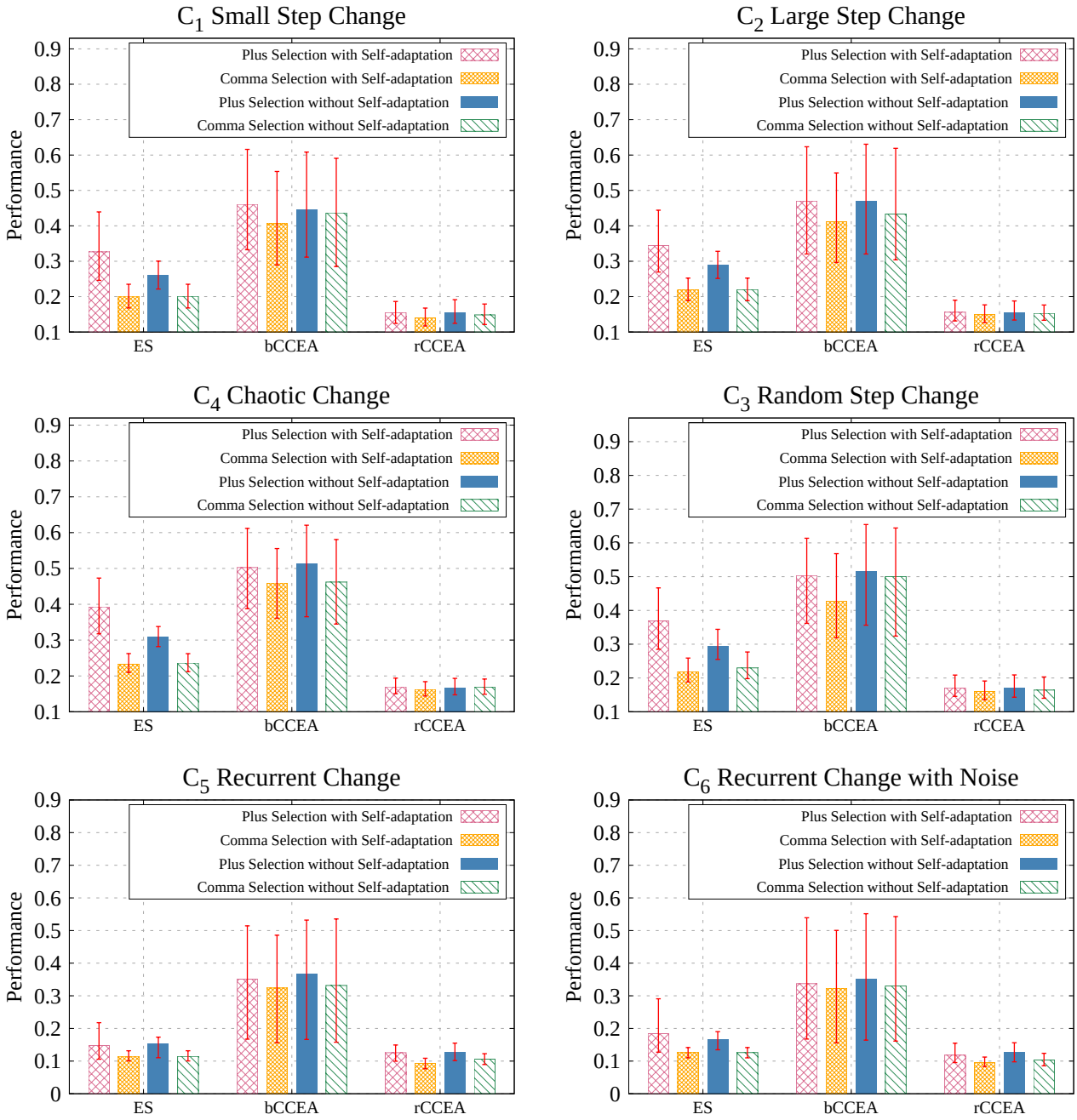

Figure 8.2: Graphs showing the median performance of the ES and the CCEAs on F_1 rotation problem when the plus-comma selection methods and the mutative σ-self adaptation are used. In all the algorithms, the numbers of the parents, the offsprings, the RI individuals and the elitist individuals are 10, 20, 10 and 10 respectively. The whiskers mark the 25th and 75th percentiles.

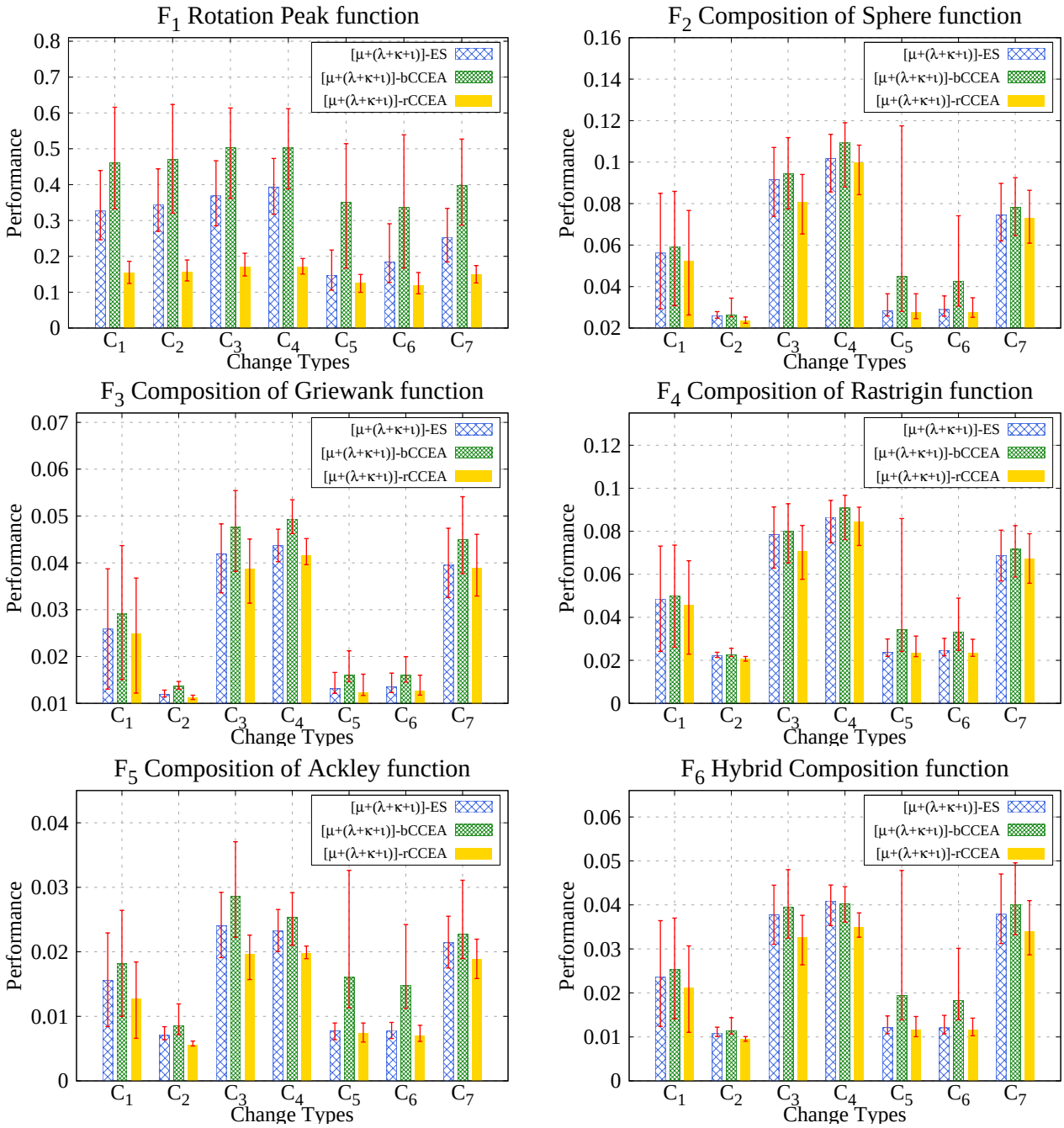

Figure 8.3: Graphs showing the median performance of the ES and CCEAs on six dynamic functions. In all the algorithms, the plus selection and the mutative σ-self adaptation are used. The numbers of the parents, the offsprings, the RI individuals and the elitist individuals are 10, 20, 10 and 10 respectively. The whiskers mark the 25th and 75th percentiles.

provide the best individuals. Another interesting observation is on the performance for the point-based search algorithm including the $(1+1)$-ES and the $(1+1)$-bCCEA. Compared with the population-based algorithms, the point-based algorithms relatively under-perform. This demonstrates that using the population based algorithms is more promising in dynamic environments.

8.4.2 Performance when mutative σ-self adaptation is used

In this section, we compare the performance of the algorithms when the mutative σ-self adaptations are used. Figure 8.2 shows the median performance of the standard ES and the CCEAs on F_1 Rotation Problem when the mutative σ-self adaptation is used. The numbers of the parents, the offsprings, the RI individuals and the elitist individuals are 10, 20, 10 and 10 respectively. Firstly, we compare a CCEA using the plus selection with that using the comma selection. For most of the statistical tests in Table 8.3, a CCEA using the plus selection outperform a CCEA using the comma selection. Similarly, if we compare the bCCEA when the mutative σ-self adaptation is used, mutative σ-self adaptation does not improve a CCEA significantly. Most of the statistical comparisons for bCCEA show the insignificance unless when the dynamic change is the random step change C_3. Similar results are also observed for the rCCEA when it uses the plus selection. However, when the rCCEA uses the comma selection, five out of the six comparisons shows that using the fixed mutation sizes is more promising than using the mutative σ-self adaptation. Lastly when the self-adaptation is used in the standard ES, a significant improvement can be achieved. The statistical comparisons show that an ES with the use of self-adaptation outperforms the one using the fixed mutation

step sizes. From all the results, having the self-adaptation in a CCEA does not appear to improve its performance in dynamic environment and our results show that the performances are insignificant with or without using mutative σ-self adaptation. Unlike the behaviour in the standard ES using the mutative σ-self adaptation, CCEAs show the prominent advantage when self-adaptation is not used.

8.4.3 Performance in different dynamic problems

The first and the second sets of experiments focus on the rotation dynamic problem that is relatively smooth in the fitness landscape. In this section we investigate the dynamic problems where the landscape is rather rugged. Figure 8.3 shows the median performance of the ES and CCEAs on six dynamic functions from F_1 to F_6. In all the algorithms, we use the plus selection and the mutative σ-self adaptation because the settings achieve the best results obtained in the previous subsection. The number of the parents, the offsprings, the RI individuals and the elitist individuals are 10, 20, 10 and 10 respectively. Firstly in the F_1 rotation peak function, the bCCEA achieves the best median performance. The statistical tests in Table 8.4 show that there are only a few cases in F_2, F_4 and F_6 where the performance of the ES and the bCCEA are statistically indistinguishable. Comparing the two variants of the CCEAs, the bCCEA is superior to the rCCEA. Notice that we have conducted the dimensional changes where the number of the problem dimensions can be changed during the optimization. In this type of dynamic change C_7, the bCCEA generally outperforms the other two algorithms. We can see from F_2 to F_6 that the overall performance of all three search algorithms are generally worse. This can be explained by the fact that these five problems are composed of functions that are more rugged than the first problem

F_1. There are many cases that the bCCEA is statistically equivalent to the ES but in most of the cases, the bCCEA using the best collaboration method still outperform the ES.

8.4.4 Sensitivity to population sizes and problem dimensions

Investigating the sensitivity to the properties of the search algorithm is important for us to understand how robust the algorithms can be for different dynamic problems. In this section, we investigate the performance of a CCEA when it uses different population sizes. We also report its performance when the problem dimensions are scaled up.

Offspring population sizes λ

Figure 8.4 plots the median performance of the ES and the CCEAs against the offspring population sizes λ from 1 to 160 on F_1 rotation problem having C_3 random step changes. When we increase the number of the offsprings, it cannot improve both the ES and the rCCEA. The bCCEA can achieve its best median performance when λ is 40. The median performance starts to degrade when we further increase the number of the offsprings to larger than 40. This can be explained by the fact that the number of function evaluations for a generation is $\lambda \cdot n$. When the number of offspring is increased, the number of the generations and hence the number of selection are scaled down by n. This becomes undesirable in the dynamic environments where the function evaluations can be constrained and limited, particularly in the fast changing environment. Therefore the bCCEA underperforms when we increase the number of the offsprings. An ES only requires λ number of function evaluations for its offsprings. When λ is increased, the number of the generations is decreased linearly with λ only.

RI population sizes κ

Figure 8.4 shows the median performance of the ES and the CCEAs against the RI population sizes on the F_1 rotation problem. The number of the offsprings λ is 40. Comparing these three algorithms, a large number of the RI individuals generally does not improve the performance. In the bCCEA, its performance becomes worse when more RI individuals are generated. Similar to the results of the offsprings, a large number of RI individuals means a smaller number of generations. Hence the number of selection is scaled down by n. The other reason is that the generation of the RI individuals is independent from the current populations of the algorithms. Therefore the probability of exploring a promising solution is not improved as expected. The increased volume of search space means that the algorithms require more individuals that are randomly distributed in the search space in order to track the optima. We observe the same behaviour in the ES, which shows having a larger number of the RI individuals does not improve its performance. The best performance of the ES and the bCCEA is around 4.7 and 3.2 respectively, but still they are worse than the performance of the corresponding mean performance in Table 8.4 that is around 0.5 and 0.37 respectively. This indicates the use of the RI individuals alone is not as promising as the use of the RI individuals and the elitist individuals together. It also means the additional diversity brought by the RI individuals cannot improve the algorithms.

Elitist population sizes ι

Figure 8.5 shows the median performance of the ES and the CCEAs against the elitist population sizes ι on the rotation problem F_1. The number of the offspring λ is 40 and no RI individuals are generated during the course of optimization. The

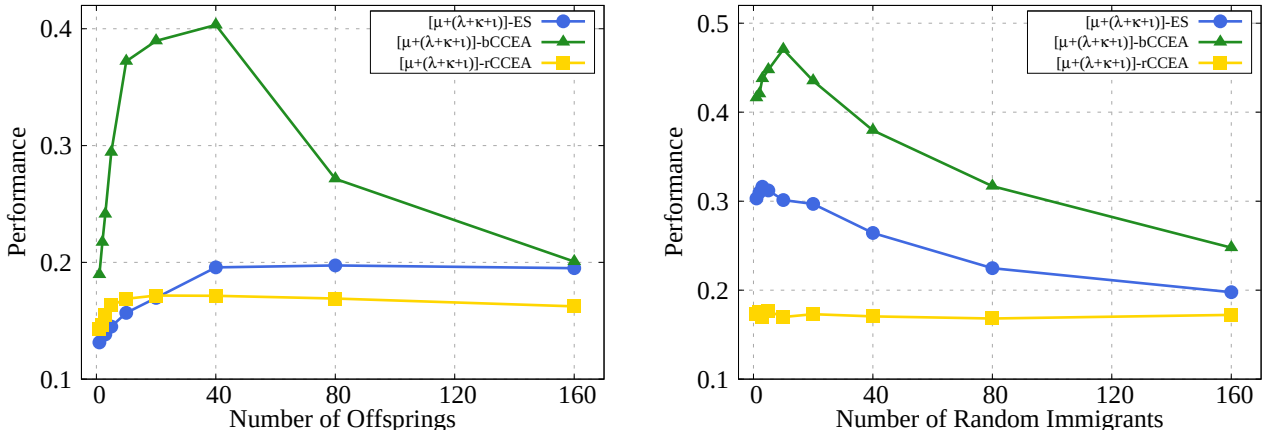

Figure 8.4: The median performance of the ES and the CCEAs on F_1 rotation problem having C_3 random step changes, when the change frequency τ is $100 \cdot n \cdot$FES. From top to bottom, the x-axis of the graphs represent offspring population sizes and RI population sizes.

bCCEA gives the best median performance over a wide range of the elitist population sizes ι from 2 to 160. One interesting observation is that increasing the number of the elitist individuals does not always provide a good performance. Rather, this increases the number of function evaluations as well as the number of the generations. This is similar to the cases in the last two subsections where increasing the population sizes will degrade the performance of the bCCEA. The best median performance of the ES and the bCCEA is all below 0.4 and 0.2 respectively. These two values are outperformed by the corresponding one in Table 8.4. This means that using elitist individuals alone is not as good as using both RI individuals and elitist individuals together. This further demonstrates the need to balance the exploration and the exploitation on the search space by generating these two types of individuals.

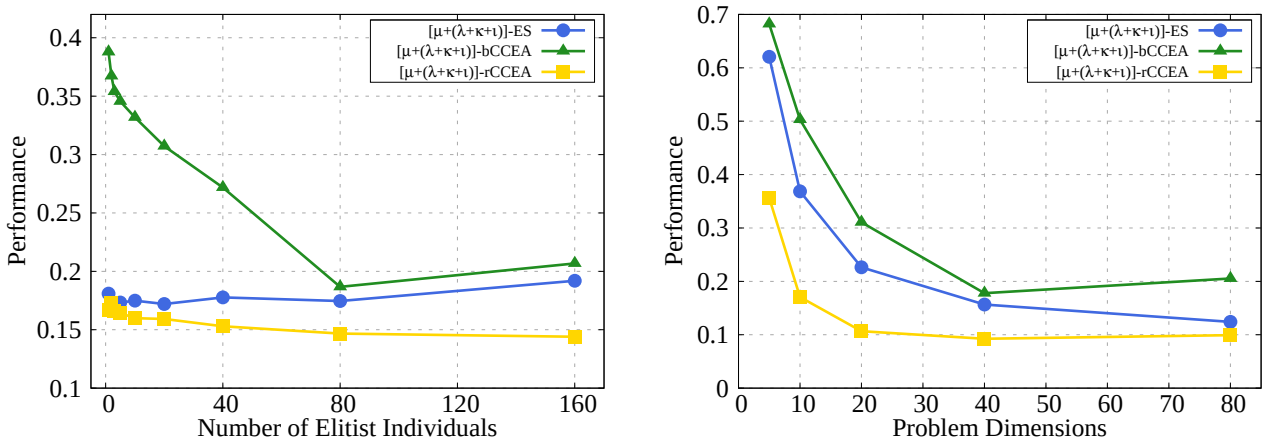

Figure 8.5: The median performance of the ES and the CCEAs on F_1 rotation problem having C_3 random step changes, when the change frequency τ is $100 \cdot n \cdot$ FES. From top to bottom, the x-axis of the graphs represent elitist population sizes and problem dimensions.

Problem dimensions n

The last set of experiments is to investigate the scalability of the algorithms when we increase the number of the problem dimensions. Figure 8.5 plots the median performance of the ES and the CCEAs against the problem dimensions $n \in \{5, 10, 20, 40, 80\}$ on F_1 rotation problem. The number of the offsprings, the RI individuals and the elitist individuals are 20, 10 and 10 respectively. Notice that the change frequency is still $100 \cdot n$ for all problem dimensions. From the results, the bCCEA is more scalable than the rCCEA. This is consistent with the previous set of experiments. Using the best collaboration method is more promising than using the random collaboration method. The performance of the bCCEA in high dimensional dynamic problems is promising and it is even more scalable than the ES in our simulations.

8.5 Future Perspective

We investigated the behaviour of a CCEA on the state-of-the-art dynamic optimization benchmarks. We first reviewed the background of the CCEAs for static optimization and then we discussed the four major approaches used in EAs to address the dynamic optimization. One major difference between the CCEA individuals and the EA individuals is that a CCEA individual has to collaborate with another $n-1$ CCEA individuals for its fitness evaluation. We formally discussed the two major collaboration methods used in a CCEA: 1) the best collaboration method in which a CCEA individual always chooses the best individual in terms of fitness, and 2) the random collaboration method in which a CCEA individual always randomly select other individuals and their fitness is not considered. The previous study shows that using the best collaborations in static optimization always yields the best performance. We extended this study under the context of the dynamic optimization and investigate if the choices for these two collaboration methods are same as those in the context of the static optimization. Our simulation results show that a CCEA using the best collaboration method outperforms a CCEA using the random collaboration method. The results are consistent for different types of the dynamic changes and the problem dimensions.

We also introduced two types of individuals for the CCEA to address dynamic optimization: 1) the RI individuals that increase the diversity of the populations in order to adapt to the changing environments quickly, and 2) the elitist individuals that increase the local convergence rate to the optima by generating individuals from the best individuals. Both of these types of individuals have to be used together in a CCEA because there is a need to balance the effect of the RI individuals and that of the elitist individuals. While the RI individuals increase

the diversity in exploring the search space that has not yet been explored by the current population, the elitist individuals exploit the search space by increasing the convergence rate to the local optima. Our experiments show that the use of the RI individuals and the elitist individuals not only improves a CCEA that outperforms the other CCEAs with the offsprings only, but it also improves an ES.

We also investigated the behaviour of a CCEA when it uses mutative σ-self adaptation for its mutation step sizes. Using self-adaptation in the context of the ES is common and well-developed. Therefore, it is natural to ask whether mutative σ-self adaptation in a CCEA is promising or not. Our results indicate that using self-adaptation in a CCEA does not appear to show any advantages, most of the cases using a fixed mutation size in a CCEA can have an equivalent performance. Besides studying the use of self-adaptation in a CCEA, we also compared the performance of a CCEA using plus selection and a CCEA using comma selection. Specifically, we were interested in understanding if the selection method can have any impact on the performance. Our results show that using plus selection is more promising than using comma selection; similar results were also observed in the standard ES.

Lastly we investigated the effect of using different population sizes in a CCEA, as we were interested in understanding whether the CCEAs are sensitive to population sizes. Obviously, setting large population sizes for individuals is not desirable in dynamic environments, because an increase in the population sizes means a decrease in the number of generations between dynamic changes. This will lower the number of selections and therefore decrease the number of the best individuals to be selected, significant for the CCEA, particularly when it uses the best collaboration method and when it uses elitist individuals. Both of them depend on the current best individuals in the

populations. The scalability of a CCEA is also reported. Comparing the bCCEA with the standard ES, its performance is marginally better than the standard ES. Therefore using the bCCEA in higher dimensional dynamic problems is promising.

□ **End of chapter.**

Chapter 9

Conclusion

If you're going through hell, keep going.

Winston Churchill

This thesis focused on the design of new sampling methods for optimization algorithms to solve continuous optimization problems. Continuous optimization problems have a wide range of applications in many disciplines and these problems are often hard to solve due to inherent difficulties such as a large dimensionality, multi-modality or other factors which make problems hard. The new sampling methods were proposed to improve the state-of-the-art optimization algorithms for continuous optimization problems. The resulting new algorithms are promising in the standard benchmark problems, encouraging further development of new sampling methods for optimization algorithms.

Another goal of this thesis was to study the state-of-the-art optimization algorithms in dynamic optimization problems. Dynamic optimization is an active area of research in the optimization community. However, there are still open questions. One of them is understanding the state-of-the-art optimization algorithms for dynamic optimization problems, as well as their capabilities and limits. In this thesis, we used the standard dynamic optimization benchmark and investigated two state-of-the-art evolutionary algorithms, CMA-ES and CCEA. Their

behavior in optimizing these dynamic problems was experimentally studied.

9.1 Summary of Contributions

In the first part of the thesis, we first discussed why sampling methods play an important part in evolutionary algorithms. Samples in evolutionary algorithms are always random and independent from each other. Evolutionary algorithms in optimization problems with problem-specific properties, which employ a degree of randomness as part of its logic, can be further improved by derandomizing their randomness. One popular method is to replace the independent samples by dependent ones. Samples for the new candidate solutions can be directly dependent on previous samples for previous candidate solutions which are of a good quality. We proposed two novel sampling methods in this thesis: halfspace sampling and eigenspace sampling.

In Chapter 4, we described how halfspace sampling works in a simple elitist (1+1) evolution strategies. In halfspace sampling, the supporting hyperplane going through a parent separates the search space into a positive halfspace and a negative halfspace. If an offspring lies in the negative halfspace, it will be reflected with respect to the parent so that it lies in the positive halfspace. We then derive theoretically the log-linear convergence of a scale-invariant step size (1+1)-ES with halfspace sampling on spherical functions in finite dimensions and infinite dimensions. The speed-up factor is derived theoretically when the optimal halfspace is used on spherical functions. We also introduced a new concept of evolution halfspaces. Evolution halfspaces accumulate information of the previous successful and unsuccessful steps so the optimal positive halfspace can be estimated. We also implemented the halfspace sampling in the state-of-the-art (1+1)-CMA-ES and the resulting algorithm was benchmarked

on the BBOB noise-free testbed. Our results showed that the use of halfspace sampling can improve a (1+1)-CMA-ES on some unimodal functions, and it does not worsen the algorithms.

Chapter 5 further extended the use of halfspace sampling in another evolutionary algorithm: Evolutionary Gradient Search (EGS). In basic EGS, random samples are used to estimate the true gradient vector with respect to a parent. In an EGS with halfspace sampling, we proposed to estimate the normal of the optimal halfspace in an iteration by computing the weight sum of random vectors with weights proportional to fitness of offspring. Any random samples that do not lie in the positive halfspace will be reflected with respect to the parent. We then proved the log-linear convergence of the scale-invariant step size EGS with and without halfspace sampling. The convergence rates were derived, expressed in terms of expectations of random variables and then numerically compared by means of Monte-Carlo simulations. The results show an improvement of 42% to 68% asymptotically when the dimension goes to infinity, regardless of the number of offspring used in the EGS.

Chapter 6 described the use of eigenspace sampling in the CMA-ES. Using eigenspace sampling reduces the number of function evaluations for learning the optimal covariance matrix in the algorithm. In eigenspace sampling, the minor eigenspace in the Hessian matrix of the underlying objective functions, which have repeated eigenvalues or clustered eigenvalues, is first identified. Instead of evaluating all the directions of the dominant eigenspaces, only a direction in the minor eigenspace is evaluated in an iteration. The resulting algorithm is evaluated on a set of ill-conditioned functions that have a few repeated eigenvalues in their Hessian matrices. A significant improvement was observed on functions that have one or two dominating eigenspaces. Eigenspace sampling does not improve CMA-ES on functions that have evenly distributed eigenspectra. Eigenspace

sampling was also implemented in the mirrored variant of $(1, \lambda)$-CMA-ES. When an unsuccessful step is found, the direction of the step is reversed but only the directions in the mirrored eigenspace are sampled. Using the mirrored eigenspace sampling method can improve a $(1, \lambda)$-CMA-ES in functions that have large dominant eigenspace.

The second part of this thesis focused on the empirical investigation of two evolutionary algorithms in dynamic environments. In dynamic environments, the objective function changes during the course of optimization. Chapter 7 considers three variants of CMA-ES including the elitist (1+1)-CMA-ES, the standard (μ,λ)-CMA-ES and the sep-(μ,λ)-CMA-ES. On dynamic problems, the elitist version of CMA-ES has the best performance. Comparing the conventional (1+1)-ES with one with a one-fifth success rule, (1+1)-CMA-ES is statistically the same as (1+1)-ES with one-fifth success rule. On the non-elitist algorithms, they do not appear to perform better than the elitist variants. However, when the dimensions of the problem is large, both non-elitist and elitist algorithms perform statistically the same. The empirical results suggest that using the elitist variant of CMA-ES for dynamic problems is more promising than using the non-elitist variants.

Chapter 8 studied the use of the cooperative coevolutionary algorithms (CCEA), which were commonly used for static large scale optimization problems, in the dynamic environments. The CCEA are investigated on the standard dynamic benchmark problems. Different settings of algorithms, including the use of plus or comma selection and the use of mutative step size adaptation, were experimentally compared. Two new individuals were proposed in CCEA, the elitist individuals and the random immigrant (RI) individuals. The elitist individuals were used for local performance of the algorithm to exploit the neighbourhood of the previous candidate solutions. The RI individuals

were used for global performance of the algorithm to explore the new region of the search space. The resulting algorithm with these two new individuals was experimentally compared with the evolution strategies using the same algorithms settings. The results show that CCEA perform better for most of the dynamic problems. The sensitivities of the population sizes for different individuals was also investigated, and the empirical results suggest that using a cooperative coevolutionary approach for dynamic problems is promising.

9.2 Summary of Future Directions

There are a few promising research directions to extend the work presented in this thesis. Some of them were already discussed in the corresponding Chapters and Sections. We can summarize them as follows:

- Halfspace sampling in Chapters 4 & 5 can be further extended to multi-membered evolution strategies with the use of recombinations.
- Chapter 4 proposed the concept of using evolution halfspaces to learn the optimal halfspaces. It basically computes the weighted sum of the successful and unsuccessful steps in the recent iterations to estimate the normal of the positive halfspaces. On the other hand, Chapter 5 proposed using the weighted sum of random vectors in an iteration to learn the optimal halfspaces. It will be interesting to compare these two methods experimentally and theoretically.
- Chapter 5 has proven theoretically the log-linear convergence of EGS with halfspace sampling and the improvements brought by halfspace sampling when the dimension goes to infinity. In practice, we plan to implement the halfspace sampling in the CMA-ES which basically an extended

version of an EGS with the use of the covariance matrix adaptation.

- Chapter 6 proposed the use of eigenspace sampling to learn the optimal covariance matrix in the CMA-ES and it was experimentally evaluated on standard benchmark functions that were ill-conditioned. It is still open to study whether the gain brought by eigenspace sampling is closely correlated to the eigenvalue distributions of the Hessian matrices of the underlying optimization functions. Theoretical investigations on the eigenspace sampling for evolution strategies is one of the challenging area for future research.

- Chapter 7 investigated the basic version of CMA-ES for dynamic problems. It is natural to extend CMA-ES by introducing it to the popular approaches for dynamic problems, including diversity-maintaining methods, multi-population method and memory-based methods.

- The CCEA in Chapter 8 is a basic version of CCEA. It will be interesting to investigate the use of decomposition approaches [238, 132, 156, 138], which are designed for static optimization, in dynamic problems.

□ **End of chapter.**

Bibliography

[1] Y. Akimoto, J. Sakuma, I. Ono, and S. Kobayashi. Functionally specialized cma-es: a modification of cma-es based on the specialization of the functions of covariance matrix adaptation and step size adaptation. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 479–486, New York, NY, USA, 2008. ACM. 55

[2] J. T. Alander. *An indexed bibliography of genetic algorithms: Years 1957-1993*. Art of CAD, 1994. 45

[3] E. Alba. *Parallel metaheuristics: a new class of algorithms*, volume 47. John Wiley & Sons, 2005. 17

[4] E. Alba and B. Sarasola. Measuring fitness degradation in dynamic optimization problems. In *Applications of Evolutionary Computation*, pages 572–581. Springer, 2010. 33

[5] V. S. Aragón and S. C. Esquivel. An evolutionary algorithm to track changes of optimum value locations in dynamic environments. *Journal of Computer Science & Technology*, 4, 2004. 28

[6] D. Arnold and R. Salomon. Evolutionary gradient search revisited. *Evolutionary Computation, IEEE Transactions on*, 11(4):480–495, Aug 2007. 105, 106

[7] D. V. Arnold. *Noisy optimization with evolution strategies*, volume 8. Springer, 2002. 21

[8] D. V. Arnold and H.-G. Beyer. *Random dynamics optimum tracking with evolution strategies*. Springer, 2002. 20

[9] D. V. Arnold and H.-G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1):135–159, 2003. 21

[10] D. V. Arnold and H.-G. Beyer. Optimum tracking with evolution strategies. *Evolutionary Computation*, 14(3):291–308, 2006. 20

[11] C.-K. Au and H. fung Leung. Eigenspace sampling in the mirrored variant of (1, λ)-cma-es. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012. 12

[12] C.-K. Au and H. fung Leung. Cooperative coevolutionary algorithms for dynamic optimization: An experimental study. *Evolutionary Intelligence*, 2014. 12

[13] C.-K. Au and H.-F. Leung. On the behavior of cooperative coevolution in dynamic environments. *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2827–2836, June 2008. 13, 164, 165

[14] C.-K. Au and H.-F. Leung. Investigating collaboration methods of random immigrant scheme in cooperative coevolution. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 2700–2707. IEEE Press, 2009. 13, 164, 165, 182

[15] C.-K. Au and H.-F. Leung. Improving cma-es by random evaluation on the minor eigenspace. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010. 12

[16] C.-K. Au and H.-F. Leung. An empirical comparison of cma-es in dynamic environments. In C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 529–538. Springer Berlin Heidelberg, 2012. 12

[17] C.-K. Au and H.-F. Leung. Halfspace sampling in evolution strategies. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 381–388, New York, NY, USA, 2014. ACM. 12, 105, 106

[18] A. Auger. Benchmarking the (1+ 1) evolution strategy with one-fifth success rule on the bbob-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2447–2452. ACM, 2009. 52

[19] A. Auger, D. Brockhoff, and N. Hansen. Analyzing the impact of mirrored sampling and sequential selection in elitist evolution strategies. In *FOGA*, pages 127–138, 2011. 87, 90, 91, 93

[20] A. Auger, D. Brockhoff, and N. Hansen. Analyzing the impact of mirrored sampling and sequential selection in elitist evolution strategies. In *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*, FOGA '11, pages 127–138, New York, NY, USA, 2011. ACM. 104

[21] A. Auger, D. Brockhoff, and N. Hansen. Mirrored sampling in evolution strategies with weighted recombination. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 861–868, New York, NY, USA, 2011. ACM. 139, 140, 146

[22] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1777–1784 Vol. 2, Sept. 2005. 55

[23] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776 Vol. 2, Sept. 2005. 55, 61

[24] A. Auger and N. Hansen. Reconsidering the progress rate theory for evolution strategies in finite dimensions. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 445–452. ACM, 2006. 46

[25] A. Auger and N. Hansen. Reconsidering the progress rate theory for evolution strategies in finite dimensions. In *GECCO*, pages 445–452, 2006. 80, 109

[26] A. Auger, M. Schoenauer, and N. Vanhaecke. Ls-cma-es: A second-order algorithm for covariance matrix adaptation. In *PPSN*, pages 182–191, 2004. 126

[27] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996. 53

[28] T. Back. On the behavior of evolutionary algorithms in dynamic environments. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 446–451. IEEE, 1998. 28, 32, 33

[29] T. Back. On the behavior of evolutionary algorithms in dynamic environments. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational*

Intelligence., The 1998 IEEE International Conference on, pages 446 –451, may 1998. 157

[30] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997. 17, 45

[31] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*, volume 1. CRC Press, 2000. 17, 45

[32] T. Bäck and M. Schütz. Evolution strategies for mixed-integer optimization of optical multilayer systems. In *Evolutionary Programming*, pages 33–51, 1995. 47

[33] N. A. Barricelli. Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9(35-36):143–182, 1957. 45

[34] N. A. Barricelli et al. Esempi numerici di processi di evoluzione. *Methodos*, 6(21-22):45–68, 1954. 45

[35] R. Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166, 1992. 2

[36] R. Bellman, R. E. Bellman, R. E. Bellman, and R. E. Bellman. *Adaptive control processes: a guided tour*, volume 4. Princeton university press Princeton, 1961. 18

[37] R. Bellman and R. Corporation. *Dynamic Programming*. Rand Corporation research study. Princeton University Press, 1957. 18

[38] D. P. Bertsekas. Constrained optimization and lagrange multiplier methods. *Computer Science and Applied Mathematics, Boston: Academic Press, 1982*, 1, 1982. 16

[39] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995. 46

[40] D. Bertsimas and R. Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005. 16

[41] H. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, 2001. 52, 72

[42] H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1995. 52

[43] P. A. Bosman. Learning and anticipation in online dynamic optimization. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 129–152. Springer, 2007. 28, 31

[44] P. A. Bosman and H. La Poutre. Learning and anticipation in online dynamic optimization with evolutionary algorithms: the stochastic case. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1165–1172. ACM, 2007. 31

[45] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.*, 3:–1882 Vol. 3, 1999. 30, 35, 173

[46] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2001. 28, 29, 33, 157, 164

[47] J. Branke, T. Kausler, C. Schmidt, and H. Schmeck. A multi-population approach to dynamic optimization prob-

lems. In *In Adaptive Computing in Design and Manufacturing*, pages 299–308. Springer, 2000. 31

[48] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In *Advances in evolutionary computing*, pages 239–262. Springer, 2003. 33

[49] H. J. Bremermann. *The evolution of intelligence: The nervous system as a model of its environment*. University of Washington, Department of Mathematics, 1958. 45

[50] H. J. Bremermann. Optimization through evolution and recombination. *Self-organizing systems*, pages 93–106, 1962. 45

[51] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *Evolutionary Computation, IEEE Transactions on*, 10(6):646–657, 2006. 49

[52] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm. Mirrored sampling and sequential selection for evolution strategies. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part I*, PPSN'10, pages 11–21, Berlin, Heidelberg, 2010. Springer-Verlag. 104

[53] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm. Mirrored sampling and sequential selection for evolution strategies. In *Proceedings of the 11th international conference on Parallel problem solving from nature: Part I*, PPSN'10, pages 11–21, Berlin, Heidelberg, 2010. Springer-Verlag. 139, 146

[54] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. 20

[55] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 2. the new algorithm. *IMA Journal of Applied Mathematics*, 6(3):222–231, 1970. 20

[56] L. Bull. Evolutionary computing in multi-agent environments: Operators. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, EP '98, pages 43–52. Springer-Verlag, 1998. 63

[57] L. Bull, T. C. Fogarty, and M. Snaith. Evolution in multi-agent systems: Evolving communicating classifier systems for gait in a quadrupedal robot. In *ICGA*, pages 382–388, 1995. 62

[58] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, 1990. 30, 32, 33, 164

[59] H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 523–530. Morgan Kaufmann Publishers Inc., 1993. 32, 164

[60] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009. 2, 17

[61] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In *Proceedings of the First International Conference on Genetic Algorithms*, pages 183–187, 1985. 48

[62] G. B. Dantzig. *Linear programming and extensions*. Princeton university press, 1965. 46

[63] S. Das and P. Suganthan. Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems. *Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India*, 2010. 24

[64] D. Dasgupta and D. R. McGregor. Nonstationary function optimization using the structured genetic algorithm. In *Parallel Problem Solving from Nature II*, pages 147–156, 1992. 30

[65] K. A. De Jong. *Evolutionary computation: a unified approach*. MIT press, 2006. 17, 45

[66] K. Deb. A population-based algorithm-generator for real-parameter optimization. *Soft Computing*, 9(4):236–253, 2005. 47

[67] K. Deb et al. *Multi-objective optimization using evolutionary algorithms*, volume 2012. John Wiley & Sons Chichester, 2001. 21

[68] M. Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992. 48

[69] M. Dorigo and M. Birattari. Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer, 2010. 17, 48

[70] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006. 48

[71] M. Eigen. *Ingo Rechenberg Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. mit einem Nachwort von Manfred Eigen, Friedrich Frommann Verlag, Struttgart-Bad Cannstatt, 1973. 45, 46, 47, 50

[72] R. Eriksson and B. Olsson. Cooperative coevolution in inventory control optimisation. In *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms, University of East Anglia*, pages 583–587. Springer, 1997. 62

[73] W. Feng, T. Brune, L. Chan, M. Chowdhury, C. K. Kuek, and Y. Li. Benchmarks for testing evolutionary algorithms. In *Asia-Pacific Conference on Control and Measurement*, pages 134–138, 1998. 33

[74] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009. Updated February 2010. 98

[75] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970. 20

[76] C. A. Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Marcombo, 1995. 16

[77] D. B. Fogel. *Evolutionary computation: the fossil record*. Wiley-IEEE Press, 1998. 45

[78] D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons, 2006. 17, 45

[79] L. J. Fogel, A. J. Owens, and M. J. Walsh. Artificial intelligence through simulated evolution. 1966. 46

[80] A. S. Fraser. Simulation of genetic systems by automatic digital computers vi. epistasis. *Australian Journal of Biological Sciences*, 13(2):150–162, 1960. 45

[81] A. Gasper and P. Collard. From gas to artificial immune systems: improving adaptation in time dependent optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999. 32, 33

[82] P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM journal on optimization*, 12(4):979–1006, 2002. 22

[83] F. Glover and G. A. Kochenberger. *Handbook of metaheuristics*. Springer, 2003. 17

[84] C.-K. Goh and K. Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 13(1):103–127, 2009. 33

[85] D. E. Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989. 45

[86] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithm with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 59–68. Lawrence Erlbaum Associates, Inc., 1987. 31

[87] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970. 20

[88] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985. 16

[89] J. J. Grefenstette. Genetic algorithms for changing environments. In R. Manner and B. Manderick, editors, *Parallel Problem Solving from Nature II*, pages 137–144. Elsevier, 1992. 30, 32, 33, 164

[90] J. J. Grefenstette. Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999. 32, 33

[91] I. Griva, S. G. Nash, and A. Sofer. *Linear and nonlinear optimization*. Siam, 2009. 2

[92] N. Hansen. An analysis of mutative σ-self-adaptation on linear fitness functions. *Evolutionary Computation*, 14(3):255–275, 2006. 52

[93] N. Hansen. The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, 2006. 54

[94] N. Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, pages 2389–2396, New York, NY, USA, 2009. ACM. 55

[95] N. Hansen. Benchmarking a bi-population cma-es on the bbob-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2389–2396. ACM, 2009. 61

[96] N. Hansen. A cma-es for mixed-integer nonlinear optimization. 2011. 47

[97] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA, 2012. 98

[98] N. Hansen, A. Auger, S. Finck, R. Ros, et al. Real-parameter black-box optimization benchmarking 2010: Experimental setup. 2010. 25, 26, 27, 43

[99] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009. Updated February 2010. 98

[100] N. Hansen, S. Finck, R. Ros, A. Auger, et al. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. 2009. 25

[101] N. Hansen, S. Finck, R. Ros, A. Auger, et al. Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. 2009. 25

[102] N. Hansen and S. Kern. Evaluating the cma evolution strategy on multimodal test functions. In *PPSN*, pages 282–291, 2004. 53, 125

[103] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strat-

egy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003. 6, 20, 53

[104] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003. 53, 55, 125, 131, 145, 147, 156, 158

[105] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001. 6, 20, 53

[106] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. 53, 55, 125, 131, 145, 147, 156, 158

[107] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. 73

[108] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. *Applied Soft Computing*, 11(8):5755–5769, 2011. 17

[109] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, et al. Pso facing non-separable and ill-conditioned problems. 2008. 17

[110] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975. 3, 46, 47

[111] R. Hooke and T. A. Jeeves. "direct search"solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961. 17

[112] H. H. Hoos and T. Stützle. Evaluating las vegas algorithms: pitfalls and remedies. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 238–245. Morgan Kaufmann Publishers Inc., 1998. 27

[113] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004. 2

[114] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 453–460, New York, NY, USA, 2006. ACM. 55

[115] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+1)-cma for evolution strategies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, pages 453–460, 2006. 59, 73, 94, 97, 156, 158

[116] G. Jastrebski and D. Arnold. Improving evolution strategies through active covariance matrix adaptation. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2814–2821, 0-0 2006. 55

[117] M. Jebalia, A. Auger, and P. Liardet. Log-linear convergence and optimal bounds for the (1+1)-es. In *Artificial Evolution*, pages 207–218, 2007. 80, 109

[118] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evo-*

lutionary Computation, 9(3):303–317, June 2005. 20, 30, 157

[119] Y. Jin and B. Sendhoff. Constructing dynamic optimization test problems using the multi-objective optimization concept. In *Applications of Evolutionary Computing*, pages 525–536. Springer, 2004. 35

[120] J. Kennedy, R. Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995. 17, 49

[121] S. Kern, S. D. Müller, N. Hansen, D. Büche, J. Ocenasek, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms– a comparative review. *Natural Computing: an international journal*, 3(1):77–112, 2004. 54, 125

[122] J. N. Knight and M. Lunacek. Reducing the space-time complexity of the cma-es. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 658–665, New York, NY, USA, 2007. ACM. 55

[123] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482, 2003. 17

[124] J. R. Koza. Concept formation and decision tree induction using the genetic programming paradigm. In *Parallel Problem Solving from Nature*, pages 124–128. Springer, 1991. 48

[125] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992. 45, 48, 49

[126] P. Larrañaga and J. A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer, 2002. 48

[127] E. L. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley New York, 1985. 16

[128] C. Li and S. Yang. A generalized approach to construct benchmark problems for dynamic optimization. In *SEAL '08: Proceedings of the 7th International Conference on Simulated Evolution and Learning*, pages 391–400. Springer-Verlag, 2008. 34, 35, 40, 43, 165, 173

[129] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H. g. Beyer, and P. N. Suganthan. Benchmark generator for cec 2009 competition on dynamic optimization, 2008. 35, 40, 43, 158, 165, 173, 175

[130] R. Li, M. T. Emmerich, E. G. Bovenkamp, J. Eggermont, T. Bäck, J. Dijkstra, and J. H. Reiber. Mixed-integer evolution strategies and their application to intravascular ultrasound image analysis. In *Applications of Evolutionary Computing*, pages 415–426. Springer, 2006. 47

[131] R. Li, M. T. Emmerich, J. Eggermont, T. Bäck, M. Schütz, J. Dijkstra, and J. H. Reiber. Mixed integer evolution strategies for parameter optimization. *Evolutionary computation*, 21(1):29–64, 2013. 47

[132] X. Li and X. Yao. Cooperatively coevolving particle swarms for large scale optimization. *Evolutionary Computation, IEEE Transactions on*, 16(2):210–224, 2012. 69, 199

[133] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212, 2013. 24

[134] J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. C. Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41, 2006. 24

[135] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 22

[136] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6):448–462, 2005. 49

[137] R. Mallipeddi and P. N. Suganthan. Problem definitions and evaluation criteria for the cec 2010 competition on constrained real-parameter optimization. *Nanyang Technological University, Singapore*, 2010. 24

[138] Y. Mei, X. Li, and X. Yao. Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems. 2013. 69, 199

[139] S. Meyer-Nieberg and H.-G. Beyer. Mutative self-adaptation on the sharp and parabolic ridge. In *Foundations of Genetic Algorithms*, pages 70–96. Springer, 2007. 52

[140] S. Meyer-Nieberg and H.-G. Beyer. Self-adaptation in evolutionary algorithms. In *Parameter setting in evolutionary algorithms*, pages 47–75. Springer, 2007. 52

[141] Z. Michalewicz and D. B. Fogel. *How to solve it: modern heuristics*. Springer, 2004. 43

[142] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the feedback thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature?PPSN V*, pages 149–158. Springer, 1998. 33

[143] R. Morrison and K. De Jong. A test problem generator for non-stationary environments. *Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99.*, 3:–2053 Vol. 3, 1999. 35, 164, 173

[144] R. Morrison and K. De Jong. Triggered hypermutation revisited. *Proceedings of the 2000 Congress on Evolutionary Computation, 2000*, 2:1025–1032 vol.2, 2000. 30

[145] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. SpringerVerlag, 2004. 28, 29, 164

[146] R. W. Morrison and K. A. De Jong. Measurement of population diversity. In *Artificial Evolution*, pages 31–41. Springer, 2002. 33

[147] H. Muehlenbein and T. Mahnig. Evolutionary computation and wright's equation. *Theoretical Computer Science*, 287(1):145–165, 2002. 48

[148] V. Mutseniyeks and L. Rastrigin. Extremal control of continuous multi-parameter systems by the method of random search. *Engineering Cybernetics*, 1:82–90, 1964. 46

[149] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965. 2

[150] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988. 16

[151] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994. 46

[152] T. T. Nguyen. *Continuous dynamic optimisation using evolutionary algorithms*. PhD thesis, University of Birmingham, 2011. 33

[153] T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012. 30, 31, 32, 43

[154] T. T. Nguyen and X. Yao. Benchmarking and solving dynamic constrained problems. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 690–697. IEEE, 2009. 32, 33

[155] T. T. Nguyen and X. Yao. Dynamic time-linkage problems revisited. In *Applications of Evolutionary Computing*, pages 735–744. Springer, 2009. 31

[156] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. Cooperative co-evolution with differential grouping for large scale optimization. 2014. 69, 199

[157] F. Oppacher, M. Wineberg, et al. The shifting balance genetic algorithm: Improving the ga in a dynamic environment. In *Proceedings of the genetic and evolutionary computation conference*, volume 1, pages 504–510. Citeseer, 1999. 33

[158] I. H. Osman and J. P. Kelly. *Meta-heuristics: theory and applications*. Springer, 1996. 17

[159] L. Panait and S. Luke. Time-dependent collaboration schemes for cooperative coevolutionary algorithms. In *Proceedings of the AAAI 2005 Fall Symposium on Coevolutionary and Coadaptive Systems*. AAAI Press, 2005. 64

[160] L. Panait, S. Luke, and J. F. Harrison. Archive-based cooperative coevolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 345–352. ACM, 2006. 64

[161] L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, Dec. 2006. 62, 164

[162] L. Panait, R. P. Wiegand, and S. Luke. A visual demonstration of convergence properties of cooperative coevolution. In *PPSN*, pages 892–901, 2004. 64

[163] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998. 16

[164] M. Patriksson, N. Andreasson, A. Evgrafov, E. Gustavsson, and M. Önnheim. *Introduction to Continuous Optimization*. Studentlitteratur, 2013. 2

[165] M. E. H. Pedersen. Good parameters for differential evolution. *Magnus Erik Hvass Pedersen*, 2010. 49

[166] M. E. H. Pedersen. *Tuning & simplifying heuristical optimization*. PhD thesis, University of Southampton, 2010. 49

[167] H.-O. Peitgen. *Newton's method and dynamical systems*. Springer, 1989. 2

[168] E. Popovici. *An analysis of two-population coevolutionary computation*. PhD thesis, 2006. Adviser-Jong, Kenneth De. 64

[169] E. Popovici and K. De Jong. Sequential versus parallel cooperative coevolutionary algorithms for optimization. *IEEE Congress on Evolutionary Computation, 2006. CEC 2006.*, pages 1610–1617, 0-0 0. 64

[170] E. Popovici and K. De Jong. The effects of interaction frequency on the optimization performance of cooperative coevolution. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 353–360. ACM, 2006. 64

[171] E. Popovici and K. D. Jong. The dynamics of the best individuals in co-evolution. *Natural Computing*, 5(3):229–255, 2006. 62, 64, 164

[172] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature?PPSN III*, pages 249–257. Springer, 1994. 6, 48

[173] M. A. Potter and K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000. 48, 61, 62, 163

[174] M. A. Potter and K. A. D. Jong. The coevolution of antibodies for concept learning. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 530–539. Springer-Verlag, 1998. 62

[175] M. A. Potter, K. A. D. Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 366–372. Morgan Kaufmann Publishers Inc., 1995. 62

[176] M. A. Potter, L. Meeden, and A. C. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: the emergence of specialists. In *IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence*, pages 1337–1343. Morgan Kaufmann Publishers Inc., 2001. 62

[177] M. J. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964. 2

[178] M. J. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In *Numerical analysis*, pages 144–157. Springer, 1978. 16

[179] K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer, 2006. 49

[180] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1785–1791. IEEE, 2005. 49

[181] W. Rand and R. Riolo. Measurements for understanding the behavior of the genetic algorithm in dynamic environments: A case study using the shaky ladder hyperplane-defined functions. In *Proceedings of the 2005 workshops on Genetic and evolutionary computation*, pages 32–38. ACM, 2005. 33

[182] L. Rastrigin. Extremal control by the method of random scanning. *Automation and Remote Control*, 21:891–896, 1960. 45, 46

[183] L. Rastrigin. Convergence of random search method in extremal control of many-parameter system. *Automation and Remote Control*, 24(11):1337, 1964. 45, 46

[184] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, TU Berlin, 1971. 3, 59, 158

[185] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973. 72

[186] H. Richter. Evolutionary optimization and dynamic fitness landscapes. In *Evolutionary Algorithms and Chaotic Systems*, pages 409–446. Springer, 2010. 32

[187] P. Rohlfshagen and X. Yao. Attributes of dynamic combinatorial optimisation. In *Simulated Evolution and Learning*, pages 442–451. Springer, 2008. 28

[188] P. Rohlfshagen and X. Yao. Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. *Soft Computing*, 15(9):1723–1734, 2011. 28

[189] R. Ros and N. Hansen. A simple modification in cma-es achieving linear time and space complexity. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, pages 296–305, Berlin, Heidelberg, 2008. Springer-Verlag. 55, 58, 156, 158

[190] C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Compututation*, 5(1):1–29, 1997. 61, 163

[191] F. Rothlauf. *Design of modern heuristics: principles and application*. Springer, 2011. 43

[192] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004. 48

[193] R. Salomon and D. Arnold. The evolutionary-gradient-search procedure in theory and practice. In R. Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*, pages 77–101. 2009. 105, 106

[194] R. Salomon and P. Eggenberger. Adaptation on the evolutionary time scale: A working hypothesis and basic experiments. In *Artificial Evolution*, pages 251–262. Springer, 1998. 33

[195] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. 16

[196] M. Schumer and K. Steiglitz. Adaptive step size random search. *Automatic Control, IEEE Transactions on*, 13(3):270–276, 1968. 46

[197] H. Schwefel. *Evolution and Optimum Seeking*. A Wiley-Interscience publication. Wiley, 1995. 104

[198] H.-P. Schwefel. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik. *Master's thesis, Hermann Föttinger Institute for Hydrodynamics, Technical University of Berlin*, 1965. 3, 45, 50

[199] H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer ver-*

gleichenden Einführung in die Hill-Climbing-und Zufallsstrategie. Birkhäuser, 1977. 3, 51

[200] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc., 1993. 47, 53

[201] L. Schnemann. Evolution strategies in dynamic environments. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 51–77, 2007. 157

[202] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970. 20

[203] M. Shi and H. Wu. Pareto cooperative coevolutionary genetic algorithm using reference sharing collaboration. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 867–874. ACM, 2009. 64

[204] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European journal of operational research*, 185(3):1155–1173, 2008. 48

[205] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983. 22

[206] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12. Springer, 2002. 2

[207] R. Storn. On the usage of differential evolution for function optimization. In *Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American*, pages 519–523. IEEE, 1996. 49

[208] R. Storn and K. Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995. 49

[209] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997. 49

[210] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL Report*, 2005005, 2005. 24

[211] E.-G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009. 17

[212] K. Tang, X. Yáo, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang. Benchmark functions for the cec?2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, 2007. 24

[213] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on optimization*, 7(1):1–25, 1997. 2

[214] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999. 32, 34

[215] P. J. Van Laarhoven and E. H. Aarts. *Simulated annealing*. Springer, 1987. 17

[216] K. Weicker. An analysis of dynamic severity and population size. In *Parallel Problem Solving from Nature PPSN VI*, pages 159–168. Springer, 2000. 28

[217] K. Weicker. Performance measures for dynamic environments. In *Parallel Problem Solving from Nature?PPSN VII*, pages 64–73. Springer, 2002. 33

[218] K. Weicker. *Evolutionary algorithms and dynamic optimization problems*. Der Andere Verlag Berlin, 2003. 29

[219] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999. 33

[220] K. Weicker and N. Weicker. On evolution strategy optimization in dynamic environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 3 vol. (xxxvii+2348), 1999. 157

[221] K. Weicker and N. Weicker. Dynamic rotation and partial visibility. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 1125–1131, 2000. 28

[222] P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, 2004. 62, 63, 164

[223] R. P. Wiegand, W. C. Liles, and K. A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1235–1242. Morgan Kaufmann, 7-11 2001. 62, 63, 64

[224] R. P. Wiegand, W. C. Liles, and K. A. D. Jong. The effects of representational bias on collaboration methods in cooperative coevolution. In *PPSN VII: Proceedings of the*

7th International Conference on Parallel Problem Solving from Nature, pages 257–270. Springer-Verlag, 2002. 62

[225] M. Wineberg and F. Oppacher. Enhancing the ga's ability to cope with dynamic environments. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 3–10, 2000. 31

[226] Y. G. Woldesenbet and G. G. Yen. Dynamic evolutionary algorithm with variable relocation. *Evolutionary Computation, IEEE Transactions on*, 13(3):500–513, 2009. 28

[227] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997. 4

[228] A. H. Wright et al. Genetic algorithms for real parameter qptimization. In *FOGA*, pages 205–218. Citeseer, 1990. 47

[229] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 3, pages 2246–2253. IEEE, 2003. 35

[230] S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1115–1122. ACM, 2005. 168

[231] S. Yang. Genetic algorithms with elitism-based immigrants for changing optimization problems. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog: Applications of Evolutionary Computing*, pages 627–636. Springer-Verlag, 2007. 168

[232] S. Yang. Genetic algorithms with memory-and elitism-based immigrants in dynamic environments. *Evol. Comput.*, 16(3):385–416, 2008. 33

[233] S. Yang, Y.-S. Ong, and Y. Jin. *Evolutionary Computation in Dynamic and Uncertain Environments (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., 2007. 29, 164

[234] S. Yang and X. Yao. Dual population-based incremental learning for problem optimization in dynamic environments. In *7th Asia Pacific Symposium on Intelligent and Evolutionary Systems: 49-56, 2003*, 2003. 34

[235] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Comput.*, 9(11):815–834, 2005. 31, 35

[236] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *Evolutionary Computation, IEEE Transactions on*, 12(5):542 –561, oct. 2008. 30, 35

[237] S. Yang and X. Yao. *Evolutionary Computation for Dynamic Optimization Problems (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., 2013. 29, 31, 32, 43

[238] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Inf. Sci.*, 178(15):2985–2999, 2008. 69, 199

[239] X. Yu, K. Tang, T. Chen, and X. Yao. Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. *Memetic Computing*, 1(1):3–24, 2009. 168

[240] M. Zeleny and J. L. Cochrane. *Multiple criteria decision making*, volume 25. McGraw-Hill New York, 1982. 21